Integrated Circuit White Space Redistribution for Temperature Optimization

Yuankai Chen†, Hai Zhou†, and Robert P. Dick‡
† EECS Department, Northwestern University, Evanston, IL, U.S.A.
‡ EECS Department, University of Michigan, Ann Arbor, MI, U.S.A.

***Abstract*—Thermal problems are important for integrated circuits with high power densities. Three-dimensional stacked-wafer integrated circuit technology reduces interconnect lengths and improves performance compared to two-dimensional integration. However, it intensifies thermal problems. One remedy is to redistribute white space during floorplanning. In this paper, we propose a two-phase algorithm to redistribute white space. In the first phase, the lateral heat flow white space redistribution problem is formulated as a minimum cycle ratio problem, in which the maximum power density is minimized. Since this phase only considers lateral heat flow, it also works for traditional two-dimensional integrated circuits. In the second phase, to consider inter-layer heat flow in three-dimensional integrated circuits, we discretize the chip into an array of tiles and use a dynamic programming algorithm to minimize the maximum stacked tile power consumption. We compared our algorithms with a previously proposed technique based on mathematical programming. Our iterative minimum cycle ratio algorithm achieves 35% more reduction in peak temperature. Our two-phase algorithm achieves 4.21× reduction in peak temperature for three-dimensional integrated circuits compared to applying the first phase, alone.**

I. Introduction

Increasing power densities are making thermal problems more important during integrated circuits (IC) design. Three-dimensional (3D) ICs, which are commonly composed of vertically stacked dies, have the potential to improve integration density and reduce interconnect delay [1]. However, 3D IC designers face a major challenge in controlling temperature due to the high power densities possible in stacked-wafer integration [1], [2]; thermal issues are a central concern during 3D IC design. We consider the problem of automatically distributing white space among blocks to reduce the peak temperature in 2D and 3D ICs. The proposed technique can work with existing 2D/3D floorplanning algorithms to help to optimize chip temperature.

Many existing floorplanning algorithms consider temperature but do not explicitly optimize the insertion of white space to control temperature. In some search-based algorithms such as simulated annealing [3], [4], the peak temperature is included in the cost function. Zhou et al. [5] developed a force-directed algorithm for 3D floorplanning. It treats thermal gradient as a force on blocks to repel them from high-temperature regions, but does not currently use thermal gradient information to insert white space. A temperature-driven white space redistribution algorithm can be used to appropriately add white space, either by embedding it within an existing floorplanner or by using it in a post-processing step.

2D and 3D white space redistribution algorithms have been developed for different goals. Tang, Tian, and Wong [6] formulate white space redistribution as a min-cost flow problem with the objective of minimizing half-perimeter wirelength.

This work was supported in part by SRC under awards 2009-HJ-2024 and 2007-HJ-1593, and in part by NSF under awards CCF-0811270, CCF-0964763, and CNS-0347941.

978-3-9810801-7-9/DATE11/©2011 EDAA

Researchers have also considered 3D IC white space redistribution. Li et al. [7] describe an integrated white space and thermal via planning algorithm. In each iteration, the algorithm computes the amount of white space around a each block required for temperature-constraining thermal vias. Then, white space is budgeted among blocks using linear programming. The objective is to minimize the sum of required area minus actually assigned area. However this objective does not prevent scenarios in which one block is allocated too much white space and another allocated too little white space.

Wong and Lim [8] describe a mathematical programming formulation that iteratively repels blocks from hot tiles. They only consider the closest block for each tile, and may ignore blocks with much higher power densities that are only slightly farther away. Their formulation also contains numerous non-linear constraints, increasing computational cost.

Yan et al. [9] describe a method to iteratively insert white space to the block with the highest temperature. They use block-level spatial discretization during temperature computation, which may be inappropriate because it neglects intra-block temperature variation. They insert white space evenly at the four boundaries of the block, although in reality an uneven insertion may be better.

These existing techniques consider only the heat flow among blocks on the same layer, i.e., they consider lateral heat flow but neglect the stronger vertical inter-block thermal relationships in 3D ICs. Temperature is governed by the following differential equation:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = -\frac{p}{k}, \tag{1}$$

where T is the spatial thermal profile, p is power density profile, and k is thermal conductivity. The first two terms on the left-hand side represent lateral heat flow and the third term represents vertical heat flow.

Thermal problems are especially important in 3D ICs because blocks on different layers may overlap. In the overlap region, the power sources on different layers all contribute to the temperature in layers farther from the heat sink. Therefore, in order to achieve lower peak temperature, vertical heat flow must be considered.

The contribution of our work is a two-phase algorithm that considers both lateral and vertical heat flow. In the first phase, the lateral heat flow problem is formulated as a minimum cycle ratio problem that is solved optimally and efficiently. The second phase considers vertical heat flow. We discretize the chip area into tiles and use a dynamic programming algorithm to optimize tile stacked power consumptions.

