Performance and Thermal-Aware Steiner Routing for 3-D Stacked ICs

Mohit Pathak, *Student Member, IEEE*, and Sung Kyu Lim, *Senior Member, IEEE*

***Abstract*—In this paper, we present a performance and thermal-aware Steiner routing algorithm for three-dimensional (3-D) stacked integrated circuits. Our algorithm consists of two steps: tree construction and tree refinement. Our tree construction algorithm builds a delay-oriented Steiner tree under a given thermal profile. We show that our 3-D tree construction involves minimization of two-variable Elmore delay function. In our tree refinement algorithm, we reposition the through-silicon-vias (TSVs) used in existing Steiner trees while preserving the original routing topology for further thermal optimization under a performance constraint. We employ a novel scheme to relax the initial nonlinear programming formulation to integer linear programming and consider all TSVs from all nets simultaneously. Our tree construction algorithm outperforms the popular 3-D maze routing by 52% in terms of performance at the cost of 15% wirelength and 6% TSV count increase for four-die stacking. In addition, our TSV relocation results in 9% maximum-temperature reduction at no additional area cost. We also provide extensive experimental results, including the following: 1) the wirelength and delay distribution of various types of 3-D interconnects; 2) the impact of TSV *RC* parasitics on routing and TSV relocation; and 3) the impact of various bonding styles on routing and TSV relocation. Last, we provide results on two-die stacking.**

***Index Terms*—Steiner routing, thermal optimization, three-dimensional (3-D) integrated circuit (IC), through-silicon-via (TSV).**

I. Introduction

TECHNOLOGY feature sizes continue to shrink to meet performance demands on integrated circuits (ICs). This, coupled with growing overall chip dimensions, leads to greater consumption of the available delay and power budgets by the interconnect structures on these chips. As global and semiglobal wires become increasingly expensive and clock frequencies become higher and higher, designers seek new architectures and technologies that rely less on sending signals across the chip. However, few scalable solutions have been proposed. One such solution is three-dimensional (3-D) integration.

In a 3-D IC, transistors may be fabricated on top of other transistors, resulting in multiple layers of active components. These transistors may then be wired to other transistors on the same device layer, to transistors on different device layers, or both, depending on the process technology. In the wafer-bonding approach [1], discrete wafers are "glued" together using vertical copper interconnects, permitting multiple wafers and multiple 3-D interconnects and thus overcoming the aforementioned limitations. Three-dimensional integration offers tremendous potential for keeping Moore's law on track. It provides a means to continue to increase device density by stacking more transistors in the same footprint. Three-dimensional integration also addresses the wire-delay problem by enabling the replacement of long and slow global interconnects with short and fast vertical routes.

Conventional 3-D integration, so-called system-in-package (SIP), involves stacking packaged chips with wire-bonding-based communication. Our target technology is to stack bare dies, not packaged chips, and utilize through-silicon-vias (TSVs) to establish interconnect among the dies. With this approach, the savings on the total amounts of wiring, delay, and its power consumption easily outnumber that of stacked packages. In addition, the absence of off-chip communication naturally translates into smaller delay and low power. Compared with system-on-chip implementation, TSV-based 3-D die stacking helps reduce noise and interference among mixed-signal components since these are separated into different dies. The advancement of TSV technology has matured enough to shrink the via size to a few micrometer dimensions, thereby contributing little to the area, delay, and power consumption of the overall system.

One of the major concerns of 3-D ICs is thermal dissipation. Stacking of different device layers combined with the low thermal conductivity of the bonding material may result in excessively high on-chip temperature. The location of TSVs in a Steiner tree has a high impact on the overall topology, as well as the delay at the sink nodes of the tree, since it determines the amount of wiring done at all intermediate dies that the tree spans. Moreover, TSVs play a significant role in lowering the temperature of the chip. The reason is that they establish thermal paths to the heat sink when placed in the middle of a hot spot. Thus, several existing works utilize TSVs to lower the on-chip temperature of 3-D ICs [2]–[7].

In this paper, we formulate and solve the new *Performance and Thermal-Aware 3-D IC Steiner Routing* problem for multipin net routing in 3-D stacked ICs. We emphasize that this problem is different from the conventional 2-D *Steiner Routing With Multiple Routing Layers*. The main reason is that the pins in 3-D ICs are located in *multiple* device layers, whereas the pins in 2-D ICs are located in a single device layer,

Manuscript received September 11, 2008; revised January 3, 2009 and March 16, 2009. Current version published August 19, 2009. This work was supported in part by grants from SRC C2S2 and in part by the NSF CAREER Award under Grant CCF-0546382. A short version of this paper was presented in part at the 2007 International Conference on Computer-Aided Design, San Jose, CA, November 5–8. This paper was recommended by Associate Editor J. Hu.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: limsk@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2009.2024707

0278-0070/$26.00 © 2009 IEEE

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

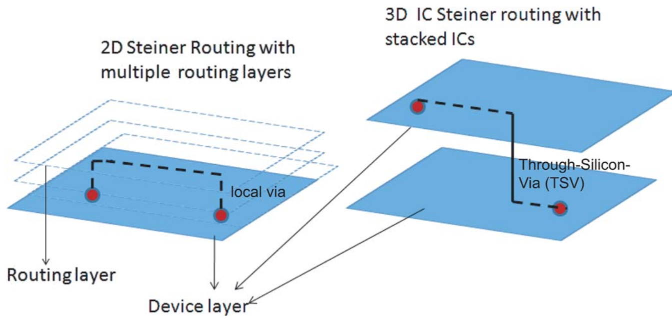

Fig. 1. Difference between 2- and 3-D IC routing. In the 3-D IC, pins are located in multiple device layers, whereas in the 2-D IC, all pins lie in the same device layer.

as shown in Fig. 1. The contributions of this paper are as follows.

1) Unlike the existing works on 3-D Steiner tree construction [2], [4], [8], [9] that focus on wirelength and thermal optimization, our tree construction algorithm optimizes thermal-aware performance. We believe that performance still remains as an important design metric, and the thermal-aware delay model is important in Steiner tree construction. Unlike the existing works that decompose a 3-D interconnect into a set of 2-D interconnects, our router considers all pins in all dies and their bonding styles simultaneously and constructs performance-oriented Steiner trees while determining the optimal location for TSVs.
2) We formulate and solve the new *TSV Relocation* problem for thermal optimization in 3-D stacked ICs. Unlike the existing works on TSV-related thermal optimization that insert *additional* dummy TSVs for thermal optimization, our thermal optimization is based on relocating *existing* TSVs while maintaining the original routing topology. Thus, our thermal optimization requires neither valuable routing resource nor rip-up-and-reroute. We employ a novel scheme to relax the initial nonlinear programming (NLP) formulation to integer linear programming (ILP) and consider all TSVs from all nets simultaneously.
3) Our tree construction algorithm outperforms the popular 3-D maze routing by 52% in terms of performance at the cost of 15% wirelength and 6% TSV count increase for four-die stacking. In addition, our TSV relocation results in 9% maximum-temperature reduction at no additional area cost. We also provide extensive experimental results, including the following: 1) the wirelength and delay distribution of various types of 3-D interconnects; 2) the impact of TSV RC parasitics on routing and TSV relocation; and 3) the impact of various bonding styles on routing and TSV relocation. Last, we provide results on two-die stacking.

The remainder of this paper is organized as follows. Section II presents related works. Section III provides the problem formulation. Section IV presents our 3-D Steiner tree construction algorithm. Section V presents our thermal-aware TSV relocation algorithm. Experimental results are presented in Section VI, and we conclude this paper in Section VII.

II. Related Works

The history of routing algorithm development for 3-D ICs is relatively short. Das *et al.* [8] presented a set of standard cell-based physical design tools for 3-D ICs. Their 3-D global routing algorithm is based on a 3-D extension of [10], where the routing region is recursively partitioned into a series of x-, y-, and z-direction cuts in a top–down fashion. Routing topologies are then gradually optimized and refined as more and more subregions are generated by the partitioning. The z-direction cuts introduce vertical connection $(=$ TSVs$)$, and the authors utilize the routing channels in between the cell rows to insert TSVs. The objective is to reduce wirelength, and thermal or performance effects are not considered.

Cong and Zhang [2] presented a global routing algorithm for 3-D ICs that is based on maze routing. Their goal is to minimize wirelength and TSV counts under a given thermal constraint. Given a set of points in a 3-D grid, they first build a 3-D minimum spanning tree (MST). Then, for each edge $e(s,t)$ in the MST, shortest-path-based maze search from s to t is performed.[1] Once the routing is completed, dummy TSVs are inserted in the layout whitespace for thermal optimization. Then, the whole process of routing and TSV insertion is performed in a multilevel routing framework. They later presented a follow-up work [3], where TSV insertion and refinement are performed based on the NLP formulation.

Zhang *et al.* [4] performed thermal-aware global routing for 3-D ICs while utilizing "thermal vias" and "thermal wires." The authors first decompose each multipin net in the given netlist into a set of two-pin nets based on MST construction. Next, a routing congestion map is obtained based on the L/Z shape topology assumption. The locations of TSVs are then determined for all interdie two-pin nets based on the congestion map. The authors complete the routing for all nets by performing 2-D maze routing in each die separately. Based on this initial tree construction, thermal analysis is performed to identify hotpots. The authors then perform thermal-via/wire insertion and rip-up-and-reroute alternatively to remove temperature and congestion violations iteratively.

Note that Cong and Zhang [2] and Zhang *et al.* [4] insert *additional* dummy TSVs for thermal optimization. On the other hand, our thermal optimization is based on relocating *existing* TSVs while maintaining the original routing topology. Thus, our thermal optimization requires neither valuable routing resource nor rip-up-and-reroute. In addition, our routing trees are built under performance and thermal constraints.[2]

Pavlidis and Friedman [11] presented an analytical delay model for interdie 3-D interconnects. Their delay model is a function of TSV location/height and the related wires. Using this model, they determine the delay-optimal TSV location along a 3-D interconnect that connects gates in different dies.

[1] We provide comparison between our 3-D router to this so-called 3-D maze router in Section VI.

[2] In addition to the thermal-via insertion during routing, Goplen and Sapatnekar [5] insert thermal vias after placement, while Li *et al.* [6] try to redistribute whitespace in a given floorplan to allocate space for thermal vias. Yu *et al.* [7] insert both dummy thermal TSVs and power/ground TSVs simultaneously to reduce thermal and power-supply noise.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

However, this model is based on two-pin connections, and they do not perform routing. Minz and Lim [9] perform block-level global routing for 3-D SIP, where the fundamental problem is similar: Utilize routing layers in between multiple device layers as well as the routing channels around the modules in each device layer to complete routing. The goal is to minimize wirelength, layer, congestion, and crosstalk. They formulate and design heuristics for 3-D pin redistribution, net distribution, and channel assignment problems.

