Chapter 2
Steiner Routing for 3D IC

Abstract In this chapter, we study a performance and thermal-aware Steiner routing algorithm for 3D stacked ICs. 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 3D 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 performance constraint. We employ a novel scheme to relax the initial NLP formulation to ILP and consider all TSV from all nets simultaneously. Our tree construction algorithm outperforms the popular 3D-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 (i) the wirelength and delay distribution of various types of 3D interconnects, (ii) the impact of TSV RC parasitics on routing and TSV relocation, and (iii) the impact of various bonding styles on routing and TSV relocation. Lastly, we provide results on two-die stacking.

The materials presented in this chapter are based on [13].

2.1 Introduction

Technology feature sizes continue to shrink to meet performance demands on integrated circuits. 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 semi-global wires become increasingly expensive and clock frequencies become higher and higher, designers seek new

S.K. Lim, *Design for High Performance, Low Power, and Reliable 3D Integrated Circuits*, DOI 10.1007/978-1-4419-9542-1_2,

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 (3D) integration.

In a 3D integrated circuit, 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 [8], discrete wafers are "glued" together using vertical copper interconnects, permits multiple wafers and multiple third-dimension interconnects, overcoming the above limitations. 3D 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. 3D integration also addresses the wire delay problem by enabling the replacement of long and slow global interconnects with short and fast vertical routes.

The conventional 3D 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-Via (TSV) to establish interconnect among the dies. With this approach, the saving on the total amount of wiring, delay, and its power consumption easily outnumbers 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 (SOC) implementation, TSV-based 3D 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 micron dimensions, thereby contributing little to the area, delay, and power consumption of the overall system.

One of the major concerns of 3D 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 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 hotspot. Thus, several existing works utilize TSVs to lower on-chip temperature of 3D ICs [4, 5, 10, 11, 16, 17].

In this chapter, we formulate and solve the new *Performance and Thermal-aware 3D IC Steiner Routing* problem for multi-pin net routing in 3D stacked ICs. We emphasize that this problem is different from the conventional *2D Steiner Routing with Multiple Routing Layers*. The main reason is that the pins in 3D ICs are located in *multiple* device layers, whereas the pins in the 2D ICs are located in a single device layer as shown in Fig. 2.1. The following specific topics are covered in this chapter:

- We build efficient algorithms to construct Steiner trees for 3D ICs. Unlike the existing works on 3D Steiner tree construction [4, 7, 12, 17] that focus on

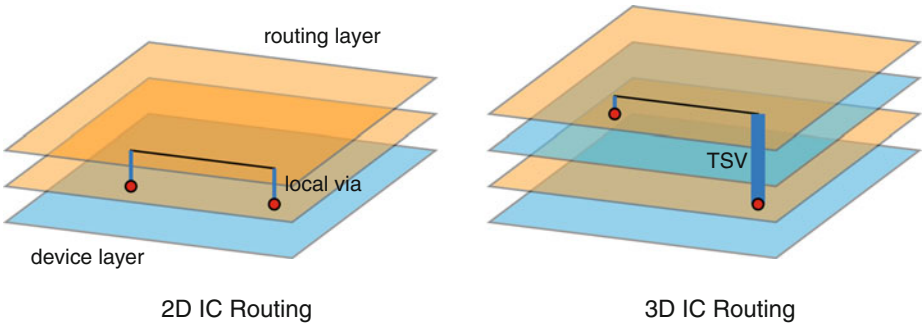

Fig. 2.1 Difference between 2D and 3D IC routing. In 3D IC, pins are located in multiple device layers, whereas in 2D IC, all pins lie in the same device layer

wirelength and thermal optimization, our tree construction algorithm optimizes thermal-aware performance. We believe that performance still remains as an important design metric, and thermal-aware delay model is important in Steiner tree construction. Unlike the existing works that decompose a 3D interconnect into a set of 2D 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.

- We formulate and solve the new *TSV Relocation* problem for thermal optimization in 3D 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 does not require valuable routing resource nor rip-up-and-reroute. We employ a novel scheme to relax the initial NLP formulation to ILP and consider all TSV from all nets simultaneously.
- We study related experimental results. Our tree construction algorithm outperforms the popular 3D-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.

2.2 Existing Works

The history of routing algorithm development for 3D ICs is relatively short. The authors of [7] presented a set of standard cell-based physical design tools for 3D ICs. Their 3D global routing algorithm is based on a 3D extension of [3], 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.

The authors of [4] presented a global routing algorithm for 3D 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 3D grid, they first build a 3D 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 multi-level routing framework. They later presented a follow-up work [5], where the TSV insertion and refinement is performed based on non-linear programming formulation.

The authors of [17] performed thermal-aware global routing for 3D ICs while utilizing "thermal vias" and "thermal wires". The authors first decompose each multi-pin net in the given netlist into a set of two-pin nets based on minimum spanning tree (MST) construction. Next, routing congestion map is obtained based on L/Z shape topology assumption. The location of TSVs are then determined for all inter-die two-pin nets based on the congestion map. The authors complete the routing for all nets by performing 2D 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 the authors of [4] and [17] 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 does not require valuable routing resource nor rip-up-and-reroute. In addition, our routing trees are built under performance and thermal constraints.[2]

The authors of [14] presented an analytical delay model for inter-die 3D interconnects. Their delay model is a function of TSV location/height as well as the related wires. Using this model, they determine the delay-optimal TSV location along a 3D interconnect that connects gates in different dies. However, this model is based on two-pin connections, and they do not perform routing. The authors of [12] perform block-level global routing for 3D System-In-Package, 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

[1] We provide comparison between our 3D router to this so called 3D maze router in the experimental section.

[2] In addition to the thermal via insertion during routing, the authors of [10] insert thermal vias after placement, while the authors of [11] try to redistribute whitespace in a given floorplan to allocate space for thermal vias. The authors of [16] insert both dummy thermal TSVs and power/ground TSVs simultaneously to reduce thermal and power supply noise.

layer to complete routing. The goal is to minimize wirelength, layer, congestion, and crosstalk. They formulate and design heuristics for 3D pin redistribution, net distribution, and channel assignment problems.

2.3 Preliminaries

2.3.1 Problem Formulation

We assume 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 as the driver, (2) a 3D routing grid G that represents the routing resource in a given 3D 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 3D thermal grid Z with thermal resistance on all edges and power consumption on all nodes.[3] A *3D Steiner Tree* is defined to be a set of 2D (= planar) Steiner trees connected by TSVs.