II. Lateral Heat Flow Optimization with Minimum Cycle Ratio Algorithm

This section describes our problem formulation and solution for white space insertion considering lateral thermal interaction. We generalize to inter-layer thermal interaction in Section III.

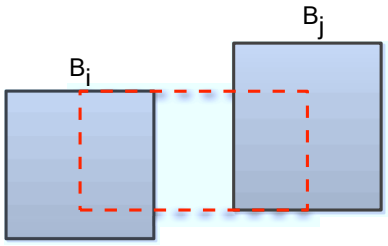
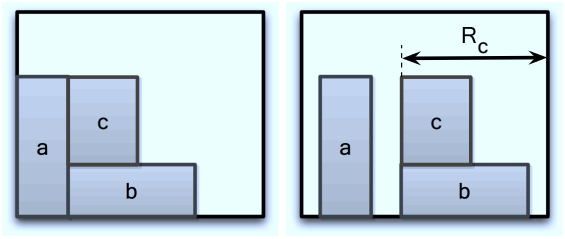


Fig. 1. Illustration of rectangular window for average power density calculation.



(a) (b)

Fig. 2. Example block relationships and remaining white space flexibility.

II.A. Problem Formulation

A stacked-layer 3D IC consists of L layers, $D_1, D_2, \ldots, D_L$. Blocks in a layer are placed in the x–y plane and layers are stacked on the z axis. In each layer D_k, there is a set of rectangular blocks $B_1^k, B_2^k, \ldots, B_{n_k}^k$. Each block B_i^k has a fixed shape (w_i^k, h_i^k), where w_i^k and h_i^k are width and height. The tuple (x_i^k, y_i^k) represents the coordinates of the center of block B_i^k. Each block B_i^k is also assigned a fixed power consumption P_i^k. When referring to blocks on the same layer, we omit the layer index in our notations. The dimensions of stacked layers are given by W (width) and H (height). The topology of each layer, i.e., the relative locations of blocks, is described by constraint graphs generated by a floorplanner, e.g., a sequence pair [10].

Net heat flow proceeds laterally from power sources to surrounding lower-temperature regions. Inserting white space around a particular block increases the area available for its heat to spread laterally, which eventually flows vertically toward the heat sink. Our problem in this phase is to assign more white space to blocks with higher power densities and less white space to blocks with lower power densities for a given block topology.

Our approach requires the use of a closed-form expression for estimated temperature within our MCR formulation. This makes it challenging to use conventional thermal analysis algorithms. To support rapid inner-loop estimation, we estimate the impact of local design decisions on local thermal profile with an expression based on *average power density*, i.e., the total power consumption of a chosen region divided by its area. We treat x and y directions separately in redistribution. Taking x direction for example, given a layer and the topology of blocks on this layer, if blocks B_i and B_j are adjacent on the x-axis, we can draw a rectangular power density window spanning the regions between the midpoints of these two blocks. We use x_1, x_2, y_1 and y_2 to represent coordinates of left, right, lower, and upper edges of this window. We have $x_1 = x_i$, $x_2 = x_j$, $y_1 = \max(y_i - h_i, y_j - h_j)$ and $y_2 = \min(y_i + h_i, y_j + h_j)$. Figure 1 gives an example of two blocks and their power density window, which is represented by the dashed rectangle. Within the window B_i and B_j, are the only heat sources. When we redistribute white space on the x-axis, the average power density of this window is primarily affected by the x-axis separation between these blocks. Since the height of the rectangle does not change during x-axis white space redistribution, we can approximate thermal impact with power density, i.e., the total power consumption in the window divided by its width. The total power consumption in this window also depends on y-axis block positions. In order to separate the x- and y-axis white space insertion problems, we use $P_i/2 + P_j/2$ for the power consumption within the window, which gives an upper bound the power density in x direction:

$$pd_{ij}^x = (P_i/2 + P_j/2)/(x_j - x_i). \tag{2}$$

We define the linear power density between a block and left (right) boundary of the layout as follows:

$$pd_{iL}^x = (P_i/2)/x_i \text{ and} \tag{3}$$
$$pd_{iR}^x = (P_i/2)/(W - x_i). \tag{4}$$

Our objective is to minimize the maximum pd_{ij}^x, which is equivalent to maximizing the minimum reciprocal of pd_{ij}^x. Let $l_x = \min 1/pd_{ij}^x$. We formulate the Maximum Linear Power Density Minimization (MLPDM) problem as follows.