III. Preliminaries

A. Problem Formulation

We assume that the following are given: 1) a set of m nets $\{n_0, n_1, \cdots, n_{m-1}\}$, where each net is represented by a list of pins $n_i = \{p_0, p_1, \cdots, p_{k-1}\}$, with p_0 being the driver; 2) a 3-D routing grid G that represents the routing resource in a given 3-D stacked IC, where each grid node represents a routing region and each edge denotes the adjacency among the regions; 3) each x/y grid edge is associated with horizontal/vertical wire capacity and z with TSV capacity; 4) the location of each pin $p(x, y, z)$ in G; and 5) a 3-D thermal grid Z with thermal resistance on all edges and power consumption on all nodes.[3] A 3-D *Steiner Tree* is defined to be a set of 2-D ($=$ planar) Steiner trees connected by TSVs.

The goal of the *Performance and Thermal-Aware 3-D Steiner Routing* problem is to generate a 3-D Steiner tree for each net while satisfying the capacity constraints specified in the underlying G.[4] The objective is to minimize the following: 1) the maximum temperature among all nodes in the thermal grid and 2) the maximum Elmore delay among all pins in each tree, where the delay is computed based on the current thermal distribution. Note that the thermal resistance values for some edges in Z changes based on the number of TSVs assigned, which changes during routing and TSV relocation.

This paper uses the temperature-dependent interconnect delay model presented in [12]. The line resistance per unit length can be calculated as $r(x) = r_0(1 + \beta \cdot T(x))$, where r_0 is the resistance at 0 °C, β is the temperature coefficient of resistance, and $T(x)$ denotes the temperature at location x.

Depending on the number and the location of pins in each net, there exist the following four types of nets to be routed.

1) *Single-die–two-pin (SD2P) nets*: A net in this group connects two pins that are located in the same die.
2) *Single-die–multipin (SDMP) nets*: A net in this group connects more than two pins that are located in the same die.
3) *Multidie–two-pin (MD2P) nets*: A net in this group connects two pins that are located in two different dies.
4) *Multidie–multipin (MDMP) nets*: A net in this group connects more than two pins that are located in multiple different dies.

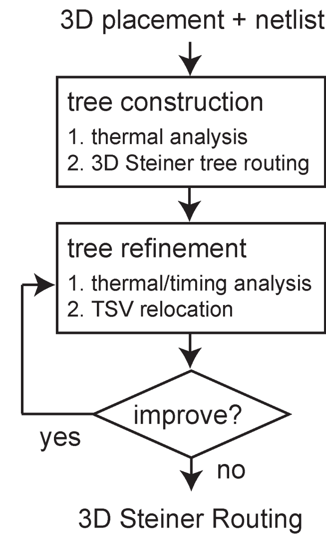

Fig. 2. Overview of our performance and thermal-aware 3-D Steiner routing.

Note that Steiner routers for conventional 2-D ICs deal with the first two types, whereas 3-D Steiner routers need to route all four types. The last two types require TSVs for interdie connections.

B. Overview of the Approach

Optimizing performance and thermal objectives simultaneously during 3-D Steiner routing is a challenging task. We solve this problem in two phases, namely, tree construction and tree refinement. The main goal during tree construction is to obtain initial trees that optimize performance under the given thermal profile. The main goal of tree refinement is to relocate the TSVs used in the initial trees for thermal optimization under the given timing constraint.

Fig. 2 shows an overview of our performance and thermal-aware 3-D Steiner routing process. Given a netlist and its 3-D placement, we first perform thermal analysis to obtain the thermal distribution to be used during tree construction. Next, we construct a performance-oriented routing tree for each net one by one under the nonuniform thermal profile. The temperature values are updated periodically to reflect the thermal resistance and temperature changes from TSV usage.[5] In our tree refinement phase, we first perform timing analysis and obtain timing slack for each pin. We also perform thermal analysis to identify thermal hot spots. We then minimize the 3-D on-chip temperature by relocating the TSVs used in each tree under the given timing constraints. The goal is to move TSVs closer to the hot spots so that the thermal resistance values are reduced in those regions. Note that our TSV relocation preserves the original tree topology while optimizing the thermal objective. We recompute the timing slacks and

[3] Our thermal-grid-Z dimension is an integer multiple of our routing-grid-G dimension. This ensures that all nets in a routing grid can be assigned to a single thermal grid.

[4] Note that we do not explicitly minimize routing/via congestion but implicitly address it by satisfying the capacity constraints. Each edge capacity can be set independently to reflect the availability of the routing resource. For example, we reduce the capacity of vertical routing-grid edges that go through a high-placement-density region.

[5] The thermal resistance change from a single TSV insertion is very small. In addition, updating thermal map after every net is computationally prohibitive for a large design. Thus, we choose to update thermal distribution periodically. We use the full 3-D thermal-grid model for our temperature analysis [13].

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

```
3D Steiner Tree Construction
input: netlist NL, routing graph G, thermal profile Z
output: 3D Steiner tree for each net
1.  for (each net n ∈ NL)
2.      T_n = p_0(n);
3.      Q_n = set of pins of n except p_0;
4.      while (Q_n ≠ ∅)
5.          for (each pin a ∈ Q_n)
6.              for (each edge e ∈ T_n)
7.                  x = connection point for a → e;
8.                  y = TSV location on e(x, a);
9.                  update dly(p) for all p ∈ T_n ∪ a;
10.                 X(a, e) = max{dly(p)};
11.         (a_min, e_min) = pin+edge pair with min X;
12.         T_n = T_n ∪ e_min;
13.         remove a_min from Q_n;
14.     update Z periodically;
15. for (each non-timing critical T_n violating capacity)
16.     rip-up-and-reroute T_n under Z;
```

Fig. 3. Pseudocode of the performance and thermal-aware 3-D Steiner tree construction algorithm. In case e and a are located in different planes, $e(r, a)$ will utilize TSVs.

temperature values to reflect the relocation at every iteration. Last, we repeat the whole tree refinement phase until no more thermal improvement is possible.

IV. 3-D Steiner Tree Construction

A. Overview of the Algorithm

The basic approach of our 3-D Steiner tree construction algorithm is similar to SERT [14], where an existing tree is incrementally grown by connecting a new sink pin to it. SERT starts with the driver pin and selects the next sink pin that minimizes Elmore delay when connected to the driver. This process continues until all sink pins are connected to the tree that is growing. The goal is to minimize the maximum Elmore delay among all sink pins of the tree. Here, the biggest challenge is to compute the point on the tree where the new pin connects to. There are three major differences between SERT and our work. First, all the pins in SERT are located in the same die, whereas our 3-D algorithm handles the pins located in multiple dies. This 3-D case requires the usage of TSVs, and the location of these TSVs has a huge impact on the topology of the tree as well as the sink pin delay. Second, the delay optimization in SERT is based on a single variable, whereas our algorithm deals with two-variable function optimization. Third, our interconnect delay is computed based on the given thermal profile.

A pseudocode of our algorithm is shown in Fig. 3. Our routing algorithm consists of two phases: construction (lines 1–14) and rip-up-and-reroute (lines 15 and 16). We construct 3-D Steiner trees during the construction phase while ignoring congestion and then fix the capacity violation by rip-up-and-reroute. Given a net n, our 3-D Steiner tree T_n initially contains the driver pin (line 2). We store the remaining pins of n in a set Q_n (line 3). We then examine all pin–edge pairs (lines 5 and 6) and compute the impact of connecting the pin to the edge on Elmore delay under the given thermal profile Z, where the pin is chosen from Q_n and the edge is from T_n. Specifically, the delay impact is calculated based on the increase in temperature-dependent Elmore delay among all pins currently in T_n (lines 9 and 10), where $dly(p)$ is the Elmore delay at pin p. This requires the computation of connection point x and TSV location y (lines 7 and 8) (to be discussed in Section IV-B). Next, we select the pin–edge pair that results in the minimum max-delay increase (line 11) and add the pin to T_n (lines 12 and 13). Since TSV insertion affects the thermal resistance of the related area, we perform thermal analysis periodically (not after every net routing) (line 14).

Our rip-up-and-reroute is done on nontiming critical nets, i.e., the nets with smaller max-delay values (line 15 and 16). Specifically, we first sort the nets that utilize routing edges that violated the capacity constraint based on their timing slack values. We then rip up the nets one by one in the sorted order and reroute it until the violation is completely removed. We use a maze router that minimizes weighted path length to reroute a net, where the weight considers the remaining routing capacity and temperature. In this case, our cost function penalizes routing edges that are more congested and/or located in a nearby hot spot. For a two-pin net, our maze router tries to find the source-to-sink shortest weighted path such that the routing capacity is not violated. For a multipin net, we first decompose the net into a set of two-pin nets based on their MST. We then route each two-pin subnet using our maze router described earlier.

B. Computing Connection Point and TSV Location

For a given multipin net and a partial tree, our goal is to find the next pin (and its connection point and TSV location) so that adding this pin, compared with other pins, minimizes the delay increase in the overall tree. A final Steiner tree is obtained once all the pins are added. This section discusses how to compute the connection point and TSV location. Our discussion is based on the two-die case for the simplicity of the discussion, but our algorithm is applicable to multiple-die stacking without any modification. Let r_1 and c_1 denote the unit length resistance and capacitance values for die 1. r_2 and c_2 are similarly defined for die 2. The capacitance and resistance of a TSV connecting the two dies are denoted as C_{via} and R_{via}, respectively.

Given a pin p and an edge $e \in T$, the *connection point* is defined as the point on e to which p is connected. The connection-point computation for the 2-D case has been presented in [14], where the Elmore delay change on an entire tree caused by adding a new pin to the tree is a function of a single variable x, the location of connection point. We extend this work by introducing a second variable y that represents the location of TSV. We then optimize the two-variable delay function and determine the location of connection point $(= x)$ and TSV $(= y)$ for the 3-D case.

Referring to Fig. 4, $e(p, c)$ and $e(q, b)$ are the edges on T. p is the parent node of $e(p, c)$, and q is the parent node of $e(q, b)$. a is the new pin that needs to connect to $e(p, c)$. Edge $e(p, c)$ lies on die 1 with interconnect parasitics r_1 and c_1, whereas a lies on die 2 with interconnect parasitics r_2 and c_2. d is the point on $e(p, c)$ that is of the shortest distance to a. x is the connection point, and y is the location of TSV.

Our first goal is to derive Elmore delay equations that are the functions of x and y. In what follows, we let δx denote

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

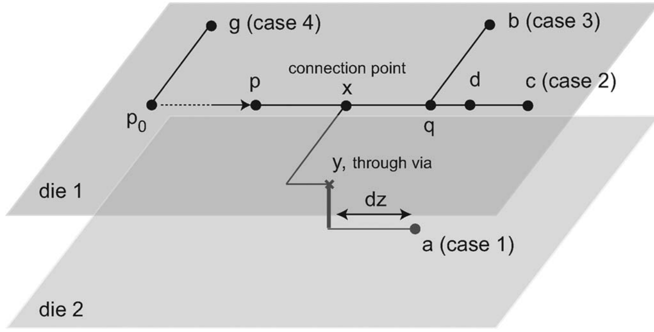

Fig. 4. Illustration of how pin a connects to $e(p,c) \in T$. $e(y,a)$ is routed in the bottom die, whereas all other edges are routed in the top die. x is the location of connection point on $e(p,c)$. y is the location of the TSV inserted on $e(x,a)$. $e(q,b)$ is another branch in T. g is another sink that is not a part of the subtree rooted at p. d is the shortest distance point on $e(p,c)$ from a, and δz is the distance between the TSV and a. The Elmore delay of $T \cup a$ is a function of both x and y.

the distance between nodes p and x, and δq, δa, δb, δc, and δd are defined similarly. δy is the distance between x and y, and δz is the distance between y and a. Let T_b denote the subtree rooted at node b. In order to compute the Elmore delay change on all sink pins in T caused by adding a to T, we consider the following four cases:

1) delay at the node to be added ($=$ node a);
2) delay at the subtree located after the connection point ($=$ node c);
3) delay at the subtree that could be located either before or after the connection point ($=$ node b);
4) delay of the nodes not in T_p.

