Construction of All Multilayer Monolithic RSMTs and Its Application to Monolithic 3D IC Routing

MONZURUL ISLAM DEWAN, Washington State University
SHENG-EN DAVID LIN, Cadence Design Systems Inc.
DAE HYUN KIM, Washington State University

Monolithic three-dimensional (M3D) integration allows ultra-thin silicon tier stacking in a single package. The high-density stacking is acquiring interest and is becoming more popular for smaller footprint areas, shorter wirelength, higher performance, and lower power consumption than the conventional planar fabrication technologies. The physical design of M3D integrated circuits requires several design steps, such as three-dimensional (3D) placement, 3D clock-tree synthesis, 3D routing, and 3D optimization. Among these, 3D routing is significantly time consuming due to countless routing blockages. Therefore, 3D routers proposed in the literature insert monolithic interlayer vias (MIVs) and perform tier-by-tier routing in two substeps. In this article, we propose an algorithm to build a routing topology database (DB) used to construct all multilayer monolithic rectilinear Steiner minimum trees on the 3D Hanan grid. To demonstrate the effectiveness of the DB in various applications, we use the DB to construct timing-driven 3D routing topologies and perform congestion-aware global routing on 3D designs. We anticipate that the algorithm and the DB will help 3D routers reduce the runtime of the MIV insertion step and improve the quality of the 3D routing.

CCS Concepts: • **Hardware → 3D integrated circuits**; **Wire routing**; Partitioning and floorplanning; Placement;

Additional Key Words and Phrases: Monolithic three-dimensional integration, monolithic inter-layer vias, multilayer monolithic rectilinear Steiner minimum trees, 3D position sequence, tier sequence, 3D potentially optimal Steiner tree, 3D rectilinear Steiner minimum tree

ACM Reference format:
Monzurul Islam Dewan, Sheng-En David Lin, and Dae Hyun Kim. 2023. Construction of All Multilayer Monolithic RSMTs and Its Application to Monolithic 3D IC Routing. *ACM Trans. Des. Autom. Electron. Syst.* 29, 1, Article 17 (December 2023), 28 pages.
https://doi.org/10.1145/3626958

This work was supported by the Defense Advanced Research Projects Agency Young Faculty Award under Grant D16AP00119 and the New Faculty Seed Grant (125679-002) funded by the Washington State University.
Authors' addresses: M. I. Dewan and D. H. Kim, School of Electrical Engineering and Computer Science, Washington State University, 355 NE Spokane Street, Pullman, WA 99164; e-mails: {monzurulislam.dewan, daehyun.kim}@wsu.edu; S.-E. D. Lin, Cadence Design Systems Inc., 12301 Research Boulevard, Building V, Suite 200, Austin, TX 78759; e-mail: slin3@eecs.wsu.edu.

1084-4309/2023/12-ART17 $15.00
https://doi.org/10.1145/3626958

1 INTRODUCTION

Monolithic Three-Dimensional (M3D) integration stacks very thin silicon tiers and electrically connects transistors in different tiers by **Monolithic Interlayer Vias (MIVs)**. Unlike through-silicon vias, MIVs are tiny in width, shorter in vertical length (z-directional), and presumed to have insignificant parasitic **Resistance and Capacitance (RC)**. The dimensions of MIVs are even smaller than those of top-level interlayer vias and comparable with those of lower-level interlayer vias (even smaller than 100 nm in diameter). As a result, the use of MIVs is considered to have almost negligible area and capacitance overheads in the M3D **Integrated Circuit (IC)** layout design. Moreover, M3D ICs are favorable choices in terms of power, performance, and area for advanced technology nodes and future transistor architectures over their through-silicon-via-based counterparts [22]. In addition, designing vertical processors using M3D showed power, performance, and thermal efficiency [3]. However, profusely inserting MIVs into an M3D IC layout increases routing congestion since planar wires are connected to the MIVs, and routing of the planar wires requires much larger area than the MIV area. Therefore, M3D IC layout design generally tries to minimize the number of MIVs inserted into a layout [1, 11, 20], which necessitates the **Three-Dimensional (3D)** routing to reduce the number of MIVs and routing congestion.