Problem MLPDM: Given a topology of a set of blocks and a fixed chip area, our objective function is

$$\max l_x. \tag{5}$$

The following constraints are considered: $\forall i \in 1...n$

$$x_i - \frac{w_i}{2} \geq 0 \text{ and} \tag{6}$$
$$x_i + \frac{w_i}{2} \leq W. \tag{7}$$

If block i is to the immediate left of block j,

$$x_j - x_i \geq \frac{w_i}{2} + \frac{w_j}{2} \text{ and} \tag{8}$$
$$\frac{x_j - x_i}{P_i/2 + P_j/2} \geq l_x. \tag{9}$$

Constraints (6) and (7) state that all blocks should be placed within fixed chip area. Constraint (8) ensures that any pair of blocks that have a horizontal relationship do not overlap with each other. Constraint (9) ensures that l_x is the minimum reciprocal of linear power density. The y-axis formulation is analogous.

II.B. Solving MLPDM Problem by Minimum Cycle Ratio Algorithm

In this subsection, we solve MLPDM problem by converting it to minimum cycle ratio (MCR) problem, which can be solved efficiently and optimally. The MCR problem is defined as follows. Given a digraph $G = (V, E)$, each edge e has a weight w_e and a non-negative transit time t_e. The cycle ratio λ of a cycle C is the ratio of its sum of weights to its sum of transit times:

$$\lambda(C) = \frac{\sum_{e \in C} w_e}{\sum_{e \in C} t_e}. \tag{10}$$

The smallest ratio λ^* among all cycles in the graph G must be found.

We rearrange Eq. (6)–(9) as follows:

$$0 - x_i \leq -\frac{w_i}{2} - 0 \cdot l_x, \tag{11}$$
$$x_i - W \leq -\frac{w_i}{2} - 0 \cdot l_x, \tag{12}$$
$$x_i - x_j \leq -(\frac{w_i}{2} + \frac{w_j}{2}) - 0 \cdot l_x, \text{ and} \tag{13}$$
$$x_i - x_j \leq 0 - r_{ji} l_x. \tag{14}$$

where $r_{ji} = (P_i/2 + P_j/2)$. Note that these constraints coincide with the form of the dual of the MCR problem [11], indicating that we can

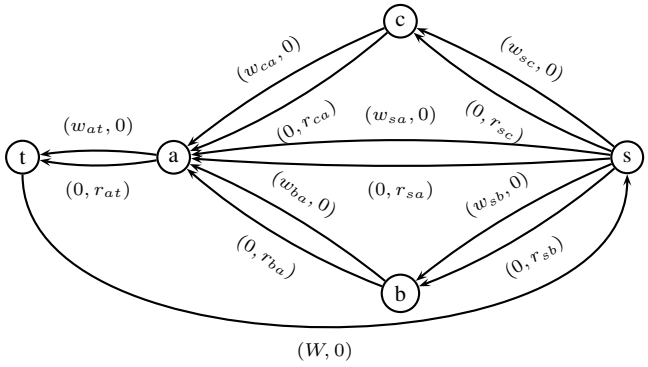


Fig. 3. MCR graph in x direction for the block relationships in Fig. 2.

construct the graph for the MCR problem G_H^{MCR} in the following way.

1) If B_j has an incident edge from B_i in the constraint graph G_H, we add two directed edges in G_H^{MCR} from B_j to B_i; one has weight $w_{ji} = -(\frac{w_i}{2} + \frac{w_j}{2})$ and transit time 0 and the other has zero weight and transit time r_{ji}.
2) We add two vertices s and t to the graph, which represent the right and left boundaries respectively. We add two edges from s to each block B_i if it does not have any incoming edges from any other blocks, and two edges from B_i to t if the block does not have any outgoing edges. The weights and transit times of the two edges are $(-\frac{w_i}{2}, 0)$ and $(0, r_i)$ with $r_i = P_i/2$.
3) We add one more edge from t to s with weight W and zero transit time to ensure that there exist cycles in the graph G_H^{MCR} and all blocks are placed within the layout boundary length W.

The MCR graph for y-axis constraints (G_V^{MCR}) can be constructed in a similar way. Figure 3 gives an example of the MCR graph in x direction for the blocks in Figure 2(a). The objective functions l_x and l_y become the minimum cycle ratio in the graph G_H^{MCR} and G_V^{MCR}.

II.C. Further Improvement by Iterative MCR Algorithm

Solving the MCR problem obtains the maximal l_x for G_H^{MCR} and a critical cycle. We define the cost of an edge c_e as follows:

$$c_e = w_e - l_x \cdot r_e. \tag{15}$$

A critical cycle is a cycle in the graph for which the sum of the costs of edges is zero, which means that l_x cannot be further increased for this cycle, i.e., no more white space can be further inserted without violating the constraints. In our problem, critical cycles always start at s, travel through some blocks, reach t, and return to s. For example, see $s \rightarrow b \rightarrow a \rightarrow t \rightarrow s$ in Figure 3. The location of each block in the critical cycle can be computed by adding the location of its predecessor in the critical cycle and the cost of the edge to its predecessor (the cost is negative except for e_{ts}).