Fig. 4 shows these four cases.

1) Case 1: We handle the delay at node a. In this case, $d(a)$ is a sum of four functions

$$d(a) = f_1 + f_2 + f_3 + f_4.$$

f_1 is the delay from node p_0 to p. The delay from node p to a can be further divided into the following: 1) the delay from node p to x ($= f_2 + f_3$), and 2) the delay from node x to a ($= f_4$). In addition, the delay from node p to x depends on edge $e(q,b)$. Thus, we consider the delay from p to x as a summation of two terms f_2 and f_3, where f_2 is the delay from p to x without considering $e(q,b)$ and T_b. f_3 is the additional delay from p to x when considering $e(q,b)$ and T_b. Thus

$$\begin{aligned} f_1 &= K_0 + K_1 \{c_1\delta y + C_{\text{via}} + c_2\delta z + c_1\delta c \\ &\qquad + C_c + C_b + c_1(\delta b - \delta q)\} \\ f_2 &= r_1\delta x\left(c_1\frac{\delta x}{2} + c_1\delta y + C_{\text{via}} + c_2\delta z + c_1(\delta c - \delta x) + C_c\right) \\ f_3 &= \begin{cases} r_1\delta x\,(c_1(\delta b - \delta q) + C_b), & \text{if } \delta x \le \delta q \\ r_1\delta q\,(c_1(\delta b - \delta q) + C_b), & \text{if } \delta x \ge \delta q \end{cases} \\ f_4 &= r_1\delta y\left(c_1\frac{\delta y}{2} + C_{\text{via}} + c_2\delta z\right) \\ &\quad + R_{\text{via}}\left(\frac{C_{\text{via}}}{2} + c_2\delta z\right) + r_2c_2\frac{\delta z^2}{2} \end{aligned}$$

where $\delta z = \delta a - (\delta x + \delta y)$, K_0 is the sum of resistance and capacitance products along the $p_0 \to p$ path, K_1 is the sum of resistances along the $p_0 \to p$ path, and C_i is the capacitance of the subtree rooted at node i.

2) Case 2: The new delay at node c is given by

$$d(c) = f_1 + f_2 + f_3' + f_4'$$

where

$$\begin{aligned} f_3' &= r_1\delta q\,(c_1(\delta b - \delta q) + C_b) \\ f_4' &= r_1(\delta c - \delta x)\left\{\frac{c_1(\delta c - \delta x)}{2} + C_c\right\} \end{aligned}$$

f_2' is the delay seen at node c due to branch $e(q,b)$, and f_4' is the delay from node x to node c without considering branch $e(q,b)$.

3) Case 3: The new delay at node b is given by

$$d(b) = f_1 + f_2'' + f_3''$$

where

$$\begin{aligned} f_2'' &= \begin{cases} r_1\delta x(c_1\delta y + C_{\text{via}} + c_2\delta z), & \text{if } \delta x \le \delta q \\ r_1\delta q(c_1\delta y + C_{\text{via}} + c_2\delta z), & \text{if } \delta x \ge \delta q \end{cases} \\ f_3'' &= r_1\delta q\left\{c_1\frac{\delta q}{2} + c_1(\delta b - \delta q) + C_b + C_c\right\}. \end{aligned}$$

f_2'' is the added delay at node b from adding a new pin a. f_3'' is the delay from node p to q without considering the effect of the new pin a.

4) Case 4: For all other nodes not in T_p, the added delay is a function of the added capacitance, which is linear in terms of x and y and given by

$$\Delta C = c_1(\delta x + \delta y) + C_{\text{via}} + c_2\delta z.$$

These cases identify the four possible ways by which the delay at the new node a and the other existing nodes of the tree may change due to the addition of a. The objective is to find the location of connection point x and the location of TSV y for the new node a such that the total increase in delay is minimal under the given thermal profile. As discussed earlier, we use this connection-point computation to identify the pin–edge pair that results in the minimum increase of maximum Elmore delay under the given thermal distribution.[6]

C. Optimization of Delay Equations

We discuss how the delay equations derived in the previous section can be used to generate a small set of possible optimum location points. We first consider the conditions needed to determine the minimum of a general quadratic function of two variables. We later show how the delay equations derived in the previous section can be optimized using these conditions.

In general, for a quadratic function of two variables $F(\delta x, \delta y)$, the maximum or the minimum of the function

[6]In the case of connecting two pins located in nonadjacent dies, we use a stacked TSV so that no routing in the intermediate layers is used. For example, a pin in dies 1 and 3 requires two TSVs that are vertically aligned so that there is no routing necessary in die 2.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

depends upon the values of $\partial^2 F/\partial\delta x^2$ and the determinant of the Hessian matrix H_1

$$\begin{vmatrix} \frac{\partial^2 F}{\partial\delta x^2} & \frac{\partial^2 F}{\partial\delta x\partial\delta y} \\ \frac{\partial^2 F}{\partial\delta x\partial\delta y} & \frac{\partial^2 F}{\partial\delta y^2} \end{vmatrix}$$

where F is the delay function under consideration. The aforementioned values for a quadratic function of two variables are always constant.

We have $0 \leq \delta x \leq \delta d$ and $0 \leq \delta y \leq \delta a$, so we consider the following cases:

Case 1) If $(\partial^2 F/\partial\delta x^2) \leq 0$ and $H_1 \geq 0$, the minimum can be found at the boundary points, i.e., $\delta x = 0$ or $\delta x = \delta d$ and $\delta y = 0$ or $\delta y = \delta a$. Thus, we have four points to look for the minimum.

Case 2) If $(\partial^2 F/\partial\delta x^2) \leq 0 (\partial^2 F/\partial\delta y^2) \leq 0$ and $H_1 = 0$, we have a concave function, and the minimum lies on the boundary points.

Case 3) If $(\partial^2 F/\partial\delta x^2) = 0$, $(\partial^2 F/\partial\delta y^2) = 0$, and $H_1 = 0$, then $F(\delta x, \delta y)$ is a linear function of δx and δy, and the minimum lies at the boundary points.

Case 4) If $H_1 < 0$, the critical point found is a saddle point, and the minimum lies at the boundary. The set of boundary points may be found by setting $\delta x = 0$ or $\delta x = \delta d$ and minimizing $F(\delta x, \delta y)$ as a function of δy or by setting $\delta y = 0$ or $\delta y = \delta a$ and minimizing $F(\delta x, \delta y)$ as a function of δx.

We show that the Elmore delay at each sink node in T can be optimized by considering any of the four cases shown previously. Thus, there is only a fixed number of points (x, y) for which the Elmore delay values are minimized. Details are included in Appendix A.

V. 3-D Tree Refinement With TSV Relocation

A. Overview of the Algorithm

The motivation behind our TSV relocation is to move as many TSVs into thermal hot spots as possible while preserving the original tree topology that we obtain during our construction step. The TSVs in hot spots reduce the thermal resistance in these areas and establish heat-conducting paths to the heat sink. The objective is to remove hot spots while not violating the timing and routing-capacity constraints.

TSVs are usually etched or drilled through device layers by special techniques and are costly to fabricate [3]. Thus, a large number of TSVs will degrade the yield and reliability of the 3-D chip. This is the drawback of the existing works that add *additional* dummy TSVs to reduce temperature [2]–[7]. During our TSV relocation, however, no new dummy TSVs are added, but the existing TSVs are relocated. The objective is to reduce the maximum on-chip temperature as much as possible using TSV relocation so that the additional TSVs needed may be kept at minimum.[7]

[7] An approach that combines both TSV relocation and dummy TSV insertion should provide the best results and is outside the scope of this paper.

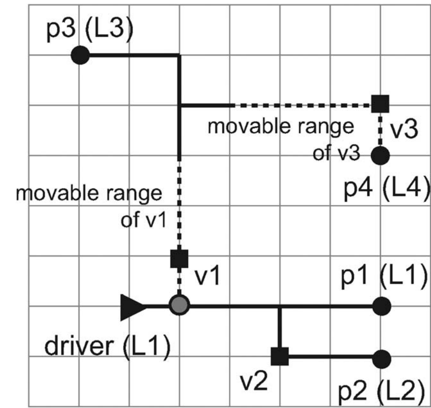

Fig. 5. Illustration of the movable range. This figure is a top–down view of a multidie stack. The driver is represented by a triangle, the sinks are represented by the dots, and the TSVs are represented by the square boxes. The driver is located in die 1 (= L1). The movable range of each TSV is represented by the dotted lines.

In general, thermal optimization with TSVs is nonlinear in nature due to the well-known relation $T = PR$, where T is the temperature matrix, P is the power vector, and R is the thermal resistance matrix. We have $R \propto (1/a)$, where a is the number of TSVs. In addition, general solutions available for solving nonlinear problems cannot be applied directly to large-size problems. In this paper, we propose a novel solution that helps us effectively overcome the nonlinear nature of this problem. We propose a relaxed ILP-based formulation in which the number of integer variables is kept at minimum. Our ILP-based method optimizes TSVs on *all* nets simultaneously, which is more rigorous than a sequential approach that optimizes the nets one by one. In addition, we target all hot spots simultaneously instead of iteratively targeting one by one. Our experimental results in Section VI demonstrate the advantage of this approach.

B. Movable Range

We start our TSV relocation phase with the set of 3-D Steiner trees that we obtain from the construction step. All pins in each tree T_i are associated with the timing constraint that denotes the required arrival time in terms of Elmore delay. Each TSV $v \in T_i$ is associated with the *movable range* that denotes the range of new location along its route to the connection point so that the timing constraints are not violated. We perform thermal-aware static timing analysis to compute timing slack for all points. This static timing analysis gives us the available slack on each pin. The TSVs are then moved along the Steiner tree edge to determine the movable range of each TSV. Note that new TSV location translates to new delay at the sinks. The range ensures that no timing constraints are violated during the relocation. An illustration is shown in Fig. 5. In case the movable range of a TSV v is a single point, v is nonmovable; otherwise, it is movable. Our goal is then to find a new location for each movable TSV in each Steiner tree so that the maximum temperature among all nodes in the thermal grid is minimized while the timing and routing-resource capacity constraints are not violated. Note that we preserve the original topology of the Steiner trees. All that is changing is the location of TSVs for thermal

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

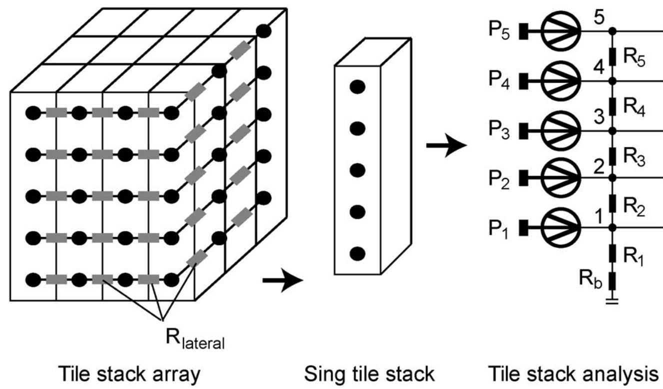

Fig. 6. Thermal model used, where R_b denotes the thermal resistance to the heat sink. The convention used in our ILP formulation is that adding TSVs at point (= tile) i reduces the value of R_i.

optimization, where movable TSVs are moved into thermal hot spots under the timing constraint to reduce thermal resistivity.