Algorithms for 3D routing could route both **Two-Dimensional (2D)** and 3D nets of a design separately or concurrently.[1] For example, the routing methodology used in previous work [4] routes 3D nets first and routes 2D nets after that. On the contrary, Panth et al. [19] route 2D and 3D nets simultaneously by using the modified library files into a commercial tool. However, the latter has some drawbacks compared to the former. According to our routing simulations using modified library files, the runtime of the simultaneous routing of 2D and 3D nets increases significantly as the complexity (the average net degree, the net counts, the number of tiers, the number of instances, and most importantly the number of routing blockages representing MIVs) of a design goes up. On the contrary, if the 3D nets are routed first, 2D nets can be routed separately in each tier (routing tier by tier or simultaneously but separately in each tier). Thus, the 3D-net-first routing methodology has been used extensively in the literature [4, 6, 20, 21].

Figure 1 illustrates the 3D-net-first routing methodology that finds MIV locations for each 3D net first, then inserts MIVs into the locations and decomposes the 3D net into multiple 2D nets, and finally routes all the 2D nets separately in each tier. In Figure 1(a), eight pins are spread out in two tiers with one 3D net connecting all four a pins and two 2D nets connecting two b and two c pins. In Figure 1(b), a 3D routing topology using two z-directional edges is constructed for the 3D net. The z-directional edges are replaced by MIVs, and the 3D net is decomposed into three 2D nets, n_1, n_2, and n_3 in Figure 1(c). Decomposing the 3D net into three 2D nets and the MIV locations, the 3D net routing is converted into the routing of three 2D nets, two in the bottom tier with one in the top tier. Finally, the 2D nets are routed in each tier in Figure 1(d).

As mentioned earlier, 3D routing should minimize the number of MIVs used to route 3D nets and evenly distribute the MIVs as well as the planar wires over the entire layout area for routing congestion minimization. The MIV insertion methodologies used in the literature, however, do not control the MIV count, MIV locations, and planar wires of 3D nets effectively. For example, the 3D **Rectilinear Steiner Tree (RST)** algorithms used in other works [5, 6, 20, 21] do not guarantee the minimization of the MIV count. The MIV insertion algorithm used in previous work [4] minimizes the MIV count but fails to minimize the planar wirelength. **Multilayer Obstacle-Avoiding Rectilinear Steiner Tree (MLOARST)** construction algorithms can minimize both the MIV count and the planar wirelength [8, 14]. However, they do not generate multiple routing topologies that have different MIV locations and planar wire distributions.

[1] A 2D (3D) net is a net connecting instances placed in a single tier (different tiers).

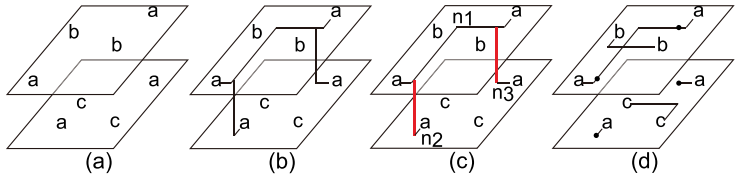

Fig. 1. 3D-net-first routing. (a) Three nets to route. (b) 3D routing topology generation for the 3D net. (c) MIV insertion. (d) Tier-by-tier routing.

In our preliminary work [10], we proposed an algorithm to build a routing topology **Database (DB)** for the construction of **All Multilayer Monolithic Rectilinear Steiner Minimum Trees (AMM-RSMTs)** on the 3D Hanan grid for a set of pin locations for up to six-pin nets and four tiers. **Multilayer Monolithic Rectilinear Steiner Minimum Trees (MMRSMTs)** have the shortest planar wirelength with the minimum number of MIVs, so MIV insertion algorithms can use the DB to effectively optimize MIV locations and planar wires of 3D nets. We also proposed DB size reduction techniques for practical use of the DB in previous work [10]. This article is an extension of that previous work [10]. In this article, we include the algorithm to construct all the 3D **Potentially Optimal Steiner Trees (POSTs)** and show the generation and techniques to reduce the size of the DB from our previous work [10]. We present two applications using the DB: timing-driven 3D routing topology generation and congestion-aware 3D global routing. We also propose a 3D optimal net-breaking technique for the congestion-aware 3D global routing. Our core contributions in this article are listed as follows:

- We apply the AMM-RSMT DB from the previous work to construct timing-driven MMRSMTs considering several objectives and compare the outcomes with a FLUTE-like **Brute-Force (BF)** approach.
- In addition, we apply the **AMM-RSMT DB (ARD)** to congestion-aware global routing of 3D designs aiming to minimize the total overflow of both planar and 3D global routing edges.
- We propose a hybrid 3D net-breaking technique for the higher-degree nets and introduce an optimal 3D net-breaking technique as a part of that for the congestion-aware 3D global routing.
- We also present **Position Sequence (PS)** algebra to aid readers in applying congruent rules to generate 2D and 3D POSTs at the very end.

The rest of this article is organized as follows. In Section 2, we discuss the terminologies used in this work, review the **Rectilinear Steiner Minimum Tree (RSMT)** construction of FLUTE and the necessity of generating POSTs in the 2D Hanan grid, and discuss the concept of MMRSMT. In Sections 3 and 4, we present the algorithm to construct the ARD and show the outcomes obtained from the construction of all the 3D POSTs in the 3D Hanan grid, and details of the DB generation and size reduction from previous work [10]. Sections 5 and 6 demonstrate the applications of our ARD to timing-driven 3D routing topology generation and congestion-aware global routing of 3D designs, respectively, and compare a FLUTE-like BF approach with ours showing the detailed results for several two-, three-, and four-tier 3D designs. Finally, we summarize and conclude in Section 7.

2 PRELIMINARIES

In this section, we explain terminologies used in this article, review two papers on the construction of RSMTs [2, 9], and formulate the problem we solve in this work.

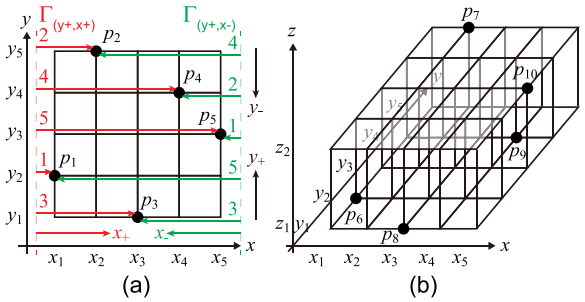

Fig. 2. 2D and 3D Hanan grids.

2.1 Terminologies

2.1.1 2D Hanan Grid. Suppose a finite set S of points is given in the 2D plane. Let $X_S = \{x_1, \ldots, x_L\}$ $(x_1 \leq \cdots \leq x_L)$ and $Y_S = \{y_1, \ldots, y_M\}$ $(y_1 \leq \cdots \leq y_M)$ be the sets of the x- and y-coordinates of the points in S, respectively. Then, the *2D Hanan grid* constructed for S is a graph $G_S = (V_S, E_S)$, where V_S and E_S are defined as follows:

$$V_S = \{(x, y) | x \in X_S, y \in Y_S\}, E_S = E_{S,X} \cup E_{S,Y},$$
$$E_{S,X} = \{(v_1, v_2) | v_1 = (x_i, y_j) \in V_S, v_2 = (x_{i+1}, y_j) \in V_S\},$$
$$E_{S,Y} = \{(v_1, v_2) | v_1 = (x_i, y_j) \in V_S, v_2 = (x_i, y_{j+1}) \in V_S\}.$$

Figure 2(a) shows the 2D Hanan grid constructed for the five points, $\{p_1, \ldots, p_5\}$.