The locations of the blocks outside the critical cycle are determined by using the shortest path algorithm to find the cost of the shortest (most negative) path from s to the block; the block location is given by W plus the most negative cost. However, the locations of the remaining blocks are still flexible. For instance, in Figure 2(b), block c is allowed to be moved within R_c as long as its local power density for neither block a or the right boundary exceed $1/l_x$.

To determine the locations of the remaining blocks, we iteratively fix the blocks in the current critical cycle and then solve MCR problem for the remaining flexible blocks until all blocks are fixed. When the location of block B_i is fixed at x_i, two operations are required to update the graph:

Algorithm 1 Iterative MCR Algorithm

1: Given an MCR graph G_H^{MCR} or G_V^{MCR}.
2: **while** there exist flexible blocks **do**
3: Solve MCR problem in current graph.
4: Fix the blocks in the critical cycle.
5: Update graph: do OP_1 and OP_2 on every recently fixed block.
6: **end while**
7: Output locations of each block.

1) OP_1, which removes all edges between B_i and all other fixed blocks, and
2) OP_2, which adds an edge with weight $-x_i$ and zero transit time from B_i to t and an edge with weight $x_i + err$ and zero transit time from t to B_i, where err is a small positive number we add to the weight to prevent the ratio of cycle $B_i \rightarrow t \rightarrow B_i$ from being 0/0.

The iterative MCR algorithm is summarized in Figure 1.

The iterative MCR algorithm only considers blocks on the same layer and can be applied to traditional 2D floorplans. For 3D floorplans, the iterative MCR algorithm can be applied to each layer separately, but this is suboptimal. It appears challenging to extend the iterative MCR algorithm to solve multiple layers simultaneously because the algorithm requires that each pair of blocks have fixed x-axis or y-axis relationships. For multiple layers, blocks on different layers do not necessarily have such fixed relationships.

There exist many algorithms for solving the minimal cycle ratio problem [12]. We choose Howard's algorithm [12] due to its efficiency. The run time of Howard's algorithm is $O(nmN_2)$ where n is the number of vertices, m is the number of arcs, and N_2 is the number of simple cycles in the graph. The number of iterations is at most n. Therefore, the total run time of iterative MCR algorithm is $O(n^2 m N_2)$.

III. Inter-Layer Heat Flow Optimization with Dynamic Programming

The iterative MCR algorithm treats each layer separately, assigning more white space between blocks with higher power densities. However, in 3D ICs, blocks on different layers may overlap with each other. Inter-layer heat flow heavily influences thermal profile, introducing a problem not addressed by the Iterative MCR algorithm. We now describe a method of extending our iterative MCR algorithm to consider inter-layer heat flow optimization.

III.A. Problem Formulation of Inter-Layer Heat Flow Optimization

In this phase, we discretize the IC layout into $M \times N$ tiles in M rows and N columns. We assume that each tile is small enough that very few will overlap more than one block within a layer. A block may span several tile grid rows and columns. We also assume that the left (lower) boundary of each block coincides with the left (lower) boundary of the leftmost (lowest) occupied column (row). When tile $Tile_{i,j}^l$ overlaps with block B_k^l, we define tile power consumption in $Tile_{i,j}^l$ to be

$$PT_{i,j}^l = pd_k^l \times Overlap_{i,j}^l(k), \tag{16}$$

where $Overlap_{i,j}^l(k)$ is the area that $Tile_{i,j}^l$ overlaps with block B_k^l and pd_k^l is the power density of that block.

We use a compact resistive thermal model to model inter-layer heat flow [13], [14]. Each tile is an isothermal node in the thermal model. Adjacent nodes have thermal conductances. Power sources in this model are analogous to current sources in electrical networks, thermal conductance is analogous to electrical conductance, and heat

flow is analogous to electrical current. In this phase, we consider only inter-layer heat flow, and consider only inter-layer thermal resistances R_L and R_H, where R_L is the thermal resistance of dielectric layers in z direction and R_H is the thermal resistance of the heat sink. Temperature $T_{i,j}$ on the top layer at tile $Tile^L_{i,j}$ can be expressed in the following form:

$$T_{i,j} = \sum_{l=1}^{L} [(l-1)R_L + R_H] PT^l_{i,j}. \quad (17)$$

Eq. (17) shows that power sources on all layers contribute to the temperature of the layer farthest from the heatsink. We define weighted power density ${pd'}^l_k$ for blocks as follows:

$${pd'}^l_k = [(l-1)R_L + R_H] pd^l_k. \quad (18)$$

Weighted overlapped power consumption ${OP'}^l_{i,j}(k)$ and weighted tile power consumption ${PT'}^l_{i,j}$ are calculated in a similar way, but using pd' instead of pd.