C. Compact Thermal Analysis

Fig. 6 shows the fast thermal model used in our TSV relocation method, which we adopted from [3]. In this model, each heat source is considered as a current source, and the temperature is regarded as a voltage level. The 3-D structure is divided into smaller region, which is represented by its thermal resistance. In this model, a tile structure is imposed on the surface, where each tile is approximated as a resistive chain, as shown in Fig. 6. Temperature equations are then constructed based on the voltage equation $V = I \cdot R$. For example, the temperature at node 4 is given by $T_4 = T_3 + (P_5 + P_4) \cdot R_5$.

In 3-D ICs, heat sinks are attached to the bottom or top side of the 3-D IC stack, with other boundaries being adiabatic. Thus, the dominant heat flow is in the vertical direction. For the purpose of optimization, we view each tile stack in Fig. 6 as an independent thermal resistive chain. In this case, we do not consider effects of lateral thermal dissipation, which can be justified by the fact that the thermal conductivity of the epoxy material used for bonding is much lower than that of silicon itself. This essentially means that it is difficult for heat to dissipate in the vertical direction as compared with that in the horizontal direction. To accurately verify the temperature reduction, a full (= vertical and lateral) resistive thermal model [13] (considering lateral resistances) is run twice, i.e., once before and once after our TSV relocation phase. The final temperature values reported in our experiments are based on this full resistive model.

Another important reason for our adoption of this vertical heat-flow model is that we can formulate our simultaneous TSV relocation using ILP and solve it efficiently, as will be discussed in the subsequent sections. To solve for the temperature values at all nodes, all temperature equations are constructed and reduced to the form $T = P \cdot R$, where T, P, and R are all vectors. These equations can be solved directly by using the values of power and the thermal resistance at each tile.

D. NLP Formulation

In the following sections, we will first show how the TSV relocation problem can be formulated as an NLP formulation. We then show how the NLP is converted into an ILP formulation. The ILP formulation adds a large number of integer variables in the problem, thus making it difficult to solve. Last, we present our fast-ILP problem formulation that reduces the number of integer variables significantly.

TABLE I
VARIABLES AND CONSTANTS USED IN OUR NLP/ILP FORMULATIONS

Symbol	Description
$T_{i,j,k}$	temperature at tile (i,j,k)
$\alpha_{i,j,k}$	temperature-related weight for tile (i,j,k), which is computed by $T^{org}_{i,j,k}/T^{org}_{max}$, where $T^{org}_{i,j,k}$ denotes the original temperature for (i,j,k) before the optimization, and T^{org}_{max} is the maximum value among all $T^{org}_{i,j,k}$ values. *constant*
$vmax$	maximum number of TSVs each tile can accommodate. *constant*
$\beta^m_{i,j,k}$	becomes 1 when a TSV is moved to tile (i,j,k) so that the total number of movable TSVs at (i,j,k) changes from $m-1$ to m.
$V^{org}_{i,j,k}$	original number of TSVs (= movable + non-movable) in tile (i,j,k) before the optimization. *constant*
$V^m_{i,j,k}$	number of TSVs in tile (i,j,k), which is just m.
$V^{opt}_{i,j,k}$	number of TSVs (= movable + non-movable) in tile (i,j,k) after the optimization.
$M^{x,y,k}_{i,j,k}(n)$	becomes 1 if a TSV in net n is moved from tile (i,j,k) to (x,y,k); 0 otherwise.
$G^{cur}_{i,j,k}$	current usage of wires and TSVs for a grid (i,j,k) in 3D routing grid G.
$G^{max}_{i,j,k}$	capacity constraint of wires and TSVs for a grid (i,j,k) in 3D routing grid G. *constant*
$R^m_{i,j,k}$	thermal resistance of tile (i,j,k) having m TSVs. *constant*
$R^{no}_{i,j,k}$	thermal resistance of tile (i,j,k) with no TSVs. *constant*
α	thermal resistance of one TSV. *constant*
$\gamma^m_{i,j,k}$	becomes 1 if the number of TSVs in the tile (i,j,k) is m.
$P_{i,j,k}$	power at tile (i,j,k).
$\Delta G_{i,j,k}$	change in the routing resource usage at tile (i,j,k)
$\delta^m_{i,j,k}$	it is the temperature difference between tile i,j,k and $i,j,k-1$ if the number of TSVs in tile i,j,k is m. We have $\delta^m_{i,j,k} = (P_{i,j,n} + \cdots + P_{i,j,k}) \cdot R^m_{i,j,k}$. *constant*
N	entire netlist
N_{mov}	set of nets whose vias are movable.
K	total number of dies in the stack
$N^{in}_{i,j,k}$	set of nets that contain TSVs that are moved into tile (i,j,k).
$N^{out}_{i,j,k}$	set of nets that contain TSVs that are moved out of tile (i,j,k).

The NLP-based formulation is defined as follows (Table I explains the notations that we use in the formulation):

Minimize

$$\sum_{(i,j,k)\in Z} \alpha_{i,j,k} \cdot T_{i,j,k} \tag{1}$$

Subject to

$$T_{i,j,k} = T_{i,j,k-1} + (P_{i,j,K} + \cdots + P_{i,j,k}) \times R^{\text{opt}}_{i,j,k} \tag{2}$$

$$R^{\text{opt}}_{i,j,k} = \frac{\alpha}{\alpha/R^{\text{no}}_{i,j,k} + V^{\text{opt}}_{i,j,k}} \tag{3}$$

$$V^{\text{opt}}_{i,j,k} = V^{\text{org}}_{i,j,k} + \Delta V_{i,j,k} \tag{4}$$

$$\Delta V_{i,j,k} = \sum_{n\in N^{\text{in}}_{i,j,k}} M^{i,j,k}_{x,y,k}(n) - \sum_{n\in N^{\text{out}}_{i,j,k}} M^{v,w,k}_{i,j,k}(n) \tag{5}$$

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

$$G^{\text{cur}}_{i,j,k} \leq G^{\max}_{i,j,k} \qquad \forall (i,j,k) \tag{6}$$

$$G^{\text{cur}}_{i,j,k} = G^{\text{org}}_{i,j,k} + \sum_{n \in N} M^{v,w,k}_{x,y,k}(n) \cdot \Delta G_{i,j,k}, \text{ where } (i,j,k) \text{ is on path } (x,y,k) \rightarrow (v,w,k) \tag{7}$$

$$\sum_{n \in N_{\text{mov}}} M^{x,y,k}_{i,j,k}(n) = 1 \tag{8}$$

$$M^{x,y,k}_{i,j,k}(n) \in \{0,1\}. \tag{9}$$

Equation (1) is our objective function, where we minimize the weighted sum of temperature values at all thermal tiles.[8] The weights $\alpha_{i,j,k}$ are computed based on the initial temperature measured before the TSV relocation. In this case, the higher the $\alpha_{i,j,k}$, the lower the $T_{i,j,k}$ that we desire.[9]

Equation (2) gives the temperature at each tile based on our fast thermal model shown in Fig. 6, where K is the maximum height of our thermal tile ($=$ total number of dies in the 3-D stack). Equation (3) shows the variation of thermal resistance based on the number of TSVs in a tile. This is obtained from solving the following parallel resistance relation:

$$\frac{1}{R^{\text{opt}}_{i,j,k}} = \frac{1}{R^{\text{no}}_{i,j,k}} + \frac{V^{\text{opt}}_{i,j,k}}{\alpha}.$$

Equation (4) is the definition of $V^{\text{opt}}_{i,j,k}$. Equation (5) states that the total change in the number of TSVs for tile (i,j,k) is the total number of TSVs moved into tile (i,j,k) minus the total number of TSVs moved out of tile (i,j,k). Equation (6) ensures that the routing-resource ($=$ wires and TSVs) capacity constraints are satisfied.

Equation (7) shows how the routing-resource usage is updated after a TSV is moved from tile (x,y,k) to (v,w,k). Note that the usage of *all* tiles along the path from (x,y,k) to (v,w,k) is affected. Let $G^{\text{org}}_{i,j,k}$ denote the original usage at tile (i,j,k) before the move. Then, the new usage, denote by $G^{\text{cur}}_{i,j,k}$, is computed by adding the total amount of change made on (i,j,k), where the total amount of change is computed by summing the various $\Delta G_{i,j,k}$ changes based on whether the corresponding vias are moved or not. We note that $\Delta G_{i,j,k}$ can take both positive or negative values based on whether the corresponding via move increases or decreases the routing capacity at the given location. Last, the whole process is done for all nets that contain the relocated TSVs.

Equation (8) ensures that only one TSV per net is moved. Note that this restriction is unavoidable since the movable range of a TSV is computed independent of other TSVs. Once a TSV is moved, it affects the timing constraint, movability, and the range of *all* other TSVs in the same net. The ultimate way to perform TSV relocation is to consider all TSVs from all nets simultaneously, which is computationally expensive. However, our method that considers one TSV from all nets simultaneously is better than a sequential approach that considers all TSVs from a single net. Equation (9) states that $M^{x,y,k}_{i,j,k}(n)$ are binary integer variables.

We note that this original TSV relocation problem is nonlinear due to the inverse relation between thermal resistance and the number of TSVs in a tile ($= R^{\text{opt}}_{i,j,k}$ versus $V^{\text{opt}}_{i,j,k}$) in (3). In the next section, we will propose a simplified ILP formulation that overcomes this nonlinear problem formulation.