The goal of the *Performance and Thermal-aware 3D Steiner Routing* problem is to generate a 3D Steiner tree for each net while satisfying the capacity constraints specified in the underlying G.[4] The objective is to minimize (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 chapter uses the temperature dependent interconnect delay model presented in [1]. 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 co-efficient 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:

- *Single-die-two-pin (SD2P) nets*: a net in this group connects two pins that are located in the same die.
- *Single-die-multi-pin (SDMP) nets*: a net in this group connects more than two pins that are located in the same die.

[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.

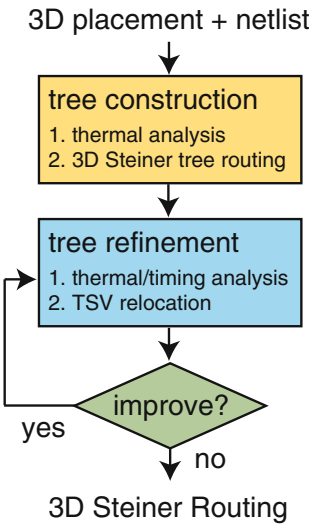

Fig. 2.2 Overview of our performance and thermal-aware 3D Steiner routing

- *Multi-die-two-pin (MD2P) nets*: a net in this group connects two pins that are located in two different dies.
- *Multi-die-multi-pin (MDMP) nets*: a net in this group connects more than two pins that are located in multiple different dies.

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

2.3.2 *Overview of the Approach*

Optimizing performance and thermal objectives simultaneously during 3D 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.

Figure 2.2 shows an overview of our performance and thermal-aware 3D Steiner routing process. Given a netlist and its 3D 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 non-uniform 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 hotspots. We then minimize the 3D on-chip temperature by relocating TSVs used in each tree under the given timing constraints. The goal is to move TSVs closer to the hotspots 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 temperature values to reflect the relocation at every iteration. Lastly, we repeat the whole tree refinement phase until no more thermal improvement is possible.

2.4 3D Steiner Tree Construction

2.4.1 Overview of the Algorithm

The basic approach of our 3D Steiner tree construction algorithm is similar to SERT [2], 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: (1) all the pins in SERT are located in the same die, whereas our 3D algorithm handles the pins located in multiple dies. This 3D case requires the usage of TSVs, and the location of these TSVs has huge impact on the topology of the tree as well as the sink pin delay. (2) The delay optimization in SERT is based on single variable, whereas our algorithm deals with two-variable function optimization, (3) our interconnect delay is computed based on the given thermal profile.

A pseudocode of our algorithm is shown in Algorithm 3. Our routing algorithm consists of two phases: construction (line 1–14) and rip-up-and-reroute (15–16). We construct 3D 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 3D 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 (line 5–6) 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

[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 3D thermal grid model for our temperature analysis [15].

Algorithm: 3D Steiner Tree Construction
input : netlist NL, routing graph G, thermal profile Z
output: 3D Steiner tree for each net
for *each net* $n \in NL$ **do**
  $T_n = p_0(n)$;
  Q_n = set of pins of n except p_0;
  while $Q_n \neq \emptyset$ **do**
    for *each pin* $a \in Q_n$ **do**
      for *each edge* $e \in T_n$ **do**
        x = connection point for $a \rightarrow e$;
        y = TSV location on $e(x,a)$;
        update $dly(p)$ for all $p \in T_n \cup a$;
        $X(a,e) = \max\{dly(p)\}$;
      end
    end
    (a_{min}, e_{min}) = pin+edge pair with min X;
    $T_n = T_n \cup e_{min}$;
    remove a_{min} from Q_n;
  end
  update Z periodically;
end
for *each non-timing critical* T_n *violating capacity* **do**
  rip-up-and-reroute T_n under Z;
end

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

delay impact is calculated based on the increase in temperature-dependent Elmore delay among all pins currently in T_n (line 9–10), where $dly(p)$ is the Elmore delay at pin p. This requires the computation of connection point x and TSV location y (line 7–8) (to be discussed in Sect. 2.4.2). Next, we select the pin-edge pair that results in the minimum max-delay increase (line 11) and add the pin to T_n (line 12–13). 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 non-timing critical nets, i.e., the nets with smaller max-delay values (line 15–16). 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 nearby hotspots. 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 multi-pin net, we first decompose the net into a set of two-pin nets based on their MST (Minimum Spanning Tree). We then route each two-pin sub-net using our maze router described above.

2.4.2 Computing Connection Point and TSV Location

For a given multi-pin 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 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 die are denoted C_{via} and R_{via}.

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 2D case has been presented in [2], 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 3D case.

Referring to Fig. 2.3, $e(p,c)$ and $e(q,b)$ are 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 the distance between node p and node x, 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.

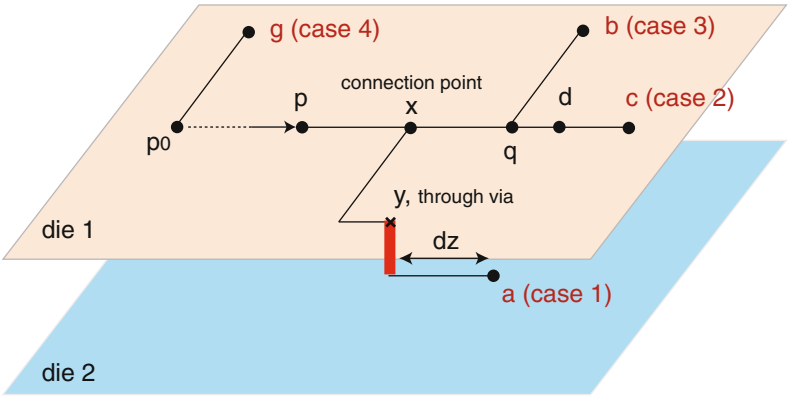

Fig. 2.3 Illustration of how pin a connects to $e(p,c) \in T$. $e(y,a)$ is routed in the *bottom die*, where as all other edges are routed in the *top die*. x is the location of connection point on $e(p,c)$. y is the location of the TSV inserted on $e(x,a)$. $e(q,b)$ is another branch in T. g is another 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

Figure 2.3 illustrates these four cases:

2.4.2.1 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 (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,

$$f_1 = K_0 + K_1\{c_1\delta y + C_{via} + c_2\delta z + c_1\delta c + C_c + C_b + c_1(\delta b - \delta q)\}$$

$$f_2 = r_1\delta x\left(c_1\frac{\delta x}{2} + c_1\delta y + C_{via} + c_2\delta z + c_1(\delta c - \delta x) + C_c\right)$$

$$f_3 = \begin{cases} r_1\delta x(c_1(\delta b - \delta q) + C_b), \text{ if } \delta x \le \delta q \\ r_1\delta q(c_1(\delta b - \delta q) + C_b), \text{ if } \delta x \ge \delta q \end{cases}$$

$$f_4 = r_1\delta y\left(c_1\frac{\delta y}{2} + C_{via} + c_2\delta z\right) + R_{via}\left(\frac{C_{via}}{2} + c_2\delta z\right) + r_2c_2\frac{\delta z^2}{2}$$

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

2.4.2.2 Case 2

The new delay at node c is given by

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

where

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

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

2.4.2.3 Case 3

The new delay at node b is given by

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

where

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

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.