Our goal in this phase is to minimize the maximum chip temperature $T_{i,j}$ by adjusting the locations of blocks. Because we start with the solution given by the iterative MCR algorithm described in Section II-B, we must constrain the locations of blocks in order to preserve high-quality characteristics of the MCR solution. We therefore impose a motion range constraint, MR. Blocks may only be moved within the range $[-MR, MR]$ with respect to the original (MCR) block location in both x and y directions. Minimizing maximum chip temperature, according to Eq. (17), is equivalent to minimizing the maximum $\sum_{l=1}^{L} {PT'}^l_{i,j}$ over all tiles. However, it is difficult to consider all layers simultaneously. We therefore consider layers sequentially. On each layer, we also separate x and y directions. In the following subsections, we only illustrate the algorithm in x-axis. The y-axis solution is analogous. We formulate our problem as follows.

Inter-Layer Heat Flow Optimization on Individual Layer Problem: We define the Stacked Tile Power Consumption $STPC_{i,j}$ as the sum of ${PT'}^l_{i,j}$ of the layers for which a dynamic programming solution has already been obtained. $STPC_{i,j}$ is initialized to zero since at the beginning no layer has been processed. Given an original floorplan for layer l, the problem is to adjust x (or y) coordinates for all blocks on layer l, within a motion range $[-MR, MR]$, such that maximum of $STPC'_{i,j}$ is minimized, where $STPC'_{i,j} = STPC_{i,j} + {PT'}^l_{i,j}$.

The order in which layers are processed influences solution quality. Our strategy is to choose the layer with the block of the highest weighted power density among all the remaining blocks at each iteration. By giving priority to blocks with higher weighted power densities, white space insertion decisions for other layers can be made in context of these prior decisions, i.e., overlapping with high power-density blocks can be more easily avoided.

III.B. Inter-Layer Heat Flow Optimization

In this subsection, we describe a dynamic programming algorithm to solve the inter-layer heat flow optimization problem. We first provide some useful definitions.

- G_H is the x-axis constraint graph for the given floorplan. We add a source vertex s and a sink vertex t representing the left and right boundaries.
- $L(k)$ is the number of tiles that block k spans on the x-axis, i.e., $L(k) = \lceil w_k / w_{tile} \rceil$, where w_{tile} is the tile width.
- If there is a path in G_H from block k' to block k, then k' is a predecessor of k and k is a successor of k'. If the path only contains one edge, k' is a direct predecessor of k and k is a direct successor of k'.
- We define a dependence set $D(k)$ of block k, which includes all blocks that are predecessors of k in G_H.
- $prev(k)$ denotes the set of vertices that are direct predecessors of vertex k.
- We define block cost $cost(k, i)$ as the maximum of $STPC'$ among all the tiles overlapped by block k if the right boundary of block k is located at column i. If i is out of motion range MR, the cost is unacceptable (infinite).
- $f(k, i)$ is the minimal maximum of $STPC'$ among all the tiles overlapped by blocks in $D(k)$ when the right boundary of block k is placed before or at column i.

Definition 1: A set of blocks S is said to be constrained by a value f if the cost of any block in S is less than or equal to f.

Definition 2: A floorplan for a set of blocks S is said to be a "feasible floorplan" constrained by a value f if the following two constraints are satisfied:

1) blocks in S do not overlap each other and
2) S is constrained by f.

Next, using the above notations and definition, we introduce two lemmas that will help us prove the correctness of the recursion equation for a dynamic programming solution of the inter-layer heat flow optimization problem.

Lemma 1: Given two dependence sets $D(k_1)$ and $D(k_2)$, and their corresponding feasible floorplans constrained by $f(k_1, i_1)$ and $f(k_2, i_2)$, respectively. There exists a feasible floorplan for set $D(k_1) \cup D(k_2)$ constrained by the value $\max\{f(k_1, i_1), f(k_2, i_2)\}$. We call such a feasible floorplan a *combined feasible floorplan*.

Proof: We combine two feasible floorplans by following the following rules.

1) If block k' appears only once in either $D(k_1)$ or $D(k_2)$, in combined floorplan it is still placed at the same location as in the original floorplan of either $D(k_1)$ or $D(k_2)$, where block k' appears.
2) If block k' appears in both dependence sets, it has two candidate locations. In the combined floorplan, it is placed at the left-most candidate location.