E. ILP Formulation

From the NLP formulation, we see that the number of TSVs in each tile is an integer variable. Our ILP-based formulation differs from the NLP one in the following way. We replace (2) and (3) with the following:

$$T_{i,j,k} = T_{i,j,k-1} + \gamma^0_{i,j,k} \times \delta^0_{i,j,k} + \cdots + \gamma^{vmax}_{i,j,k} \times \delta^{vmax}_{i,j,k} \tag{10}$$

$$1 \cdot \gamma^1_{i,j,k} + 2 \cdot \gamma^2_{i,j,k} + \cdots + vmax \cdot \gamma^{vmax}_{i,j,k} = V^{\text{opt}}_{i,j,k} \tag{11}$$

$$\sum_{m=0}^{vmax} \gamma^m_{i,j,k} = 1 \qquad \forall (i,j,k) \tag{12}$$

$$\gamma^m_{i,j,k} \in \{0,1\}. \tag{13}$$

Equation (10) is a new way to calculate the temperature at each tile (refer to Table I for the definition of the related variables and constants). In this equation, $\gamma^m_{i,j,k}$ denotes the new integer variables, whereas $\delta_{i,j,k}$ represents the constants that are calculated for each possible value of the number of TSVs in a tile. Equation (11) equates the $\gamma^m_{i,j,k}$ variable with the optimum number of TSVs in a tile. Compared with the nonlinear equation (3), (11) shows linear relation between $V^{\text{opt}}_{i,j,k}$ and $\gamma_{i,j,k}$. Equation (12) ensures that, for each tile, only one $\gamma^m_{i,j,k}$ takes a value of 1. Last, (13) ensures that $\gamma^m_{i,j,k}$ is either 0 or 1. All other equations in our NLP formulation remain the same in our ILP formulation.

The number of new integer variables $\gamma^m_{i,j,k}$ is proportional to the number of tiles in our thermal grid plus the number of TSVs that are movable in each grid. Adding such a large number of integer variables makes the problem harder to solve. In our next section, we propose our fast-ILP formulation, which removes the need of integer $\gamma^m_{i,j,k}$ variables, thus reducing the number of integer variables required significantly.

F. Fast-ILP Formulation

Our fast ILP-based TSV relocation is formulated as follows (Table I explains the notations that we use in the formulation):

Minimize

$$\sum_{(i,j,k) \in Z} \alpha_{i,j,k} \cdot T_{i,j,k} \tag{14}$$

Subject to

$$T_{i,j,k} = T_{i,j,k-1} + \delta^0_{i,j,k} - \beta^1_{i,j,k} \cdot \left(\delta^0_{i,j,k} - \delta^1_{i,j,k}\right) - \cdots - \beta^{vmax}_{i,j,k} \cdot \left(\delta^{vmax-1}_{i,j,k} - \delta^{vmax}_{i,j,k}\right) \tag{15}$$

[8]Thermal tiles and thermal-grid nodes are used interchangeably in this section.

[9]Note that this objective is not directly minimizing the maximum temperature among all tiles. This is necessary in our formulation; otherwise, we can no longer relax the $\beta^i_{i,j,k}$ variables to be continuous, as explained in Appendix B, which, in turn, means that our ILP formulation will contain excessive amount of integer variables. We note, however, that giving larger weights to the higher temperature tiles in the objective function helps us ensure that hot spots are given a preference.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

$$\beta^1_{i,j,k} + \beta^2_{i,j,k} + \cdots + \beta^{vmax}_{i,j,k} = V^{\rm opt}_{i,j,k} \quad (16)$$

$$V^{\rm opt}_{i,j,k} = V^{\rm org}_{i,j,k} + \Delta V_{i,j,k} \quad (17)$$

$$\Delta V_{i,j,k} = \sum_{n \in N^{\rm in}_{i,j,k}} M^{i,j,k}_{x,y,k}(n) - \sum_{n \in N^{\rm out}_{i,j,k}} M^{v,w,k}_{i,j,k}(n) \quad (18)$$

$$G^{\rm cur}_{i,j,k} \le G^{\rm max}_{i,j,k}, \qquad \forall (i,j,k) \quad (19)$$

$$G^{\rm cur}_{i,j,k} = G^{\rm org}_{i,j,k} + \sum_{n \in N} M^{v,w,k}_{x,y,k}(n) \cdot \Delta G_{i,j,k}, \text{ where } (i,j,k) \text{ is on path } (x,y,k) \to (v,w,k) \quad (20)$$

$$0 \le \beta^m_{i,j,k} \le 1 \quad (21)$$

$$M^{x,y,k}_{i,j,k}(n) \in \{0,1\} \quad (22)$$

$$\sum_{n \in N_{\rm mov}} M^{x,y,k}_{i,j,k}(n) = 1. \quad (23)$$

Equation (15) is a new way to compute the temperature at tile (i,j,k), which is different from (10). A detailed explanation of this equation is included in Appendix B. Equation (16) states that the total number of TSVs in a tile (i,j,k), which is specified by the $\beta^i_{i,j,k}$ values $(1 \le i \le vmax)$, should be equal to $V^{\rm opt}_{i,j,k}$. Equation (21) restricts the range of values that $\beta^n_{i,j,k}$ can take. All other equations are the same as that discussed in the NLP formulation.

A few points are worth mentioning. First, in order to overcome the restriction of moving just one TSV per net [= (23)], we repeat the entire relaxed ILP multiple times so that multiple TSVs from a single net are given a chance to relocate in an iterative fashion. We stop the iteration if the improvement on both the maximum and average temperatures is minimal. Our related experiment shown in Table V suggests that most of the temperature saving is obtained during the first iteration and that the overall algorithm converges within a small number of iterations. Second, the number of integer variables $(= M^{x,y,k}_{i,j,k}(n))$ can be huge if the number of nets is larger or the thermal grid is finer. This makes our fast-ILP formulation less desirable for a large-problem instance. However, we overcome this limitation by relaxing these integer M variables and solving our fast-ILP problem. We round the continuous variables based on a threshold value $\lambda = 0.5$. All variables above λ are converted into 1, provided that they do not violate the routing-capacity constraint, whereas all other variables are converted into zero. Table VI shows the impact of this relaxation on solution quality and runtime.

VI. Experimental Results

A. Experimental Setting

We implemented our router named 3-D Elmore Router in C++/STL and ran our experiments on a Linux server running at 2.5 GHz and having 16 GB of memory. We tested our algorithms with three sets of benchmark: ISCAS89, ITC99, and ISPD98. We report the total wirelength, the total number of TSVs used, the maximum thermal-aware Elmore delay among all sinks, the maximum temperature among all nodes in the thermal grid, and the runtime in seconds for each circuit. We obtained 3-D placement using an algorithm that is similar to that in [16]. The following are the details of our experimental setting.

TABLE II
Three-Dimensional Routing-Grid Dimensions and Edge Capacities for the Four-Die Stack

		XYZ capacities		
ckt	grid dim	X & Y	Z (F2F)	Z (B2B)
s9234	20×20	8	3	1
b14_opt	40×40	8	3	2
s13207	40×40	10	4	2
s15850	40×40	10	5	4
b20_opt	40×40	16	6	4
b21_opt	60×60	16	6	4
b22_opt	60×60	20	7	5
ibm09	80×80	60	20	10
ibm10	100×100	75	20	10
ibm11	120×120	85	25	12
ibm13	140×140	95	25	12
ibm17	140×140	300	40	20

1) We use four-die stacking for our 3-D IC, where the top two and bottom two dies are bonded face to face and the middle two back to back, unless otherwise specified.
2) We assume that all four dies have different unit-length resistance and capacitance values [11] as follows, unless otherwise specified: $r_1 = 86\ \Omega$/mm and $c_1 = 396$ fF/mm, $r_2 = 175\ \Omega$/mm and $c_2 = 100$ fF/mm, $r_3 = 74\ \Omega$/mm and $c_3 = 279$ fF/mm, and $r_4 = 154\ \Omega$/mm and $c_4 = 120$ fF/mm.
3) The dimensions of our face-to-face TSVs are $1\ \mu$m $\times$ $1\ \mu$m $\times$ $10\ \mu$m. The parasitics are $R_{\rm via} = 17.2\ \Omega$/mm and $C_{\rm via} = 371.8$ fF/mm. The dimensions of our back-to-back TSVs are $10\ \mu$m $\times$ $10\ \mu$m $\times$ $40\ \mu$m. The parasitics are $R_{\rm via} = 0.172\ \Omega$/mm and $C_{\rm via} = 1943.8$ fF/mm.
4) The routing-grid dimensions used for the four-die stack are shown in Table II. The dimensions were increased based on the circuit size (= number of nets), and the routing capacities were chosen so that about 10% of the nets need to be rerouted after the initial tree generation.
5) The thermal-grid dimensions are $20 \times 20 \times 4$ for the four-die-stacked 3-D IC. For thermal analysis, we use the following thermal conductivity values: Silicon is 150 W/mK, copper is 285 W/mK, and epoxy (= bonding) layer is 0.05 W/mK. The power generated in each thermal grid is proportional to the number of cells placed in it, multiplied by a random value ranging from 1 to 10^7 W/m^2 to account for the gate-level switching activity factor. We use our compact thermal model discussed in Section V-C for TSV relocation and that in [13] for all other purposes.

B. Tree Construction Results

We implement two existing 3-D routers for comparison. The first is a 3-D maze router by Cong and Zhang [2], as discussed in Section II. The second is a 3-D A-tree router, where we extend the original 2-D version [15] into a 3-D one. More specifically, we first converted the 3-D problem into a 2-D one by mapping the pin locations to the 2-D plane. Then, we perform 2-D A-tree routing [15]. Last, our 2-D solution is translated back into a 3-D one, where the connection to the pins not located on the same die as the driver is made using TSVs.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

TABLE III
COMPARISON BETWEEN 3-D MAZE [2], 3-D A-TREE [15], AND OUR 3-D ELMORE ROUTERS. THE "V-BOUND" COLUMN SHOWS THE LOWER BOUND ON TSV COUNT

ckt	# nets	v-bound	3D Maze Router [2] wire	delay	TSV	cpu	3D A-tree Router [15] wire	delay	TSV	cpu	3D Elmore Router wire	delay	TSV	cpu
s9234	5844	5563	0.167	0.072	5700	7.88	0.169	0.047	5734	1.42	0.18	0.023	5613	2.59
b14_opt	5646	5546	0.19	0.086	7366	10.1	0.21	0.071	7818	5.48	0.23	0.072	6937	5.95
s13207	8727	8298	0.39	0.091	9707	15.06	0.41	0.077	10342	7.77	0.43	0.064	9595	9.43
s15850	10397	9915	0.49	0.11	11763	46.1	0.51	0.11	12924	10.2	0.56	0.092	11516	15.7
b20_opt	12501	12174	0.947	0.31	18423	108.5	1.04	0.25	21730	46.8	1.19	0.22	18703	57.9
b21_opt	12678	12288	0.97	0.29	18627	128.9	1.06	0.138	21688	44.3	1.2	0.145	19314	58.4
b22_opt	18086	17911	2.15	0.48	27116	186.1	2.33	0.32	31255	87.4	2.64	0.24	29090	104.7
ibm09	52989	53483	29.4	222.2	104481	639.72	33.01	124.7	122465	245.4	36.9	103.6	118013	353.6
ibm10	68004	67651	51.4	600.9	136071	1263.9	57.4	337.5	157983	349.9	61.9	273.3	153483	601.5
ibm11	70028	70524	53.3	457.9	131815	1859.4	60.5	264.6	152422	456.8	61.2	218.4	141922	711.4
ibm13	84191	82989	83.5	953.3	167311	4000.53	90.7	509.4	195459	1113.6	94.7	438.7	183813	1411.7
ibm17	184227	180001	157.4	1723.4	398145	7754.3	169.8	1124.6	408861	2122.7	172.4	899.8	405400	2456.7
RATIO	-	-	1.00	1.00	1.00	1.00	1.09	0.59	1.1	0.28	1.15	0.48	1.06	0.36

Table III shows a comparison between 3-D maze [2], 3-D A-tree [15], and 3-D Elmore routers. Our baseline is the 3-D maze router. We observe that our 3-D Elmore router achieves 52% average delay improvement over the 3-D maze router and 11% improvement over the 3-D A-tree router. The delay reported is the maximum path delay of the final layout, which we obtain using a static timing analyzer. The TSV count is comparable between the 3-D Elmore router and the 3-D A-tree router, while the 3-D maze router uses 6% less TSVs. In the case of wirelength, the 3-D maze and 3-D A-tree routers obtained comparable results, but our 3-D Elmore router uses 15% higher wirelength. Last, our 3-D Elmore router runs three times faster than the 3-D maze router and about 40% slower than the 3-D A-tree router.