2.1.2 3D Hanan Grid. Suppose a finite set T of points is given in the 3D space. Let $X_T = \{x_1, \ldots, x_L\}$ $(x_1 \leq \cdots \leq x_L)$, $Y_T = \{y_1, \ldots, y_M\}$ $(y_1 \leq \cdots \leq y_M)$, and $Z_T = \{z_1, \ldots, z_N\}$ $(z_1 \leq \cdots \leq z_N)$ be the sets of the x-, y-, and z-coordinates of the points in T, respectively. Then, the *3D Hanan grid* constructed for T is a graph $G_T = (V_T, E_T)$, where V_T and E_T are defined as follows:

$$V_T = \{(x, y, z) | x \in X_T, y \in Y_T, z \in Z_T\}, E_T = E_{T,X} \cup E_{T,Y} \cup E_{T,Z},$$
$$E_{T,X} = \{(v_1, v_2) | v_1 = (x_i, y_j, z_k) \in V_T, v_2 = (x_{i+1}, y_j, z_k) \in V_T\},$$
$$E_{T,Y} = \{(v_1, v_2) | v_1 = (x_i, y_j, z_k) \in V_T, v_2 = (x_i, y_{j+1}, z_k) \in V_T\},$$
$$E_{T,Z} = \{(v_1, v_2) | v_1 = (x_i, y_j, z_k) \in V_T, v_2 = (x_i, y_j, z_{k+1}) \in V_T\}.$$

Figure 2(b) shows the 3D Hanan grid constructed for the five points, $\{p_6, \ldots, p_{10}\}$.

2.1.3 Position Sequence. x_+- and x_--directions are the directions along which x-coordinates increase and decrease, respectively. $y_\pm$- and $z_\pm$-directions are defined similarly.

Suppose a finite set $P = \{p_1, \ldots, p_n\}$ of n distinct pins[2] is given. Let the x-coordinates of the y-directional edges of the 2D Hanan grid G_P be x_1 to x_n from the left and the y-coordinates of the x-directional edges of G_P be y_1 to y_n from the bottom as shown in Figure 2(a). Then, we denote sorting the pins in the increasing and decreasing order of their c-coordinates (c is x or y) by c_+ and c_-, respectively. In Figure 2(a), for example, y_+ sorting leads to the ordered list $L_1 = (p_3, p_1, p_5, p_4, p_2)$.

Suppose we obtain an ordered list $L = (l_1, \ldots, l_n)$ from c_+ or c_- sorting. Then, we can obtain the indices of the $\bar{c}$-coordinates (if c is x (or y), $\bar{c}$ is y (or x)) of the pins in the $\bar{c}_+$- or $\bar{c}_-$-direction from L. For example, we obtain (31542) and (35124) if we extract the x-coordinates of the pins in L_1 in the x_+ and x_- directions, respectively. The PS $\Gamma_{(s,r)}$ for P is a sequence $(k_1 k_2 \ldots k_n)$ where $s \in \{c_+, c_-\}$, $r \in \{\bar{c}_+, \bar{c}_-\}$, and k_i is the index of the $\bar{c}$-coordinate of the i-th pin in the r-direction

[2]If the coordinate of p_i is (x_{p_i}, y_{p_i}), $x_{p_i} \neq x_{p_j}$ and $y_{p_i} \neq y_{p_j}$ for any i and j $(i \neq j)$.

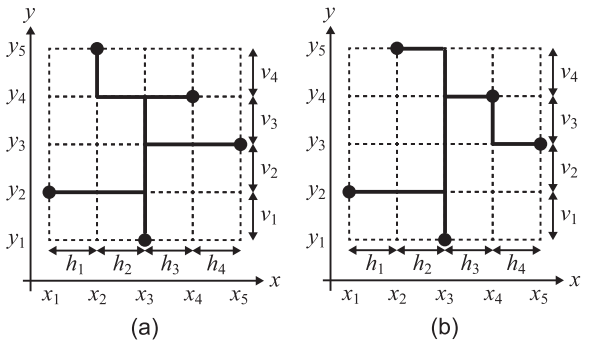

Fig. 3. Two RSTs constructed on the 2D Hanan grid.

in the list of the pins sorted by s sorting. For example, assume that (s, r) is (y_+, x_+) and P is the set of pins in Figure 2(a). Then, we first sort the pins along the y_+-direction, which leads to the ordered list $(p_3, p_1, p_5, p_4, p_2)$, and obtain $\Gamma_{(y_+,x_+)} = (31542)$. Similarly, $\Gamma_{(y_+,x_-)}$ is (35124), $\Gamma_{(y_-,x_+)}$ is (24513), $\Gamma_{(y_-,x_-)}$ is (42153), $\Gamma_{(x_+,y_+)}$ is (25143), $\Gamma_{(x_+,y_-)}$ is (41523), $\Gamma_{(x_-,y_+)}$ is (34152), and $\Gamma_{(x_-,y_-)}$ is (32514).