2.4.2.4 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

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

These cases identify the four possible ways by which 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 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 given thermal distribution.[6]

2.4.3 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 depends upon the values of $\frac{\partial^2 F}{\partial \delta x^2}$ and the determinant of the Hessian matrix H_1:

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

where F is the delay function under consideration. The above 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 A: If $\frac{\partial^2 F}{\partial \delta x^2} \leq 0$ 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 B: If $\frac{\partial^2 F}{\partial \delta x^2} \leq 0$ $\frac{\partial^2 F}{\partial \delta y^2} \leq 0$ and $H_1 = 0$, we have a concave function, and the minimum lies on the boundary points.
- Case C: If $\frac{\partial^2 F}{\partial \delta x^2} = 0$, $\frac{\partial^2 F}{\partial \delta y^2} = 0$, 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 D: 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 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 above. Thus, there is only a fixed number of points (x, y) for which the Elmore delay values are minimized. Details are included in appendix section "Optimization of Two-Variable Delay Equations".

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

2.5 3D Tree Refinement with TSV Relocation

2.5.1 Overview of the Algorithm

The motivation behind our TSV relocation is to move as many TSVs into thermal hotspots as possible while preserving the original tree topology we obtain during our construction step. The TSVs in hotspots reduce the thermal resistance in these areas and establish heat conducting paths to the heat sink. The objective is to remove hotspots 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 [5]. Thus, large number of TSVs will degrade the yield and reliability of the 3D chip. This is the drawback of the existing works that add *additional* dummy TSV to reduce temperature [4, 5, 10, 11, 16, 17]. 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 additional TSVs needed may be kept at minimum.[7]

In general, thermal optimization with TSVs is non-linear 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 $R \propto \frac{1}{a}$, where a is the number of TSVs. In addition, general solutions available for solving non-linear problems cannot be applied directly to large size problems. In this chapter, we study a novel solution that helps us effectively overcome the non-linear nature of this problem. We study a relaxed ILP based formulation in which the number of integer variables are 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 hotspots simultaneously instead of iteratively targeting one by one. Our experimental results in Sect. 2.6 demonstrate the advantage of this approach.

2.5.2 Movable Range

We start our TSV relocation phase with the set of 3D Steiner trees we obtain from the construction step. All pins in each tree T_i are associated with 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.

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

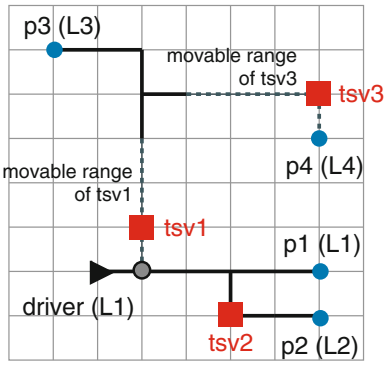

Fig. 2.4 Illustration of movable range. This figure is a *top-down* view of a multi-die stack. The driver is represented by a *triangle*, the sinks are represented by *dots*, and the TSVs are represented by *square boxes*. The driver is located in die 1 (= L1). The movable range of each TSV is represented by the *dotted lines*

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. 2.4. In case the movable range of a TSV v is a single point, v is non-movable; 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 optimization, where the movable TSVs are moved into thermal hotspots under timing constraint to reduce the thermal resistivity.

2.5.3 *Compact Thermal Analysis*

Figure 2.5 shows the fast thermal model used in our TSV relocation method, which we adopted from [5]. In this model each heat source is considered as a current source, and the temperature as voltage level. The 3D 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. 2.5. 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 3D ICs, the heat sinks are attached to the bottom or top side of the 3D 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. 2.5 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 the thermal conductivity of epoxy material used for bonding is much lower than that

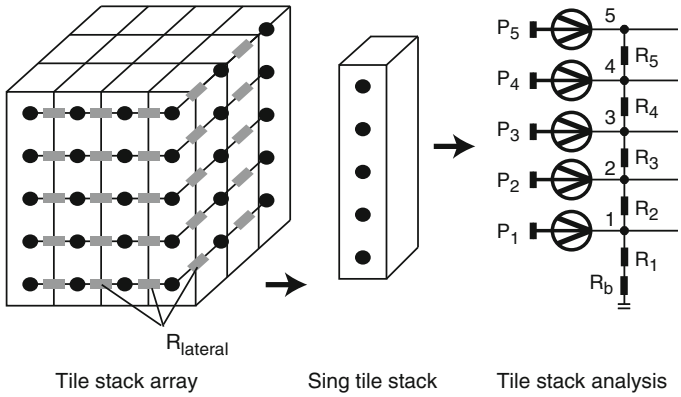

Fig. 2.5 Thermal model used, where R_b denotes the thermal resistance to the heat sink. The convention used in our ILP formulation is that adding TSVs at point (= tile) i reduces the value of R_i

of silicon itself. This essentially means that it is difficult for heat to dissipate in the vertical direction as compared with horizontal direction. To accurately verify the temperature reduction, a full (= vertical and lateral) resistive thermal model [15] (considering lateral resistances) is run twice, 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 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.