We first prove there is no overlap in the combined floorplan. If block k' is placed using Rule 1, all of block k''s successors must appear only once and in the same original floorplan as block k' does. Otherwise, by the definition of dependence set, block k' must be a predecessor of both k_1 and k_2. Thus, it must be in both dependence sets, which results in a contradiction. k' and all of its successors are in the same feasible floorplan. In the combined floorplan, they are placed at the same locations they had in the original floorplan.

If block k' is placed by Rule 2, obviously the left candidate location does not overlap with direct successors in either of the original floorplans. Therefore, there is no overlap with direct successors in the combined floorplan. It is easy to extend this argument to all successors by induction. Therefore, we have also proven that Rule 2 does not induce overlaps.

Costs of blocks in the combined floorplan do not exceed $\max\{f(k_1, i_1), f(k_2, i_2)\}$ because we do not create new locations for blocks. Any location in combined floorplan is covered by at least one of the two original floorplans. Thus no cost exceeds $\max\{f(k_1, i_1), f(k_2, i_2)\}$. ■

We have shown that by following Rules 1 and 2, we can construct a combined feasible floorplan from two feasible floorplans. If we combine multiple floorplans iteratively, this immediately leads to the following Lemma.

Lemma 2: Given several dependence sets $D(k_1)$, $D(k_2)$, ..., $D(k_m)$ and their corresponding feasible floorplans constrained by $f(k_1, i_1)$, $f(k_2, i_2)$, ..., $f(k_m, i_m)$, there exists a combined feasible floorplan for set $D(k_1) \cup D(k_2) \cup \ldots \cup D(k_m)$ constrained by the value $\max\{f(k_1, i_1), f(k_2, i_2), \ldots, f(k_m, i_m)\}$.

We are now ready to derive the recursion equation for our dynamic programming based inter-layer heat flow optimization technique.

Theorem 1: $f(k, i)$ can be computed by the following recursion equation:

$$f(k,i) = \min\{f(k,i-1), \max_{k' \in prev(k)}\{f(k', i - L(k)), cost(k,i)\}\}, \quad (19)$$

with initial conditions

$$f(s,i) = 0 : 1 \leq i \leq N \text{ and} \quad (20)$$

$$f(k,i) = \infty : i \leq 0, \quad (21)$$

where N is the total number of columns.

Proof: Eq. (21) implies that all blocks must be placed in the IC bounding box. Eq. (20) is the initial condition that sets the source node, since the source node is not a real block. $f(s, i)$ is zero. Now we will prove Eq. (19) by induction.

When calculating $f(k, i)$, there are only two cases to consider: the right boundary of block k is placed at column i or to the immediate left of column i. Each alternative results in a $STPC'$ less than $D(k)$. Obviously, $f(k, i)$ should be the smaller value for the two alternatives. If being placed left of column i results in a smaller value, we have $f(k, i-1) = f(k, i)$. Note that a feasible floorplan constrained by $f(k, i-1)$ must also be a feasible floorplan constrained by $f(k, i)$. If placing the block at column i results in a smaller value, the right boundaries of all blocks in $prev(k)$ must be placed to the left of or at column $i - L(k)$. By Lemma 2, we can combine all feasible floorplans of $D(k')$, where $k' \in prev(k)$. The combined floorplan is constrained by the value $\max_{k' \in prev(k)}\{f(k', i - L(k))\}$. We then add block k, resulting in a maximum $STPC'$ of $\max_{k' \in prev(k)}\{f(k', i - L(k)), cost(k, i)\}\}$. Therefore, both of these alternatives have a corresponding feasible floorplan, $D(k)$. By the definition of $f(k, i)$, $f(k, i)$ should be the smallest value yielded by these two cases. ■

After $f(k, i)$'s have been computed, the following theorem gives the value of minimal maximum $STPC'$.

Theorem 2: The minimal maximum $STPC'$ is given by

$$\max\{f(t, L), \max\{STPC\}\} \quad (22)$$

Proof: Tiles on current layer can be divided into two classes: those that overlapped with another block on the current layer and those that do not overlap another block. We know that $\max\{STPC'\} \geq \max\{STPC\}$ and $D(t)$ includes all blocks on this layer. Thus, $f(t, L)$ gives the minimum maximum of $STPC'$ among all overlapped tiles. If $f(t, L) \leq \max\{STPC\}$, the minimal maximum of $STPC'$ must be at a non-overlapped tile, i.e., $\max\{STPC\}$, otherwise, it is determined by $f(t, L)$. ■

The locations of blocks can be determined by combining the feasible floorplans described in Lemmas 1 and 2. Note that this is not the only feasible solution because there still remains flexibility in block positions as long as their costs do not exceed the minimal maximum $STPC'$. Our approach processes layers iteratively without information about later layers so it cannot guarantee optimality.

Given n blocks and N discretized rows (columns), $f(k, i)$ must be computed $n \cdot N$ times and there are at most n direct predecessors at each recursion step. Therefore, the run time of the dynamic programming algorithm is $O(n^2 N)$ for each layer.