2.1.4 Potentially Optimal Wirelength Vector and POST. The wirelength of an RST on the 2D Hanan grid can be expressed as a linear combination of the x- and y-directional edge vectors representing the tree as explained in the work of Chu and Wong [2]. For example, the wirelength of the tree in Figure 3(a) is

$$L = 1 \cdot h_1 + 2 \cdot h_2 + 2 \cdot h_3 + 1 \cdot h_4 + 1 \cdot v_1 + 1 \cdot v_2 + 1 \cdot v_3 + 1 \cdot v_4, \tag{1}$$

which can also be expressed as $L = C \cdot E$, where $C = (1, 2, 2, 1, 1, 1, 1, 1)$ and $E = (h_1, h_2, h_3, h_4, v_1, v_2, v_3, v_4)$. C is called a *coefficient vector* and E is called an *edge length vector*. The edge length vector is a constant vector for given pins. However, the coefficient vector is dependent on the RST. For example, the coefficient vector for the tree in Figure 3(b) is $(1, 2, 1, 1, 1, 1, 2, 1)$. Thus, the two trees in Figure 3 have the same edge length vector but different coefficient vectors.

For given pin locations, a coefficient vector $C = (c_1, \ldots, c_k)$ becomes a **Potentially Optimal Wirelength Vector (POWV)** if it satisfies the following conditions [2]:

- There exists an RST that connects all the pins and uses the edges specified in the coefficient vector C on the Hanan grid constructed for the pins.
- There is no other coefficient vector $C' = (c'_1, \ldots, c'_k)$ satisfying $c'_i \leq c_i$ for all $i = 1, \ldots, k$.

An RST corresponding to a POWV is called a *potentially optimal Steiner tree (POST)* [2]. The two RSTs shown in Figure 3(a) and (b) are POSTs.

2.2 Construction of All RSMTs on the Hanan Grid

FLUTE constructs an RSMT by a lookup table [2]. The lookup table consists of all PSs, all POWVs belonging to each PS, and one POST for each POWV. Whenever a set of pin locations is given, FLUTE first finds the PS of the pin locations, compares the wirelengths of all the POWVs belonging to the PS, and returns the POST of the POWV having the minimum wirelength. If multiple POWVs have the same wirelength, FLUTE can return their POSTs. The returned POSTs are RSMTs for the pin locations. However, FLUTE finds only one POST for each POWV, although a POWV can have multiple POSTs. Thus, Lin generated a DB (called *ARSMT DB*) storing all POSTs for each POWV in previous work [9, 12]. Figure 4 shows an overview of the ARSMT DB. The algorithm finding all

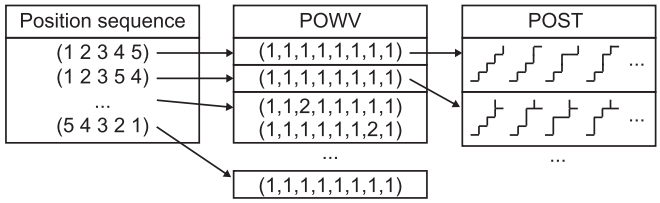

Fig. 4. An overview of the lookup table of all POSTs in other works [9, 12].

POSTs for each POWV uses a binary decision tree with several speedup techniques to reduce the DB construction time.

2.3 Multilayer Monolithic Rectilinear Steiner Minimum Trees

We first define three terminologies.

Definition 1. A *3D rectilinear tree* is a tree having only x-, y-, and z-directional edges and connecting all given pins.

Definition 2. A *3D rectilinear Steiner tree* (3D RST) is a 3D rectilinear tree with Steiner points. A Steiner point is a nonpin vertex with more than two edges.