2.5.4 Non-linear Programming Formulation

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

The NLP based formulation is defined as follows (Table 2.1 explains the notations we use in the formulation):

Table 2.1 Variables and constants used in our NLP/ILP formulations

$T_{i,j,k}$	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)

Minimize

$$\sum_{(i,j,k)\in Z} \alpha_{i,j,k} \cdot T_{i,j,k} \tag{2.1}$$

Subject to:

$$T_{i,j,k} = T_{i,j,k-1} + (P_{i,j,K} + \cdots + P_{i,j,k}) \times R^{opt}_{i,j,k} \tag{2.2}$$

$$R^{opt}_{i,j,k} = \frac{\alpha}{\alpha/R^{no}_{i,j,k} + V^{opt}_{i,j,k}} \tag{2.3}$$

$$V^{opt}_{i,j,k} = V^{org}_{i,j,k} + \Delta V_{i,j,k} \tag{2.4}$$

$$\Delta V_{i,j,k} = \sum_{n\in N^{in}_{i,j,k}} M^{i,j,k}_{x,y,k}(n) - \sum_{n\in N^{out}_{i,j,k}} M^{y,w,k}_{i,j,k}(n) \tag{2.5}$$

$$G^{cur}_{i,j,k} \leq G^{max}_{i,j,k}, \ \forall (i,j,k) \tag{2.6}$$

$$G^{cur}_{i,j,k} = G^{org}_{i,j,k} + \sum_{n \in N} M^{v,w,k}_{x,y,k}(n) \cdot \Delta G_{i,j,k}, (i,j,k) \text{ on path } (x,y,k) \rightarrow (v,w,k) \tag{2.7}$$

$$\sum_{n \in N_{mov}} M^{x,y,k}_{i,j,k}(n) = 1 \tag{2.8}$$

$$M^{x,y,k}_{i,j,k}(n) \in \{0,1\} \tag{2.9}$$

Equation (2.1) 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 $\alpha_{i,j,k}$ is, the lower the $T_{i,j,k}$ we desire.[9]

Equation (2.2) gives the temperature at each tile based on our fast thermal model illustrated in Fig. 2.5, where K is the maximum height of our thermal tile (= total number of dies in the 3D stack). Equation (2.3) 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:

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

Equation (2.4) is the definition of $V^{opt}_{i,j,k}$. Equation (2.5) 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 (2.6) ensures that the routing resource (= wires and TSVs) capacity constraints are satisfied.

Equation (2.7) 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 $G^{org}_{i,j,k}$ denote the original usage at tile (i,j,k) before the move. Then, the new usage, denote by $G^{cur}_{i,j,k}$, 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

[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 Explanation of Eq. (2.15), 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 hotspots are given a preference.

negative values based on whether the corresponding via move increases or decreases the routing capacity at the given location. Lastly, the whole process is done for all nets that contain relocated TSVs.

Equation (2.8) 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 (2.9) states that $M_{i,j,k}^{x,y,k}(n)$ are binary integer variables.

We note that this original TSV relocation problem is non-linear due to the inverse relation between thermal resistance and number of TSVs in a tile ($= R_{i,j,k}^{opt}$ *vs* $V_{i,j,k}^{opt}$) in Eq. (2.3). In the next section, we study a simplified integer linear programming formulation that overcomes this non-linear problem formulation.

2.5.5 *Integer Linear Programming 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 in the following way: we replace Eqs. (2.2) and (2.3) with the following:

$$T_{i,j,k} = T_{i,j,k-1} + \gamma_{i,j,k}^{0} \times \delta_{i,j,k}^{0} + \cdots + \gamma_{i,j,k}^{vmax} \times \delta_{i,j,k}^{vmax} \tag{2.10}$$

$$1 \cdot \gamma_{i,j,k}^{1} + 2 \cdot \gamma_{i,j,k}^{2} + \cdots + vmax \cdot \gamma_{i,j,k}^{vmax} = V_{i,j,k}^{opt} \tag{2.11}$$

$$\sum_{m=0}^{vmax} \gamma_{i,j,k}^{m} = 1, \ \forall (i,j,k) \tag{2.12}$$

$$\gamma_{i,j,k}^{m} \in \{0,1\} \tag{2.13}$$

Equation (2.10) is a new way to calculate the temperature at each tile (refer to Table 2.1 for the definition of the related variables and constants). In this equation, $\gamma_{i,j,k}^{m}$ are the new integer variables, whereas $\delta_{i,j,k}$ are the constants that are calculated for each possible value of the number of TSVs in a tile. Equation (2.11) equates the $\gamma_{i,j,k}^{m}$ variable with the optimum number of TSVs in a tile. Compared with the non-linear Eq. (2.3), this Eq. (2.11) shows linear relationship between $V_{i,j,k}^{opt}$ vs. $\gamma_{i,j,k}$. Equation (2.12) ensures that for each tile, only one $\gamma_{i,j,k}^{m}$ takes a value 1. Lastly, Eq. (2.13) ensures that $\gamma_{i,j,k}^{m}$ is either 0 or 1. All other equations in our NLP formulation remain same in our ILP formulation.

The number of new integer variables $\gamma_{i,j,k}^{m}$ is proportional to the number of tiles in our thermal grid plus the number of TSVs movable in each grid. Adding such

a large number of integer variables makes the problem harder to solve. In our next section, we study 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.

2.5.6 *Fast Integer Linear Programming Formulation*

Our fast ILP-based TSV relocation is formulated as follows (Table 2.1 explains the notations we use in the formulation):

Minimize

$$\sum_{(i,j,k)\in Z} \alpha_{i,j,k} \cdot T_{i,j,k} \tag{2.14}$$

Subject to:

$$T_{i,j,k} = T_{i,j,k-1} + \delta^0_{i,j,k} - \beta^1_{i,j,k} \cdot (\delta^0_{i,j,k} - \delta^1_{i,j,k}) - \cdots - \beta^{vmax}_{i,j,k} \cdot (\delta^{vmax-1}_{i,j,k} - \delta^{vmax}_{i,j,k}) \tag{2.15}$$

$$\beta^1_{i,j,k} + \beta^2_{i,j,k} + \cdots + \beta^{vmax}_{i,j,k} = V^{opt}_{i,j,k} \tag{2.16}$$

$$V^{opt}_{i,j,k} = V^{org}_{i,j,k} + \Delta V_{i,j,k} \tag{2.17}$$

$$\Delta V_{i,j,k} = \sum_{n\in N^{in}_{i,j,k}} M^{i,j,k}_{x,y,k}(n) - \sum_{n\in N^{out}_{i,j,k}} M^{v,w,k}_{i,j,k}(n) \tag{2.18}$$

$$G^{cur}_{i,j,k} \le G^{max}_{i,j,k},\ \forall (i,j,k) \tag{2.19}$$

$$G^{cur}_{i,j,k} = G^{org}_{i,j,k} + \sum_{n\in N} M^{v,w,k}_{x,y,k}(n) \cdot \Delta G_{i,j,k}, (i,j,k) \text{ on path } (x,y,k) \to (v,w,k) \tag{2.20}$$

$$0 \le \beta^m_{i,j,k} \le 1 \tag{2.21}$$

$$M^{x,y,k}_{i,j,k}(n) \in \{0,1\} \tag{2.22}$$

$$\sum_{n\in N_{mov}} M^{x,y,k}_{i,j,k}(n) = 1 \tag{2.23}$$

Equation (2.15) is a new way to compute the temperature at tile (i,j,k), which is different from Eq. (2.10). A detailed explanation of this equation is included in appendix section "Explanation of Eq. (2.15)". Equation (2.16) states that the total

number of TSVs in a tile (i,j,k), which is specified by the $\beta^i_{i,j,k}$ values $(1 \leq i \leq vmax)$, should be equal to $V^{opt}_{i,j,k}$. Equation (2.21) restricts the range of values $\beta^n_{i,j,k}$ can take. All other equations are the same as discussed in NLP formulation.