The "v-bound" column in Table III shows the lower bound of the TSV usage for each circuit. For MD2P nets, the lower bound is the number of dies in between the two pins plus 1. For MDMP nets, we use the fewest possible TSVs to connect all pins in the dies. We see that the number of TSVs needed by the 3-D maze or 3-D Elmore router is about twice as many as the minimum required. The number of TSVs used in the 3-D A-tree algorithm is the highest.[10]

We observe from circuit `b21_opt` that 3-D A-tree performs better than our 3-D Elmore in terms of performance. We observe that this was due to congestion that caused a larger number of nets to be ripped and rerouted in our 3-D Elmore algorithm. In some cases, we observe that 3-D A-tree caused lower congestion and thus required less rerouting.

C. Delay and Wirelength Distribution

Our first goal is to collect the wirelength and delay statistics among the four types of nets in the 3-D Steiner routing mentioned in Section III-A: SD2P, SDMP, MD2P, and MDMP nets. Table IV shows the statistics, where we report the average max-sink delay and wirelength values among all nets in each type. We observe that MDMP nets have the largest delay and wirelength on average, which suggests that MDMP nets are the hardest to route in general (12.4× delay and 8.1× wirelength compared with that of SD2P nets). This is reasonable since they contain multiple pins in multiple dies and thus require multiple TSVs. We also observe that multipin nets incur larger delay and wirelength compared with two-pin nets (SDMP versus SD2P and MDMP versus MD2P). We also observe that MD2P nets have 8% smaller delay compared with SD2P ones on average. This is primarily due to the benefit of 3-D connection, where TSV-based 3-D connections tend to have smaller delay.

[10]Our 3-D A-tree is a performance-oriented router to be used for delay comparison, and TSV minimization is not considered.

TABLE IV
DELAY AND WIRELENGTH DISTRIBUTION AMONG THE FOUR DIFFERENT TYPES OF NETS. WE REPORT THE AVERAGE MAX-SINK DELAY AND WIRELENGTH VALUES AMONG ALL NETS IN EACH TYPE

ckt	SD2P dly	wire	SDMP dly	wire	MD2P dly	wire	MDMP dly	wire
s9234	0.004	0.014	0.005	0.029	0.005	0.013	0.128	0.107
b14_opt	0.005	0.012	0.015	0.043	0.003	0.011	0.32	0.116
s13207	0.020	0.028	0.06	0.08	0.018	0.029	0.6	0.2
s15850	0.023	0.031	0.12	0.075	0.02	0.03	0.63	0.21
b20_opt	0.021	0.03	0.11	0.08	0.019	0.028	0.68	0.246
b21_opt	0.022	0.03	0.13	0.085	0.018	0.029	0.65	0.24
b22_opt	0.047	0.042	0.16	0.153	0.041	0.046	1.1	0.36
ibm09	0.71	0.189	2.3	0.6	0.64	0.19	8.5	1.64
ibm10	1.24	0.25	3.59	0.79	1.09	0.22	12.6	1.95
ibm11	1.24	0.248	4.28	0.81	1.21	0.21	14.3	1.98
ibm13	2.15	0.34	6.4	1.04	2.01	0.33	28.4	2.85
ibm17	2.3	0.37	7.8	1.2	2.1	0.36	29.2	2.9
RATIO	1.0	1.0	3.2	3.51	0.92	0.94	12.4	8.1

D. TSV Relocation Results

To evaluate the effectiveness of our TSV relocation algorithm, we implemented a fast greedy algorithm that tries to move TSVs into thermal hot spots in an iterative fashion. We choose a single hot spot and relocate movable TSVs into it. We then repeat this process for the next hot spot until no more temperature improvement can be obtained. In addition, we developed two TSV relocation methods based on our ILP formulation introduced in Section V-F: single ILP and multiple ILP. Under the single-ILP method, we perform our ILP-based TSV relocation once, while under the multiple-ILP method, we repeat the ILP-based TSV relocation until there is no more gain on temperature reduction. In this case, we report the number of iterations taken. Note that our ILP-based methods target *all* hot spots simultaneously.

Table V shows the maximum temperature, average temperature, and standard deviation obtained by the greedy method and our ILP-based methods (single iteration versus multiple

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

TABLE V
TSV RELOCATION RESULTS. $T_{\max}$, T_{ave}, T_{std} DENOTE THE MAXIMUM TEMPERATURE, AVERAGE TEMPERATURE, AND STANDARD DEVIATION AMONG ALL THERMAL TILES, RESPECTIVELY. THE RUNTIME IS IN SECONDS

	initial temperature			greedy				single ILP				multiple ILP				
ckt	T_{max}	T_{avg}	T_{std}	T_{max}	T_{avg}	T_{std}	cpu	T_{max}	T_{avg}	T_{std}	cpu	T_{max}	T_{avg}	T_{std}	cpu	itr
s9234	92.6	52.3	23.7	91.9	51.6	23.7	2.31	84.5	45.1	20.7	67.3	83.1	40.1	18.3	246.3	5
b14_opt	114.5	46.1	32.3	114.1	45.8	32.2	1.9	107.6	38.7	27.8	89.6	105.8	33.6	25.1	315.5	6
s13207	112.1	66.2	27.6	111.3	66.1	27.6	1.88	101.2	53.2	24.3	82.6	101.1	52.8	24.0	196.4	3
s15850	115.1	44.3	34.3	114.0	44.0	34.1	2.33	103.1	37.5	27.6	175.4	102.6	35.3	26.9	324.5	3
b20_opt	108.3	38.9	37.3	108.1	38.8	37.0	2.87	98.7	30.3	32.3	209.8	96.4	27.6	29.8	813.2	7
b21_opt	114.1	45.7	24.2	113.5	45.5	24.0	3.12	109.4	37.8	20.1	206.3	108.4	35.7	19.4	612.3	3
b22_opt	114.5	54.6	18.4	114.2	54.6	18.4	3.66	105.1	49.6	15.2	228.8	104.5	47.8	14.7	388.9	2
ibm09	94.2	47.8	23.1	93.8	47.3	22.9	11.3	87.1	43.6	21.2	702.3	84.8	39.5	18.7	2113.4	6
ibm10	113.4	54.2	21.0	112.9	54.0	20.9	13.2	105.6	45.2	16.4	452.3	104.0	32.3	18.3	1566.5	5
ibm11	108.4	45.3	27.8	107.7	45.1	27.7	19.6	99.7	35.2	21.0	832.5	97.9	32.3	18.3	4211.3	8
ibm13	95.1	52.4	27.9	94.5	52.1	27.7	24.2	86.6	43.8	22.3	1387.5	86.1	41.8	21.8	2346.4	2
ibm17	111.2	48.6	19.2	110.8	48.2	19.0	38.2	101.5	41.7	15.4	3045.5	101.0	39.6	14.1	6836.7	3
RATIO	1.00	1.00	1.00	0.99	0.99	0.99	1.00	0.91	0.84	0.83	64.3	0.90	0.78	0.77	160.1	-

TABLE VI
IMPACT OF M-VARIABLE RELAXATION (FAST ILP VERSUS SLOW ILP) IN TERMS OF MAXIMUM TEMPERATURE AND RUNTIME

	fast-ILP		slow-ILP	
ckt	T_{max}	cpu	T_{max}	cpu
s9234	84.5	1.1 min	80.1	234 min
b14_opt	107.6	1.4 min	99.5	342 min
s13207	101.2	1.3 min	94.2	> 1-day
s15850	103.1	2.9 min	98.4	> 1-day
RATIO	1.0	1.0	0.94	-

iteration). We observe that our ILP-based simultaneous methods achieve consistent improvement over the greedy approach at the expense of additional runtime. We obtained 9% maximum-temperature and 16% average-temperature reductions with our ILP-based methods, whereas the greedy method improves the maximum/average temperature by only 1%. Note that this free saving does not require any additional area for dummy TSV insertion. The runtime for our biggest circuit `ibm17` that contains 184 000 nets is around 3045 s for single ILP. This shows that our fast-ILP method scales well with the complexity of the circuit while maintaining high-quality solutions. In Table V, we also show the impact of multiple iterations on our fast ILP. We observe that the temperature saving between single ILP versus multiple ILP is comparable for the maximum temperature. In the case of average temperature and standard deviation, however, our multiple ILP outperformed single ILP by 6%. We also observe that our multiple-ILP method converges quickly to a high-quality solution within a few iterations.

Table VI shows the impact of the M-variable relaxation in our fast-ILP method. Due to the excessive runtime of our original/slow ILP, we used the four smallest circuits. In the case of the two bigger circuits, we gave our slow ILP one full day and have reported the best solution discovered so far. We observe that our slow ILP obtained 6% better results on the maximum temperature, but the runtime required is prohibitive. Based on this runtime trend, one can expect that our slow ILP will not be able to handle bigger circuits, which, in turn, shows that our M-variable relaxation is the key to making our ILP method scalable.