Definition 3. A *3D rectilinear Steiner minimum tree* (3D RSMT) is a 3D RST having the minimum wirelength.

The wirelength of a 3D rectilinear tree is computed by the sum of the lengths of all the edges in the tree. In the M3D IC layout design, however, the area and capacitance overhead of an MIV is negligible, so we can set the length of an MIV (the length of a z-directional edge) to zero during 3D routing. However, minimizing the number of MIVs inserted in the layout is still crucial. Thus, we define an MMRSMT as follows.

Definition 4. A *multilayer monolithic rectilinear Steiner minimum tree* (MMRSMT) is a 3D RST satisfying the following:

- Its planar wirelength is equal to the wirelength of a 2D RSMT constructed for the pins projected onto the 2D plane.
- The number of z-directional edges is minimal.

If we project all the edges in an MMRSMT onto the xy plane, the projection becomes a 2D RSMT. Thus, an MMRSMT can be constructed from a 2D RSMT by properly placing the x- and y-directional edges of the 2D RSMT in a 3D grid and inserting z-directional edges. In addition, we obtain 2D RSMTs from POSTs as mentioned in the previous section. Thus, we can construct an MMRSMT from a POST. We define a 3D POST as follows.

Definition 5. Suppose a set of xy-distinct[3] pin locations is given. Let the set be $P = \{(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$. Let the set of the projections of the pins onto the xy plane be $P' = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Let a POST constructed for P' be $G' = (V', E')$. Let the coordinate of e' in E' be $e'(i, j)$. A *3D potentially optimal Steiner tree* (3D POST) is a tree T that connects all the pins in P, uses the minimum number of z-directional edges in the 3D Hanan grid $G = (V, E)$ constructed from P, and uses one of the edges among $e(i, j, k = 0, \ldots, t-1) \in E$ for each $e'(i, j) \in E'$. t in the definition is the number of tiers. From now on, we denote the POSTs in the ARSMT DB as 2D POSTs to distinguish them from 3D POSTs.

[3]If the coordinate of p_i is $(x_{p_i}, y_{p_i}, z_{p_i})$, $x_{p_i} \neq x_{p_j}$ and $y_{p_i} \neq y_{p_j}$ for any i and j $(i \neq j)$.

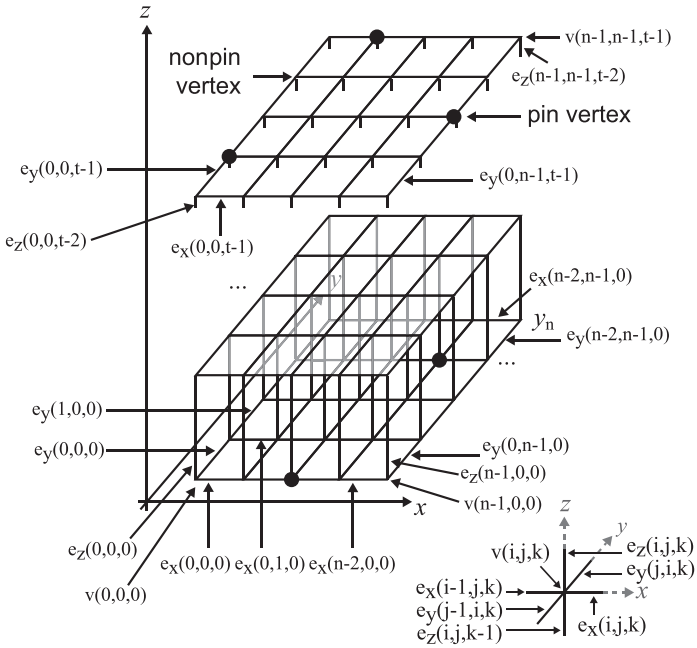

Fig. 5. An $n \times n \times t$ 3D grid, pin and nonpin vertices, and indices for x-, y-, and z-directional edges.