A few points are worth mentioning. First, in order to overcome the restriction of moving just one TSV per net (= Eq. (2.23)), 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 temperature is minimal. Our related experiment shown in Table 2.5 suggests that the 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 to zero. Table 2.6 shows the impact of this relaxation on the solution quality and runtime.

2.6 Experimental Results

2.6.1 Experimental Setting

We implemented our router named 3D 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, 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 runtime in seconds for each circuit. We obtained 3D placement using an algorithm that is similar to [9]. The following are the details of our experimental setting:

1. We use four-die stacking for our 3D IC, where the top two and bottom two dies are bonded in face-to-face and the middle two in back-to-back unless specified otherwise.
2. We assume all four dies have different unit-length resistance and capacitance values [14] as follows unless specified otherwise: $r_1 = 86\,\Omega/\text{mm}$ and $c_1 = 396\,\text{fF/mm}$, $r_2 = 175\,\Omega/\text{mm}$ and $c_2 = 100\,\text{fF/mm}$, $r_3 = 74\,\Omega/\text{mm}$ and $c_3 = 279\,\text{fF/mm}$, and $r_4 = 154\,\Omega/\text{mm}$ and $c_4 = 120\,\text{fF/mm}$.
3. The dimension of our face-to-face TSVs is $1 \times 1 \times 10\,\mu\text{m}$. The parasitics are $R_{via} = 17.2\,\Omega/\text{mm}$ and $C_{via} = 371.8\,\text{fF/mm}$. The dimension of our back-to-back TSVs is $10 \times 10 \times 40\,\mu\text{m}$. The parasitics are $R_{via} = 0.172\,\Omega/\text{mm}$ and $C_{via} = 1943.8\,\text{fF/mm}$.

Table 2.2 3D routing grid dimensions and edge capacities for four die stack

				XYZ capacities		
ckt	#nets	v-bound	Grid dim	X & Y	Z (F2F)	Z (B2B)
s9234	5,844	5,563	20×20	8	3	1
b14_opt	5,646	5,546	40×40	8	3	2
s13207	8,727	8,298	40×40	10	4	2
s15850	10,397	9,915	40×40	10	5	4
b20_opt	12,501	12,174	40×40	16	6	4
b21_opt	12,678	12,288	60×60	16	6	4
b22_opt	18,086	17,911	60×60	20	7	5
ibm09	52,989	53,483	80×80	60	20	10
ibm10	68,004	67,651	100×100	75	20	10
ibm11	70,028	70,524	120×120	85	25	12
ibm13	84,191	82,989	140×140	95	25	12
ibm17	184,227	180,001	140×140	300	40	20

4. The "v-bound" column in Table 2.2 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.
5. The routing grid dimensions used for four die stack are shown in Table 2.2. 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 re-routed after initial tree generation.
6. The thermal grid dimension is $20 \times 20 \times 4$ for four-die stacked 3D 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 $10^7\,\mathrm{W/m^2}$ to account for the gate-level switching activity factor. We use our compact thermal model discussed in Sect. 2.5.3 for TSV relocation and [15] for all other purposes.

2.6.2 Tree Construction Results

We implement two existing 3D routers for comparison. The first is 3D maze router by [4] discussed in Sect. 2.2. The second is 3D A-tree router, where we extend the original 2D version [6] into 3D. More specifically, we first converted the 3D problem to a 2D problem by mapping the pin locations to the 2D plane. Then we perform 2D A-tree routing [6]. Lastly, our 2D solution is translated back to 3D, where the connection to the pins not located on the same die as the driver are connected using TSVs.

Table 2.3 shows a comparison between 3D maze [4], 3D A-tree [6], and our 3D Elmore routing. Our baseline is 3D maze router. We observe that our 3D Elmore

Table 2.3 Comparison between 3D maze [4], 3D A-tree [6], and our 3D Elmore routers. The "v-bound" column shows the lower bound on TSV count

	3D maze router [4]				3D A-tree router [6]				3D elmore router			
ckt	Wire	Delay	TSV	cpu	Wire	Delay	TSV	cpu	Wire	Delay	TSV	cpu
s9234	0.167	0.072	5,700	8	0.169	0.047	5,734	1	0.18	0.023	5,613	2
b14_opt	0.19	0.086	7,366	10	0.21	0.071	7,818	5	0.23	0.072	6,937	5
s13207	0.39	0.091	9,707	15	0.41	0.077	10,342	7	0.43	0.064	9,595	9
s15850	0.49	0.11	11,763	46	0.51	0.11	12,924	10	0.56	0.092	11,516	15
b20_opt	0.947	0.31	18,423	108	1.04	0.25	21,730	46	1.19	0.22	18,703	57
b21_opt	0.97	0.29	18,627	128	1.06	0.138	21,688	44	1.2	0.145	19,314	58
b22_opt	2.15	0.48	27,116	186	2.33	0.32	31,255	87	2.64	0.24	29,090	104
ibm09	29.4	222.2	104,481	639	33.01	124.7	122,465	245	36.9	103.6	118,013	353
ibm10	51.4	600.9	136,071	1,263	57.4	337.5	157,983	349	61.9	273.3	153,483	601
ibm11	53.3	457.9	131,815	1,859	60.5	264.6	152,422	456	61.2	218.4	141,922	711
ibm13	83.5	953.3	167,311	4,000	90.7	509.4	195,459	1,113	94.7	438.7	183,813	1,411
ibm17	157.4	1,723.4	398,145	7,754	169.8	1,124.6	408,861	2,122	172.4	899.8	405,400	2,456
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 2.4 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

	SD2P		SDMP		MD2P		MDMP	
ckt	dly	Wire	dly	Wire	dly	Wire	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

router achieves 52% average delay improvement over 3D maze routing and 11% improvement over 3D A-tree. 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 3D Elmore router and 3D A-tree, while 3D maze uses 6% less TSVs. In case of wirelength, 3D maze and 3D A-tree obtained comparable results, but our 3D Elmore router uses 15% higher wirelength. Lastly, our 3D Elmore router runs 3× faster than 3D maze router and about 40% slower than 3D A-tree router.