E. Impact of TSV Dimension and Parasitics

Next, we investigate the impact of TSV dimension and parasitics on delay, wirelength, TSV count, and temperature. Note that TSVs play an important role in determining the overall routing topology and the underlying thermal profile. Table VII shows three different TSV sizes and their RC parasitics that we use in our experiment. Size II is the default set that is used in all of the previous experiments, as discussed in Section VI-A. Table VIII shows our 3-D Elmore routing results using these sizes, where Size I, the smallest, is our baseline. We first observe that the delay increases as the TSV dimension grows. This is mainly due to the wirelength increase, which is caused by the routing congestion from using larger TSVs. We observe that these large TSVs have a detrimental impact on MDMP nets, which are more likely to become critical nets. Another factor is

TABLE VII
VARIOUS TSV SIZES AND THEIR PARASITIC RESISTANCE (IN OHMS PER MILLIMETER) AND CAPACITANCE (IN FEMTOFARAD PER MILLIMETER). DIMENSIONS ARE IN MICROMETERS

face-to-face TSV					
	width	height	depth	R	C
Size I	0.5	0.5	5	68.8	288.5
Size II	1	1	10	17.2	371.8
Size III	2	2	15	4.3	554.2
back-to-back TSV					
	width	height	depth	R	C
Size I	5	5	20	0.688	1229.1
Size II	10	10	40	0.172	1943.8
Size III	20	20	60	0.043	2798.2
face-to-back TSV					
	width	height	depth	R	C
Size I	4	4	15	1.07	982.8
Size II	8	8	35	0.267	1519.1
Size III	15	15	55	0.076	2604.3

TABLE VIII
IMPACT OF TSV DIMENSION ON DELAY, WIRELENGTH, AND TSV COUNTS. WE USE THE THREE TSV SIZES SHOWN IN TABLE VII

	Size I			Size II			Size III		
ckt	delay	wire	TSV	delay	wire	TSV	delay	wire	TSV
s9234	0.02	0.171	5694	0.023	0.18	5613	0.032	0.183	5558
b14_opt	0.048	0.213	7062	0.072	0.23	6937	0.122	0.217	6911
s13207	0.047	0.43	9749	0.064	0.43	9595	0.077	0.44	9657
s15850	0.066	0.53	11703	0.092	0.56	11516	0.163	0.57	11351
b20_opt	0.16	1.18	19274	0.22	1.19	18703	0.31	1.21	18187
b21_opt	0.125	1.2	19818	0.145	1.2	19314	0.228	1.24	18810
b22_opt	0.229	2.61	29818	0.24	2.64	29090	0.48	2.69	28847
ibm09	96.0	36.6	119125	103.6	36.9	118013	127.8	37.3	116724
ibm10	252.1	60.6	155349	273.3	61.9	153483	390.2	65.7	151519
ibm11	198.6	60.2	142990	218.4	61.2	141922	268.7	67.6	139202
ibm13	419.5	92.5	184428	438.7	94.7	183813	563.4	109.5	181456
ibm17	824.8	171.1	416458	899.8	172.4	405400	987.6	187.4	401345
RATIO	1.0	1.0	1.0	1.07	1.01	0.98	1.34	1.1	0.97

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

TABLE IX
IMPACT OF TSV DIMENSION ON TSV RELOCATION AND MAXIMUM-TEMPERATURE REDUCTION. WE USE THE THREE TSV SIZES SHOWN IN TABLE VII. T_{ini} AND $T_{\max}$ DENOTE THE MAXIMUM TEMPERATURES BEFORE AND AFTER TSV RELOCATION, RESPECTIVELY

		Size I	Size II	Size III
ckt	T_{ini}	T_{max}	T_{max}	T_{max}
s9234	92.6	88.61	84.5	82.38
b14_opt	114.5	109.3	107.6	105.1
s13207	112.1	104.2	101.2	99.7
s15850	115.1	104.1	103.1	100.4
b20_opt	108.3	101.8	98.7	95.4
b21_opt	114.1	111.5	109.4	106.2
b22_opt	114.5	108.2	105.1	105.5
ibm09	94.2	91.3	87.1	84.5
ibm10	113.4	109.8	105.6	101.4
ibm11	108.4	105.2	99.7	94.5
ibm13	95.1	89.2	86.6	82.3
ibm17	111.2	107.8	101.5	99.2
RATIO	1.0	0.95	0.91	0.89

the higher parasitic capacitance values for larger TSVs. Since the Elmore delay model penalizes heavily on the capacitance increase, having larger TSVs results in more delay increase. Next, the actual TSV count decreases as the TSV dimension increases. This is because our delay-driven router may avoid using TSVs, particularly for short interconnects, to minimize the overall delay. However, the TSV count reduction is only 3%, indicating that our 3-D Elmore router still makes a heavy use of TSVs (up to 400 000 for `ibm17`).

We also conducted experiments to observe the impact of TSV dimension and parasitics on the temperature savings obtained by our TSV relocation algorithm. The results are shown in Table IX. We observe that larger TSVs result in more temperature savings (5%, 9%, and 11%). This is mainly due to the smaller thermal resistivity for larger TSVs. We note that circuit `b22_opt` does not follow this trend. This occurs since different TSV dimensions may result in different routing solutions, thereby influencing the thermal profile and temperature saving opportunity.

F. Impact of Bonding Style

So far, the top two and bottom two dies are bonded face to face and the middle two back to back in our four-die stack, as discussed in Section VI-A. We now investigate the impact of bonding style on 3-D-routing results. In our new four-die stack, all dies are bonded face to back. We use $R_{\text{via}} = 0.267\ \Omega/\text{mm}$ and $C_{\text{via}} = 1519.1$ fF/mm for the face-to-back TSVs. One important difference between the "F2F + B2B" and "F2B-only" stacks is that the TSV upper bound is different. Face-to-face bonding allows more TSVs than face to back or back to back, primarily due to sizes. Therefore, we reran our 3-D placer to obtain a new placement that minimizes interdie connections for all face-to-back bonding. The routing results are shown in Table X. We first observe that the TSV count is considerably lower compared with that in Table III. This is mainly due to the absence of face-to-face bonding in the new F2B-only stack. On the other hand, the delay values in Table X are larger than those in Table III. This is mainly due to the larger wirelength and parasitic capacitance for the face-to-back TSVs. The reason for this wirelength increase with F2B-only stacking is that the number of interdie connections is minimized, resulting in less opportunity for wirelength reduction. Last, both Tables III and X show the same trend in terms of delay, wirelength, and TSV count among 3-D maze, 3-D A-tree, and 3-D Elmore routers.

G. Two-Die Versus Four-Die Stacking

So far, our 3-D stack contained four dies, as discussed in Section VI-A. We now use a two-die stack in our experiment, where the two dies are bonded face to face. Our wire and TSV parasitics are as follows: $r_1 = 86\ \Omega/\text{mm}$, $c_1 = 396$ fF/mm, $r_2 = 175\ \Omega/\text{mm}$, $c_2 = 100$ fF/mm, $R_{\text{via}} = 17.2\ \Omega/\text{mm}$, and $C_{\text{via}} = 371.8$ fF/mm. Table XI shows a comparison between 3-D maze, 3-D A-tree, and our 3-D Elmore router. Our baseline is the 3-D maze router. We observe that our 3-D Elmore router achieves 54% average delay improvement over 3-D maze routing and 11% improvement over 3-D A-tree. This significant delay reduction came at the cost of 13% wirelength and 10% TSV count increase compared with that of the 3-D maze router. This trend is almost the same as what we saw in Table III for the two-die-stack case.

TSV relocation results are shown in Table XII. We again observe a similar trend as in Table V: Greedy method does not produce good results, and our single-ILP method produces comparable results to the multiple-ILP method in a shorter runtime. The difference between the two- and four-die stacks is twofold. First, the temperature reduction is smaller with the two-die stack (6%) compared with the four-die stack (9%). This is because there are fewer hot spots and fewer TSVs in the two-die stack. Second, the ILP runtime is also smaller in the two-die stack due to the smaller problem size.

VII. CONCLUSION

This paper has studied two new problems that are important for 3-D stacked IC technology: 3-D Steiner tree construction and TSV relocation. Our routing algorithm is based on a constructive method, where a 3-D Steiner tree is grown by connecting a new pin to the existing tree. We derived two-variable delay equations and optimized them to compute the location of TSVs under the given thermal profile. For TSV relocation, we devised an innovative technique that helps us avoid the nonlinear optimization required for temperature optimization. Our formulation can handle large number of TSVs simultaneously for an effective temperature optimization.

APPENDIX A
OPTIMIZATION OF TWO-VARIABLE DELAY EQUATIONS

Assuming $a_0 = r_2/r_1$ and $b_0 = c_1/c_2$, the optimization of two-variable delay functions shown in Section IV-C allows the computation of x (= connection point) and y (= TSV location) as follows.

1) For $d(a)$, we have

$$\frac{\partial^2 F}{\partial \delta x^2} = r_1 c_2 (a_0 - b_0 - 2)$$
$$\frac{\partial^2 F}{\partial \delta y^2} = r_1 c_2 (a_0 + b_0 - 2)$$
$$H_1 = -(r_1 c_2)^2 \{(a_0 + b_0 - 2) 2 b_0\}.$$

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

TABLE X
IMPACT OF BONDING STYLE, WHERE WE USE FACE-TO-BACK BONDING ONLY IN OUR FOUR-DIE STACK

	3D Maze			3D A-tree			3D Elmore		
ckt	delay	wire	TSV	delay	wire	TSV	delay	wire	TSV
s9234	0.018	0.146	2195	0.019	0.155	2402	0.015	0.161	2113
b14_opt	0.078	0.194	3680	0.18	0.042	4459	0.039	0.219	3914
s13207	0.074	0.388	3497	0.07	0.41	4262	0.07	0.43	3676
s15850	0.103	0.504	4298	0.093	0.53	5561	0.083	0.57	4680
b20_opt	0.354	0.975	8210	0.228	1.06	12207	0.186	1.15	10084
b21_opt	0.38	0.966	7977	0.163	1.05	11367	0.156	1.13	9834
b22_opt	0.555	2.18	11408	0.466	2.38	16824	0.281	2.71	15185
ibm09	228.1	30.3	44667	135.7	33.9	65692	116.1	35.8	59955
ibm10	578.7	52.3	49855	355.2	58.5	77334	312.1	59.1	74329
ibm11	409.6	52.6	67076	254.1	59.7	75281	215.5	60.2	61353
ibm13	906.1	81.5	72546	491.9	87.5	98964	451.7	90.3	79688
ibm17	2234.6	174.5	160435	1224.5	187.8	192349	978.8	206.9	178013
RATIO	1.0	1.0	1.0	0.56	1.09	1.30	0.47	1.15	1.14

TABLE XI
TWO-DIE-STACK RESULTS: COMPARISON BETWEEN 3-D MAZE [2], 3-D A-TREE [15], AND OUR 3-D ELMORE ROUTERS. THE "V-BOUND" COLUMN SHOWS THE LOWER BOUND ON TSV COUNT

			3D Maze Routing				3D A-tree				3D Elmore Routing			
ckt	# nets	v-bound	delay	wire	TSV	cpu	delay	wire	TSV	cpu	delay	wire	TSV	cpu
s9234	5844	4791	0.04	0.3	5137	13.4	0.04	0.33	5250	2.76	0.017	0.3	4976	4.3
b14_opt	5646	4656	0.083	0.38	6102	12.56	0.04	0.42	6560	4.1	0.04	0.42	6056	6.22
s13207	5727	7155	0.13	0.71	7960	15.1	0.11	0.75	8338	4.77	0.99	0.79	8216	7.04
s15850	10397	8524	0.17	0.92	9616	26.2	0.82	0.98	10049	7.91	0.068	1.04	9890	10.2
b20_opt	12501	10252	0.41	1.76	14205	35.1	0.17	1.96	15724	12.3	0.15	2.1	15445	15.62
b21_opt	12678	10397	0.35	1.75	14253	35.6	0.16	1.96	16004	12.56	0.19	2.07	15650	16.4
b22_opt	18086	14831	0.99	4.1	20557	86.1	0.38	4.54	22545	40.9	0.34	4.8	23571	56.2
ibm09	52989	43455	973.5	58.5	79677	411.9	461.9	66.7	89134	234.5	388.7	68.7	93184	287.6
ibm10	68004	77763	2245.8	98.2	109685	530.5	1156.6	106.7	119065	241.2	1023.5	113.4	128630	284.1
ibm11	70028	57424	1617.6	102.4	100321	771.6	804.8	116.1	112061	303.4	657.1	125.2	113573	344.5
ibm13	84191	89075	2578.6	174.5	139355	987.6	1667.6	186.6	140774	331.5	1473.9	195.6	142872	413.5
ibm17	184227	151065	6208.3	338.9	338854	1526.7	3177.2	357.8	359394	411.3	2790.1	375.3	370599	791.1
RATIO	-	-	1.0	1.0	1.0	1.0	0.53	1.08	1.07	0.36	0.46	1.13	1.1	0.5