In summary, if we have a DB of all 3D POSTs, we can build all MMRSMTs for given pin locations quickly. In this work, therefore, we build a DB of all 3D POSTs for all possible relative pin locations in two, three, and four tiers. Note that in the case of pins without unique x- or y-coordinates, we can assume they have distinctive coordinates by slightly adjusting their locations to the left/right (for x-coordinates) or up/down (for y-coordinates). Based on that, we generate PSs from their relative positions knowing that the lengths of the newly evolved edges (tiny extensions) are zeros. This would eventually revert the distinct coordinates to their actual nondistinct coordinates.

3 CONSTRUCTION OF ALL 3D POSTS

In this section, we present an algorithm to construct all 3D POSTs on the 3D Hanan grid. Figure 5 shows a 3D grid, pin and nonpin vertices, x-, y-, and z-directional edges, and notations used in this article.

3.1 Construction of All 3D POSTs

The input to the algorithm is a set P of pin locations and a 2D POST, $G_2 = (V_2, E_2)$, constructed from the projection of the pins onto the xy plane. For example, Figure 6(a) shows three pins in a 3D grid, the projection of the pins, and a 2D POST for them.

Algorithm 1 shows the proposed algorithm for constructing all 3D POSTs. We first set the *visited* variables of all the edges in E_2 to false (line 1). Then, we sort the edges and store the result in an ordered set E_2' (line 2). The *sort_edges* function chooses a pin vertex in G_2 and performs the breadth-first search starting from the vertex until all the pin vertices are reached. Whenever it goes through an edge, the function inserts the edge into E_2'. This order reduces the runtime of the algorithm. For example, the *sort_edges* function starts from the pin vertex p_1 in Figure 6(a). Then, E_2' becomes $(e_x(0,0), e_y(0,1), e_x(1,1), e_y(1,1))$. Then, we construct a 3D grid $G_3 = (V_3, E_3)$ from G_2 and P (line 3). The *construct_3D_grid* function expands G_2 to G_3 as shown in Figure 6(b). Then,

ALGORITHM 1: Construction of all 3D POSTs for given 3D pin locations and 2D POST.

Function: Construct all 3D POSTs for P and G_2.
Input: Pin locations (P) and a 2D POST $G_2 = (V_2, E_2)$.
1: e.visited = false for all $e \in E_2$;
2: Ordered set E_2' = **sort_edges** (G_2);
3: $G_3 = (V_3, E_3)$ = **construct_3D_grid** (G_2, P);
4: e.used = false for all $e \in E_3$;
5: T = {}; nr_MIVs = 0; min_nr_MIVs = ∞;
6: Call **recur_con** (T, G_2, G_3, E_2', 0, nr_MIVs, min_nr_MIVs);
7: Return T;

Function: **recur_con** (T, G_2, G_3, E_2', index, nr_MIVs, min_nr_MIVs)
if index $\geq |E_2'|$ **then**
  if nr_MIVs == min_nr_MIVs **then**
    Add G_3 to T;
  else if nr_MIVs < min_nr_MIVs **then**
    Clear T; Add G_3 to T; min_nr_MIVs = nr_MIVs;
  end if
  return;
end if
$e = E_2'$[index];
e.visited = true;
for $s = 0$; $s < t$; $s = s + 1$ **do**
  $e_3 = E_3[e.x][e.y][s]$;
  e_3.used = true;
  $v = e$.left (or e.bottom);
  min_t1 = max_t1 = 0;
  if all the edges connected to v have been visited **then**
    min_t1, max_t1 = **get_min_max_tier** (v, G_2, G_3);
  end if
  $v = e$.right (or e.top);
  min_t2 = max_t2 = 0;
  if all the edges connected to v have been visited **then**
    min_t2, max_t2 = **get_min_max_tier** (v, G_2, G_3);
  end if
  delta = (max_t1 - min_t1) + (max_t2 - min_t2);
  nr_MIVs = nr_MIVs + delta;
  if nr_MIVs $\leq$ min_nr_MIVs **then**
    recur_con (T, G_2, G_3, E_2', index+1, nr_MIVs, min_nr_MIVs);
  end if
  nr_MIVs = nr_MIVs - delta;
  e_3.used = false;
end for
e.visited = false;

we set the *used* variables of all the edges in E_3 to false (line 4). T is a set of graphs storing all the 3D POSTs, nr_MIVs is a variable storing the number of MIVs used in G_3, and min_nr_MIVs is a variable storing the minimum number of MIVs used in the 3D POSTs (line 5). Then, we call the *recur_con* function to recursively construct all 3D POSTs (line 6). Once the algorithm ends, we return T (line 7).