We see that the number of TSVs needed by the 3D maze or 3D Elmore router is about twice as many as the minimum required. The number of TSVs used in 3D A-tree algorithm is the highest.[10] We observe from circuit `b21_opt` that 3D A-tree performs better than our 3D Elmore in terms of performance. We observe that this was due to congestion that caused larger number of nets to be ripped and re-routed in our 3D Elmore algorithm. In some cases, we observe that 3D A-tree caused lower congestion thus required less re-routing.

2.6.3 Delay and Wirelength Distribution

Our first goal is to collect the wirelength and delay statistics among the four types of nets in 3D Steiner routing mentioned in Sect. 2.3.1: single-die-two-pin (SD2P), single-die-multi-pin (SDMP), multi-die-two-pin (MD2P), and multi-die-multi-pin (MDMP) nets. Table 2.4 shows the statistics, where we report the average max-sink

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

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 SD2P). This is reasonable since they contain multiple pins in multiple dies and thus require multiple TSVs. We also observe that multi-pin nets incur larger delay and wirelength compared with two-pin nets (SDMP vs. SD2P and MDMP vs. MD2P). We also observe that MD2P nets have 8% smaller delay compared with SD2P on average. This is primarily due to the benefit of 3D connection, where TSV-based 3D connections tend to have smaller delay.

2.6.4 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 hotspots in an iterative fashion: we choose a single hotspot and relocate movable TSVs into it. We then repeat this process for the next hotspot until no more temperature improvement can be obtained. In addition, we developed two TSV relocation methods based on our ILP formulation introduced in Sect. 2.5.6: single ILP and multiple ILP. Under the single ILP method, we perform our ILP-based TSV relocation once. 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* hotspots simultaneously.

Table 2.5 shows the maximum temperature, average temperature and standard deviation obtained by the greedy method and our ILP-based methods (single iteration vs. multiple 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 reduction with our ILP-based methods, whereas the greedy method improves the max/ave 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 K nets is around 3,045 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 2.5 we also show the impact of multiple iterations on our fast ILP. We observe that the temperature saving between single vs. multiple ILP is comparable for maximum temperature. In 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 2.6 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 case of the two bigger circuits, we gave our slow ILP one full day and have it report the best solution discovered so far. We observe that our slow ILP obtained 6% better results on the maximum temperature, but the runtime

Table 2.5 TSV relocation results. T_{max}, T_{ave}, T_{std} respectively denotes the maximum temperature, average temperature, and standard deviation among all thermal tiles. The runtime is in seconds

	Initial temp			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	84.5	45.1	20.7	67	83.1	40.1	18.3	246	5
b14_opt	114.5	46.1	32.3	114.1	45.8	32.2	2	107.6	38.7	27.8	89	105.8	33.6	25.1	315	6
s13207	112.1	66.2	27.6	111.3	66.1	27.6	2	101.2	53.2	24.3	82	101.1	52.8	24.0	196	3
s15850	115.1	44.3	34.3	114.0	44.0	34.1	2	103.1	37.5	27.6	175	102.6	35.3	26.9	324	3
b20_opt	108.3	38.9	37.3	108.1	38.8	37.0	3	98.7	30.3	32.3	209	96.4	27.6	29.8	813	7
b21_opt	114.1	45.7	24.2	113.5	45.5	24.0	3	109.4	37.8	20.1	206	108.4	35.7	19.4	612	3
b22_opt	114.5	54.6	18.4	114.2	54.6	18.4	4	105.1	49.6	15.2	228	104.5	47.8	14.7	388	2
ibm09	94.2	47.8	23.1	93.8	47.3	22.9	11	87.1	43.6	21.2	702	84.8	39.5	18.7	2,113	6
ibm10	113.4	54.2	21.0	112.9	54.0	20.9	13	105.6	45.2	16.4	452	104.0	32.3	18.3	1,566	5
ibm11	108.4	45.3	27.8	107.7	45.1	27.7	19	99.7	35.2	21.0	832	97.9	32.3	18.3	4,211	8
ibm13	95.1	52.4	27.9	94.5	52.1	27.7	24	86.6	43.8	22.3	1,387	86.1	41.8	21.8	2,346	2
ibm17	111.2	48.6	19.2	110.8	48.2	19.0	38	101.5	41.7	15.4	3,045	101.0	39.6	14.1	6,836	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 2.6 Impact of M-variable relaxation (fast-ILP vs 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	–

Table 2.7 Various TSV sizes and their parasitic resistance (Ω/mm) and capacitance (fF/mm). Dimensions are in µm

	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	1,229.1
Size II	10	10	40	0.172	1,943.8
Size III	20	20	60	0.043	2,798.2
	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	1,519.1
Size III	15	15	55	0.076	2,604.3

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.

2.6.5 *Impact of TSV Dimension and Parasitics*

Next, we study 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 as well as the underlying thermal profile. Table 2.7 shows three different TSV sizes and their RC parasitics we use in our experiment. Size II is the default set that is used in all of the previous experiments as discussed in Sect. 2.6.1. Table 2.8 shows our 3D 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 detrimental impact on MDMP (multi-die-multi-pin) nets, which are more likely to become critical nets. Another factor is the higher parasitic capacitance values for larger TSVs. Since Elmore delay model penalizes heavily

Table 2.8 Impact of TSV dimension on delay, wirelength, and TSV counts. We use the three TSV sizes shown in Table 2.7

	Size I			Size II			Size III		
ckt	Delay	Wire	TSV	Delay	Wire	TSV	Delay	Wire	TSV
s9234	0.02	0.171	5,694	0.023	0.18	5,613	0.032	0.183	5,558
b14_opt	0.048	0.213	7,062	0.072	0.23	6,937	0.122	0.217	6,911
s13207	0.047	0.43	9,749	0.064	0.43	9,595	0.077	0.44	9,657
s15850	0.066	0.53	11,703	0.092	0.56	11,516	0.163	0.57	11,351
b20_opt	0.16	1.18	19,274	0.22	1.19	18,703	0.31	1.21	18,187
b21_opt	0.125	1.2	19,818	0.145	1.2	19,314	0.228	1.24	18,810
b22_opt	0.229	2.61	29,818	0.24	2.64	29,090	0.48	2.69	28,847
ibm09	96.0	36.6	119,125	103.6	36.9	118,013	127.8	37.3	116,724
ibm10	252.1	60.6	155,349	273.3	61.9	153,483	390.2	65.7	151,519
ibm11	198.6	60.2	142,990	218.4	61.2	141,922	268.7	67.6	139,202
ibm13	419.5	92.5	184,428	438.7	94.7	183,813	563.4	109.5	181,456
ibm17	824.8	171.1	416,458	899.8	172.4	405,400	987.6	187.4	401,345
RATIO	1.0	1.0	1.0	1.07	1.01	0.98	1.34	1.1	0.97

Table 2.9 Impact of TSV dimension on TSV relocation and the maximum temperature reduction. We use the three TSV sizes shown in Table 2.7. T_{ini} and T_{max} respectively denote the maximum temperature before and after TSV relocation

		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

on 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, especially for short interconnects, to minimize the overall delay. However, the TSV count reduction is only 3%, indicating that our 3D Elmore router still makes a heavy use of TSVs (up to 400 K for `ibm17`).