Algorithm 2 Complete Two-Phase Algorithm Flow for 3D ICs

```
Run Iter-MCR algorithm for x- and y-axes on each layer.
Initialize STPC to zero.
UL ← all layers.
while UL ≠ ∅. do
    CL ← the layer in UL containing the block with the highest
    weighted power density among all blocks on unlabeled layers.
    if CL is not the first chosen layer. then
        Run DP algorithm on CL.
    end if
    UL ← UL/{CL}.
    Update STPC.
end while
```

IV. Complete Two-Phase Algorithm Flow for 3D ICs

This section summarizes the two-phase MCR and dynamic programming 3D white space insertion algorithm. In the first phase, we run Iter-MCR on each layer. Next, choose an unlabeled layer to apply the dynamic programming algorithm to. The chosen layer contains the block with the highest weighted power density among all blocks on the unlabeled layers. After running the dynamic programming algorithm, the current layer is labeled. This process repeats for all layers. $STPC$ is updated at the end of each iteration. The algorithm is described in Algorithm 2, where UL is the set of unlabeled layers and CL represents the current layer.

V. Experimental Results

We implemented our algorithm in C++ using GCC 4.3.2. Howard's algorithm [12] was used to solve the MCR problem in Phase 1. Since Phase 1 only considers lateral heat flow, we test it on one-layer floorplans. The floorplans are generated by temperature-aware 3D floorplanner 3D-STAF [5] based on one-layer MCNC and GSRC benchmarks for which each block has been assigned a random power consumption in the range $[10^6$–$10^7]$W/m^2 [5]. Since 3D-STAF does not take white space into consideration, the floorplan initially has little white space. We need to enlarge the chip area with more white space before starting white space redistribution. The ISAC-II [15] thermal analysis software is used to compute thermal profiles and peak temperatures. In all the following experiments, the chip area is enlarged to 150% that given by the white-space-unaware floorplanner and the environment temperature is set to [318.15]K. The original T_{max} is the peak temperature calculated for the original floorplan centered in the enlarged chip area.

For comparison, we implemented an Even White-space Insertion (EWI) algorithm that inserts white space evenly between every pair of adjacent blocks on x and y axes. The widths and heights of white space are determined by a binary search that finds the maximal width/height under the constraint that no block be placed beyond chip area. We also implemented the Mathematical Programming (MP) based algorithm described by Wong and Lim [8]. In our implementation, the CPLEX solver is used.

Table I shows the experimental results for Phase 1. "T_{max} Red." is the reduction in peak temperature compared to the original T_{max}. The results show that on average EWI only improves peak temperature 45% as much as MCR. MP improves peak temperature 65% as much as MCR. The run time of iterative MCR is much lower than that of MP, because the MP algorithm iteratively solves a non-linear mathematical program with a large number of constraints. There are on the order of $O(NM)$ constraints, where N is the number of blocks and M is the number of tiles (10 by 10 in our implementation).

To test Phase 2, we follow the algorithm flow described in Algorithm 2. The flow is run on 3D floorplans with four layers generated

TABLE I
EXPERIMENTAL RESULTS FOR ITERATIVE-MCR ALGORITHM ON SINGLE-LAYER FLOORPLANS

Circuit	Original T_{max}	EWI Algorithm		MP Algorithm			Iter-MCR Algorithm		
		T_{max} (K)	T_{max} Red.	T_{max} (K)	T_{max} Red.	Run time(s)	T_{max} (K)	T_{max} Red.	Run time(s)
ami33	367.97	365.21	2.76	362.80	5.17	591	361.48	6.49	0.15
ami49	420.10	407.34	12.76	414.45	5.65	1147	406.76	13.34	0.41
n100	362.15	360.96	1.18	357.45	4.70	2247	354.07	8.08	4.31
n200	357.68	354.46	3.22	349.87	7.81	8070	345.36	12.32	64.9
n300	356.63	350.96	5.67	346.46	10.17	21429	344.47	12.16	288
Average ratio			0.45		0.65	1491.53		1.00	1.00

TABLE II
EXPERIMENTAL RESULTS OF DYNAMIC PROGRAMMING ALGORITHM FOR FOUR-LAYER FLOORPLANS

Circuit	Original T_{max}	Iter-MCR Only		Iter-MCR+DP	
		T_{max} (K)	T_{max} Red.	T_{max} (K)	T_{max} Red.
ami33	395.50	393.42	2.08	391.09	4.41
ami49	450.65	449.64	1.01	438.48	12.16
n100	416.07	401.12	14.94	379.78	36.29
n200	450.68	430.09	20.59	402.97	47.72
n300	452.96	431.28	21.68	406.76	46.2
Average ratio			1.00		4.21

by 3D-STAF. In Phase 2, the chip is divided into 500 rows and 500 columns. The motion range is set to 1/10 the layout width (or height), i.e., 50 tiles. R_L and R_H are calculated by dividing the material height by its thermal conductivity. Thermal conductivities of heat sink and dielectric layer are $[237]\text{Wm}^{-1}\text{K}^{-1}$ and $[142.3]\text{Wm}^{-1}\text{K}^{-1}$. The heat sink is [800]µm high and the dielectric layer is [200]µm high.