TABLE XII
TWO-DIE-STACK RESULTS: TSV RELOCATION RESULTS. T_{ini} AND T_{max} DENOTE THE MAXIMUM TEMPERATURES BEFORE AND AFTER TSV RELOCATION, RESPECTIVELY

		greedy		single ILP		multiple ILP		
ckt	T_{ini}	T_{max}	cpu	T_{max}	cpu	T_{max}	cpu	iter
s9234	64.5	64.2	1.04	61.4	7.04	61.4	11.25	2
b14_opt	79.3	78.1	3.12	74.5	20.56	74.2	36.8	2
s13207	66.79	66.13	2.96	62.74	22.1	61.93	48.9	2
s15850	88.23	87.6	3.12	84.3	24.5	83.2	66.9	3
b20_opt	84.6	83.9	3.67	80.2	33.1	79.8	60.4	2
b21_opt	72.8	72.1	3.5	67.7	32.6	67.1	67.8	2
b22_opt	84.4	83.5	5.12	79.2	38.9	78.6	81.3	2
ibm09	77.9	76.8	12.3	73.5	148.9	73.1	405.6	3
ibm10	81.5	80.2	13.6	77.4	156.7	77.1	280.7	2
ibm11	81.9	81.1	15.3	77.2	221.3	76.5	515.7	3
ibm13	83.5	82.4	14.8	78.1	356.9	77.5	678.4	2
ibm17	86.2	85.4	19.4	81.3	623.8	80.5	1356.7	2
RATIO	1.0	0.989	1.0	0.943	17.22	0.936	36.86	-

Thus, we see that when $H_1 = 0$, $(\partial^2 F/\partial \delta x^2) \leq 0$, and $(\partial^2 F/\partial \delta y^2) = 0$, the optimal delay is found at points according to Case 2). If $H_1 < 0$, the optimal delay is found at points according to Case 4). If $H_1 > 0$, we have $(\partial^2 F/\partial \delta x^2) \leq 0$, so the optimal delay is found at points according to Case 1).

2) For $d(b)$, we need to evaluate two cases. First, when $x \geq b$, we have $(\partial^2 F/\partial \delta x^2) = 0$, $(\partial^2 F/\partial \delta y^2) = 0$, and $H_1 = 0$. Thus, the optimal delay is found at points according to Case 3). Second, when $x \leq b$, we have $(\partial^2 F/\partial \delta x^2) = -2r_1c_2$, $(\partial^2 F/\partial \delta y^2) = 0$, and $H_1 = -(r_1c_2)^2(b_0-1)^2$. Thus, if $H_1 = 0$, the optimal delay is found at points according to Case 2). Otherwise, they are found at points according to Case 4).

3) For $d(c)$, we have $(\partial^2 F/\partial \delta x^2) = -2r_1c_2$, $(\partial^2 F/\partial \delta y^2) = 0$, and $H_1 = -(r_1c_2)^2(b_0-1)^2$. If $H_1 = 0$, the optimal delay is found at points according to Case 2). Otherwise, they are found at points according to Case 4).

4) For all other nodes not in T_p, we have $(\partial^2 F/\partial \delta x^2) = 0$, $(\partial^2 F/\partial \delta y^2) = 0$, and $H_1 = 0$ since the delay is a linear function of δx and δy. Thus, the optimal delay is found at points according to Case 3).

Since a_0 and b_0 values are dependent on the interconnect parameters at each die, we see that the number of points (x, y) is a fixed constant.

APPENDIX B
EXPLANATION OF (15)

From Fig. 6, we note that the temperature at tile (i, j, k) having n TSVs is computed as follows:

$$T_{i,j,k} = T_{i,j,k-1} + (P_{i,j,K} + \cdots + P_{i,j,k}) \times R^m_{i,j,k}.$$

We can write $\delta^m_{i,j,k}$ (refer to Table I for the definition) as follows:

$$\delta^m_{i,j,k} = (P_{i,j,K} + \cdots + P_{i,j,k}) \times R^m_{i,j,k}.$$

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.

We see that $\delta^m_{i,j,k} \propto R^m_{i,j,k} \propto 1/V^m_{i,j,k}$, so $\delta^m_{i,j,k}$ is strictly decreasing for increasing values of m and $m > 0$. It can be seen easily that the temperature of a given tile having m TSVs can be rewritten as

$$T_{i,j,k} = T_{i,j,k-1} + \delta^0_{i,j,k} - \left(\delta^0_{i,j,k} - \delta^1_{i,j,k}\right) - \cdots - \left(\delta^{m-1}_{i,j,k} - \delta^m_{i,j,k}\right).$$

We define $\Delta T^m_{i,j,k}$ as follows:

$$\Delta T^m_{i,j,k} = \delta^{m-1}_{i,j,k} - \delta^m_{i,j,k}$$

which is equal to the coefficient of variable $\beta^m_{i,j,k}$. Note that $\Delta T^m_{i,j,k}$ is strictly decreasing when m is increasing. This enables us to use noninteger values for variable $\beta^m_{i,j,k}$ (refer to Table I for its definition). The reason is that, for any value of $V^m_{i,j,k}$, $\beta^m_{i,j,k}$ will always reach its maximum allowed value of 1 before $\beta^{m+1}_{i,j,k}$ starts having a nonzero value. This is due to the fact that $\Delta T^m_{i,j,k} > \Delta T^{m+1}_{i,j,k}$, which corresponds to a greater decrease in objective function per unit change of $V^m_{i,j,k}$. In other words, if $\beta^m_{i,j,k} < 1$ and $\beta^{m+1}_{i,j,k} > 0$, then we can always find a solution with a lower cost by doing the following: 1) adding γ to $\beta^m_{i,j,k}$ so that $\beta^m_{i,j,k} = 1$ and 2) adjusting $\beta^{m+1}_{i,j,k}$ with $\beta^{m+1}_{i,j,k} - \gamma$. Thus, we see in our new reduced ILP formulation that the extra $\beta_{i,j,k}$ variables are not constrained to be integers and that the only integer variables that we need are the $M^{x,y,k}_{i,j,k}(n)$ variables. Note that this assumption is no longer valid if we try to minimize the maximum temperature of the tiles since $T_{i,j,k}$ is no longer present in the objective function. There is no constraint on the $\beta^m_{i,j,k}$ values, so they cannot be relaxed as continuous variables. Thus, the resulting ILP will have an extremely large number of integer variables.

REFERENCES

[1] A. Fan, A. Rahman, and R. Reif, "Copper wafer bonding," *Electrochem. Solid-State Lett.*, vol. 2, no. 10, pp. 534–536, Oct. 1999.

[2] J. Cong and Y. Zhang, "Thermal-driven multilevel routing for 3-D ICs," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2005, pp. 121–126.

[3] J. Cong and Y. Zhang, "Thermal via planning for 3-D ICs," in *Proc. IEEE Int. Conf. Comput.-Aided Des.*, 2005, pp. 745–752.

[4] T. Zhang, Y. Zhan, and S. S. Sapatnekar, "Temperature-aware routing in 3D ICs," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2006, pp. 309–314.

[5] B. Goplen and S. Sapatnekar, "Placement of thermal vias in 3-D ICs using various thermal objectives," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 4, pp. 692–709, Apr. 2006.

[6] 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 South Pacific Des. Autom. Conf.*, 2008, pp. 209–212.

[7] H. Yu, J. Ho, and L. He, "Simultaneous power and thermal integrity driven via stapling in 3D ICs," in *Proc. IEEE Int. Conf. Comput.-Aided Des.*, 2006, pp. 802–808.

[8] S. Das, A. Chandrakasan, and R. Reif, "Design tools for 3-D integrated circuits," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2003, pp. 53–56.

[9] J. Minz and S. K. Lim, "Block-level 3D global routing with an application to 3D packaging," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2248–2257, Oct. 2006.

[10] M. Burnstein and R. Pelavin, "Hierarchical wire routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-2, no. 4, pp. 223–234, Oct. 1983.

[11] V. Pavlidis and E. Friedman, "Interconnect delay minimization through interlayer via placement in 3-D ICs," in *Proc. Great Lakes Symp. VLSI*, 2005, pp. 20–25.

[12] A. Ajami, K. Banerjee, and M. Pedram, "Effects of non-uniform substrate temperature on the clock signal integrity in high performance designs," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2001, pp. 233–236.

[13] T.-Y. Wang and C. C.-P. Chen, "3-D thermal-ADI: A linear-time chip level transient thermal simulator," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1434–1445, Dec. 2002.

[14] K. Boese, A. Kahng, B. McCoyy, and G. Robins, "Near-optimal critical sink routing tree constructions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 14, no. 12, pp. 1417–1436, Dec. 1995.

[15] J. Cong, K.-S. Leung, and D. Zhou, "Performance driven interconnect design based on distributed *RC* delay model," in *Proc. ACM Des. Autom. Conf.*, 1993, pp. 606–611.

[16] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *Proc. IEEE Int. Conf. Comput.-Aided Des.*, 2003, pp. 86–89.

Mohit Pathak (S'05) received the B.Tech degree in computer science and engineering from the Indian Institute of Technology (IIT), Kharagpur, India, in 2004. He is currently working toward the Ph.D. degree in the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, where he is also a Graduate Research Assistant.

He was with Magma Design Automation, India, for a year, where he was a Member of the Technical Staff. His areas of interests include physical design automation and VLSI design.

Sung Kyu Lim (S'94–M'00–SM'05) received the B.S., M.S., and Ph.D. degrees from the Computer Science Department, University of California, Los Angeles (UCLA), in 1994, 1997, and 2000, respectively.

From 2000 to 2001, he was a Postdoctoral Scholar at UCLA and a Senior Engineer with Aplus Design Technologies, Inc. He joined the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, in 2001, where he is currently an Associate Professor. His research focus is on the physical design automation for 3-D ICs, 3-D system-in-packages, microarchitectural physical planning, and field-programmable analog arrays. He is the author of *Practical Problems in VLSI Physical Design Automation* (Springer, 2008).

Dr. Lim received the Design Automation Conference Graduate Scholarship in 2003 and the National Science Foundation Faculty Early Career Development Award in 2006. He was a member of the Advisory Board of the Association for Computing Machinery (ACM)/Special Interest Group on Design Automation (SIGDA) during 2003–2008 and received the ACM/SIGDA Distinguished Service Award in 2008. He is currently an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS and was a Guest Editor for the *ACM Transactions on Design Automation of Electronic Systems*. He has served the Technical Program Committee of several ACM and IEEE conferences on electronic design automation.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 7, 2009 at 11:07 from IEEE Xplore. Restrictions apply.