We also conducted experiments to observe the impact of TSV dimension and parasitics on temperature saving obtained by our TSV relocation algorithm. The results are shown in Table 2.9. We observe that larger TSVs result in more temperature saving (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.

Table 2.10 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	2,195	0.019	0.155	2,402	0.015	0.161	2,113
b14_opt	0.078	0.194	3,680	0.18	0.042	4,459	0.039	0.219	3,914
s13207	0.074	0.388	3,497	0.07	0.41	4,262	0.07	0.43	3,676
s15850	0.103	0.504	4,298	0.093	0.53	5,561	0.083	0.57	4,680
b20_opt	0.354	0.975	8,210	0.228	1.06	12,207	0.186	1.15	10,084
b21_opt	0.38	0.966	7,977	0.163	1.05	11,367	0.156	1.13	9,834
b22_opt	0.555	2.18	11,408	0.466	2.38	16,824	0.281	2.71	15,185
ibm09	228.1	30.3	44,667	135.7	33.9	65,692	116.1	35.8	59,955
ibm10	578.7	52.3	49,855	355.2	58.5	77,334	312.1	59.1	74,329
ibm11	409.6	52.6	67,076	254.1	59.7	75,281	215.5	60.2	61,353
ibm13	906.1	81.5	72,546	491.9	87.5	98,964	451.7	90.3	79,688
ibm17	2,234.6	174.5	160,435	1,224.5	187.8	192,349	978.8	206.9	178,013
RATIO	1.0	1.0	1.0	0.56	1.09	1.30	0.47	1.15	1.14

2.6.6 *Impact of Bonding Style*

So far the top two and bottom two dies are bonded in face-to-face, and the middle two in back-to-back in our four-die stack as discussed in Sect. 2.6.1. We now study the impact of bonding style on 3D routing results. In our new four-die stack, all dies are bonded face-to-back. We use $R_{via} = 0.267\,\Omega$/mm and $C_{via} = 1{,}519.1$ fF/mm for the face-to-back TSVs. One important difference between the "F2F+B2B" stack and "F2B-only" stack 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 the sizes. Therefore, we re-ran our 3D placer to obtain a new placement that minimizes inter-die connections for all face-to-back bonding. The routing results are shown in Table 2.10. We first observe that the TSV count is considerably lower compared with Table 2.3. 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 2.10 are larger than those in Table 2.3. 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 because the number of inter-die connection is minimized, resulting in less opportunity for wirelength reduction. Lastly, both Tables 2.3 and 2.10 show the same trend in terms of delay, wirelength, and TSV count among 3D maze, 3D A-tree, and 3D Elmore routers.

2.6.7 *Two-die versus four-die Stacking*

So far our 3D stack contained four dies as discussed in Sect. 2.6.1. We now use two-die stack in our experiment, where the two dies are bonded face-to-face. Our wire and TSV parasitics are as follows: $r_1 = 86\,\Omega$/mm, $c_1 = 396$ fF/mm, $r_2 = 175\,\Omega$/mm,

$c_2 = 100\,\text{fF/mm}$, $R_{via} = 17.2\,\Omega/\text{mm}$, and $C_{via} = 371.8\,\text{fF/mm}$. Table 2.11 shows a comparison between 3D maze, 3D A-tree, and our 3D Elmore router. Our baseline is 3D maze router. We observe that our 3D Elmore router achieves 54% average delay improvement over 3D maze routing and 11% improvement over 3D A-tree. This significant delay reduction came at the cost of 13% wirelength and 10% TSV count increase compared with 3D maze router. This trend is almost the same as what we saw in Table 2.3 for the four-die stack case.

TSV relocation results are shown in Table 2.12. We again observe a similar trend as in Table 2.5: 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 two-die and four-die stack is twofold. First, the temperature reduction is smaller with two-die (6%) compared with four-die (9%). This is because there are fewer hotspots and fewer TSVs in two-die stack. Second, the ILP runtime is also smaller in two-die stack due to the smaller problem size.

2.7 Conclusions

We studied two new problems that are important for 3D stacked IC technology: 3D Steiner tree construction and TSV relocation. Our routing algorithm is based on a constructive method, where a 3D 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 the TSVs under given thermal profile. For TSV relocation, we devised an innovative technique which helps avoid the non-linear optimization required for temperature optimization. Our formulation can handle large number of TSVs simultaneously for an effective temperature optimization.

Appendix

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 Sect. 2.4.3 allow the computation of x (= connection point) and y (= TSV location) as follows:

- For $d(a)$ we have

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

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

$$H_1 = -(r_1 c_2)^2 \{(a_0 + b_0 - 2) 2 b_0\}$$

Table 2.11 Two-die stack results: comparison between 3D maze [4], 3D A-tree [6], and our 3D Elmore routers. The "v-bound" column shows the lower bound on TSV count

	3D maze routing				3D A-tree				3D elmore routing			
ckt	Delay	Wire	TSV	cpu	Delay	Wire	TSV	cpu	Delay	Wire	TSV	cpu
s9234	0.04	0.3	5,137	13	0.04	0.33	5,250	2	0.017	0.3	4,976	4
b14_opt	0.083	0.38	6,102	12	0.04	0.42	6,560	4	0.04	0.42	6,056	6
s13207	0.13	0.71	7,960	15	0.11	0.75	8,338	4	0.99	0.79	8,216	7
s15850	0.17	0.92	9,616	26	0.82	0.98	10,049	7	0.068	1.04	9,890	10
b20_opt	0.41	1.76	14,205	35	0.17	1.96	15,724	12	0.15	2.1	15,445	15
b21_opt	0.35	1.75	14,253	35	0.16	1.96	16,004	12	0.19	2.07	15,650	16
b22_opt	0.99	4.1	20,557	86	0.38	4.54	22,545	40	0.34	4.8	23,571	56
ibm09	973.5	58.5	79,677	411	461.9	66.7	89,134	234	388.7	68.7	93,184	287
ibm10	2,245.8	98.2	109,685	530	1,156.6	106.7	119,065	241	1,023.5	113.4	128,630	284
ibm11	1,617.6	102.4	100,321	771	804.8	116.1	112,061	303	657.1	125.2	113,573	344
ibm13	2,578.6	174.5	139,355	987	1,667.6	186.6	140,774	331	1,473.9	195.6	142,872	413
ibm17	6,208.3	338.9	338,854	1,526	3,177.2	357.8	359,394	411	2,790.1	375.3	370,599	791
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 2.12 Two-die stack results: TSV relocation results. T_{ini} and T_{max} respectively denote the maximum temperature before and after TSV relocation