Table II shows the experimental results for Phase 2. The "Iter-MCR Only" column shows the results of only applying Iter-MCR to each layer and the "Iter-MCR+DP" column is the results of applying DP after Iter-MCR. The table shows that Iter-MCR+DP achieves 4.21 times reduction in peak temperature on average compared to Iter-MCR alone.

VI. CONCLUSION

In this paper, we described a two-phase white space redistribution algorithm for 2D and 3D IC temperature optimization. We first used an MCR problem formulation for the lateral heat flow temperature minimization problem. This formulation supports an efficient and optimal solution. This first solution can be applied to traditional 2D floorplans or to each layer separately in 3D floorplans. We then considered the problems introduced by inter-layer heat flow in 3D ICs. We discretized floorplans into arrays of tiles. A dynamic programming algorithm was developed to minimize the maximum stacked tile power consumption. We tested our iterative MCR algorithm on 2D floorplans and tested our two-phase MCR and dynamic programming algorithm on multi-layer 3D floorplans. Compared with even white space insertion and a mathematical programming based white space insertion algorithm, our MCR algorithm achieved the lowest peak temperatures. The MCR and dynamic programming algorithm for 3D ICs reduced peak temperature by more than four times compared to an approach ignoring inter-layer heat flow.

REFERENCES

[1] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration," *Proc. of the IEEE*, vol. 89, no. 5, pp. 602–633, May 2001.

[2] S. Im and K. Banerjee, "Full chip thermal analysis of planar (2-D) and vertically integrated (3-D) high performance ICs," in *Technical Digest Int. Electron Devices Meeting*, Dec. 2000, pp. 727–730.

[3] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *Proc. Int. Conf. on Computer-Aided Design*, Nov. 2004, pp. 306–313.

[4] V. Nookala, D. Lilja, and S. S. Sapatnekar, "Temperature-aware floorplanning of microarchitecture blocks with IPC-power dependence modeling and transient analysis," in *Proc. Int. Symp. on Low Power Electronics and Design*, Mar. 2006, pp. 298–303.

[5] P. Zhou, Y. Ma, Z. Li, R. P. Dick, L. Shang, H. Zhou, X. Hong, and Q. Zhou, "3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits," in *Proc. Int. Conf. on Computer-Aided Design*, Aug. 2007, pp. 590–597.

[6] X. Tang, R. Tian, and M. Wong, "Optimal redistribution of white space for wire length minimization," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 2005, pp. 412–417.

[7] X. Li, Y. Ma, X. Hong, S. Dong, and J. Cong, "LP based white space redistribution for thermal via planning and performance optimization in 3D ICs," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 2008, pp. 209–212.

[8] E. Wong and S. Lim, "Whitespace redistribution for thermal via insertion in 3D stacked ICs," in *Proc. Int. Conf. Computer Design*, Oct. 2007, pp. 267–272.

[9] J. Yan, Z. Chen, Y. Chou, S. Lin, and H. Chiueh, "Thermal-driven white space redistribution for block-level floorplans," in *Proc. Int. Conf. on Electronics, Circuits and Systems*, Aug. 2008, pp. 662–665.

[10] H. Murata, K. Fijiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, Oct. 1996.

[11] M. Hartmann and J. Orlin, "Finding minimum cost to time ratio cycles with small integral transit times," *Networks*, vol. 23, no. 6, pp. 567–574, Sep. 1993.

[12] A. Dasdan, "Experimental analysis of the fastest optimum cycle ratio and mean algorithms," *ACM Trans. on Design Automation of Electronic Systems*, vol. 9, no. 4, pp. 385–418, Oct. 2004.

[13] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S.Velusamy, "Compact thermal modeling for temperature-aware design," in *Proc. Design Automation Conf.*, Jun. 2004, pp. 878–883.

[14] J. Cong and Y. Zhang, "Thermal via planning for 3-D ICs," in *Proc. Int. Conf. on Computer-Aided Design*, Nov. 2005, pp. 745–752.

[15] Z. Hassan, N. Allec, L. Shang, R. Dick, V. Venkatraman, and R. Yang, "Multiscale thermal analysis for nanometer-scale integrated circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 6, pp. 860–873, June 2009.