		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	1,356.7	2
RATIO	1.0	0.989	1.0	0.943	17.22	0.936	36.86	–

Thus, we see that when $H_1 = 0$, $\frac{\partial^2 F}{\partial \delta x^2} \leq 0$ and $\frac{\partial^2 F}{\partial \delta y^2} = 0$, the optimal delay is found at points according to the Case B. If $H_1 < 0$, optimal delay is found at points according to the Case D. If $H_1 > 0$, we have $\frac{\partial^2 F}{\partial \delta x^2} \leq 0$, so the optimal delay is found at points according to the Case A.

- For $d(b)$ we need to evaluate two cases: (1) when $x \geq b$, we have $\frac{\partial^2 F}{\partial \delta x^2} = 0$, $\frac{\partial^2 F}{\partial \delta y^2} = 0$, and $H_1 = 0$. Thus, the optimal delay is found at points according to the Case C. (2) when $x \leq b$, we have $\frac{\partial^2 F}{\partial \delta x^2} = -2r_1c_2$, $\frac{\partial^2 F}{\partial \delta y^2} = 0$, 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 the Case B. Otherwise, they are found at points according to the Case D.
- For $d(c)$ we have $\frac{\partial^2 F}{\partial \delta x^2} = -2r_1c_2$, $\frac{\partial^2 F}{\partial \delta y^2} = 0$, and $H_1 = -(r_1c_2)^2(b_0 - 1)^2$. If $H_1 = 0$, the optimal delay is found at points according to the Case B. Otherwise, they are found at points according to the Case D.
- For all other nodes not in T_p, we have $\frac{\partial^2 F}{\partial \delta x^2} = 0$, $\frac{\partial^2 F}{\partial \delta y^2} = 0$, 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 the Case C.

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.

Explanation of Eq. (2.15)

From Fig. 2.5 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 2.1 for definition) as follows:

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

We see that $\delta^m_{i,j,k} \propto R^m_{i,j,k} \propto 1/V^m_{i,j,k}$, thus $\delta^m_{i,j,k}$ 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

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

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 the 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 non-integer values for the variable $\beta^m_{i,j,k}$ (refer to Table 2.1 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 non-zero 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 (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 variables $\beta_{i,j,k}$ are not constrained to be integers and that the only integer variables 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. Ajami, K. Banerjee, M. Pedram, Effects of non-uniform substrate temperature on the clock signal integrity in high performance designs, in *Proceedings of IEEE Custom Integrated Circuits Conference*, San Diego, 2001, pp. 233–236
2. K. Boese, A. Kahng, B. McCoyy, G. Robins, Near-optimal critical sink routing tree constructions, in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **14**(12), 1417–1436 (1995)
3. M. Burnstein, R. Pelavin, Hierarchical wire routing, in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **2**(4), 223–234 (1983)
4. J. Cong, Y. Zhang, Thermal-driven multilevel routing for 3-D ICs, in *Proceedings of Asia and South Pacific Design Automation Conference*, Shanghai, 2005, pp. 121–126

5. J. Cong, Y. Zhang, Thermal via planning for 3-D ICs, in *Proceedings of IEEE International Conference on Computer-Aided Design*, San Jose, 2005
6. J. Cong, K.-S. Leung, D. Zhou, Performance driven interconnect design based on distributed RC delay model, in *Proceedings of ACM Design Automation Conference*, Dallas, 1993
7. S. Das, A. Chandrakasan, R. Reif, Design tools for 3-D integrated circuits, in *Proceedings of Asia and South Pacific Design Automation Conference*, Kitakyushu, 2003, pp. 53–56
8. A. Fan, A. Rahman, R. Reif, Copper wafer bonding. Electrochem. Solid-State Lett. **2**, 534–536 (1999)
9. B. Goplen, S. Sapatnekar, Efficient thermal placement of standard cells in 3D ICs using a force directed approach, in *Proceedings of IEEE International Conference on Computer-Aided Design*, San Jose, 2003
10. B. Goplen, S. Sapatnekar, Placement of thermal vias in 3-D ICs using various thermal objectives. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **25**(4), 692–709 (2006)
11. X. Li, Y. Ma, X. Hong, S. Dong, J. Cong, LP based white space redistribution for thermal via planning and performance optimization in 3D ICs, in *Proceedings of Asia and South Pacific Design Automation Conference*, Seoul, 2008, pp. 209–212
12. J. Minz, S.K. Lim, Block-level 3D global routing with an application to 3D packaging. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **25**(10), 2248–2257 (2006)
13. M. Pathak, S.K. Lim, Thermal-aware steiner routing for 3D stacked ICs, in *Proceedings of IEEE International Conference on Computer-Aided Design*, San Jose, 2007
14. V. Pavlidis, E. Friedman, Interconnect delay minimization through interlayer via placement in 3-D ICs, in *Proceedings of Great Lakes Symposum on VLSI*, Chicago, 2005
15. T.-Y. Wang, C.C.-P. Chen, 3-D thermal-ADI: a linear-time chip level transient thermal simulator, in *IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems*, **21**(12), 1434–1445 (2002)
16. H. Yu, J. Ho, L. He, Simultaneous power and thermal integrity driven via stapling in 3D ICs, in *Proceedings of IEEE International Conference on Computer-Aided Design*, San Jose, 2006
17. T. Zhang, Y. Zhan, S.S. Sapatnekar, Temperature-aware routing in 3D ICs, in *Proceedings of Asia and South Pacific Design Automation Conference*, Yokohama, 2006, pp. 309–314

http://www.springer.com/978-1-4419-9541-4

Design for High Performance, Low Power, and Reliable
3D Integrated Circuits
Lim, S.K.
2013, XXVIII, 560 p., Hardcover
ISBN: 978-1-4419-9541-4