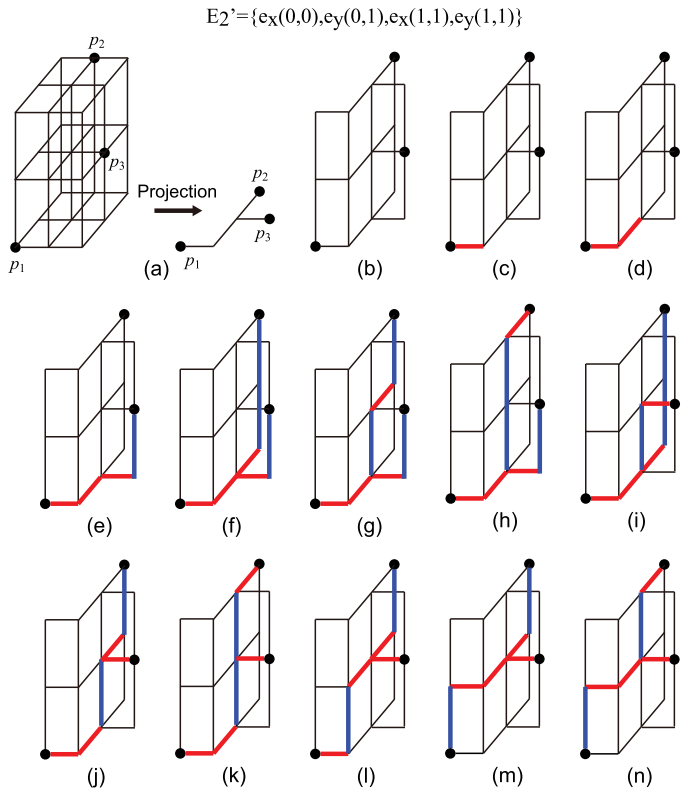

Fig. 6. Construction of all 3D POSTs in three tiers for pins $(0, 0, 0)$, $(1, 2, 2)$, $(2, 1, 1)$. (a) A 2D POST is given. (b) The construct_3D_grid function creates a 3D grid structure. (c)–(n) 3D POST construction. The red edges are used planar edges and the blue edges are used z-directional edges.

The *recur_con* function starts from checking the given index, which is used to access the edges in E'_2. The edge index is greater than the number of edges in E'_2 when there is no more edge to process in G_3 (line 1), which means that G_3 is a 3D graph connecting all the pins. In this case, if the total number of MIVs used in G_3 is equal to the minimum number of MIVs used in the best graphs found until now, we add it to T (line 3). However, if the total number of MIVs used in G_3 is less than the minimum number of MIVs used in the best graphs found until now, all the graphs in T use more MIVs than G_3, so we empty T, add G_3 to T, and update min_nr_MIVs (line 5).

If the edge index is less than the size of E'_2 (line 9), we visit the edge in E'_2 indexed by the edge index variable (line 10) and try using edges in G_3 corresponding to the indexed edge (lines 11–31). First, suppose $e'_d(i, j)$ is E'_2[index], where d is either x or y. Then, we try using $e_d(i, j, k)$ in G_3 for each $k = 0, \ldots, t - 1$ (line 13). In Figure 6(c), for example, we try using $e_x(0, 0, 0)$ in G_3. Then, if e is x-directional, we obtain its left vertex in G_2, and otherwise we obtain its bottom vertex in G_2 and assign it to v (line 14). Then, we find the bottommost and topmost tiers that should be connected along the z-axis through v in G_3 by the *get_min_max_tier* function (lines 15–18). The function finds all the visited edges connected to e in G_2, obtains the tiers of the edges in G_3 corresponding to the visited edges, and finds the bottommost and topmost tiers. In addition, if v is a pin vertex, its z-coordinate should be included in the computation of the range of the tiers. We repeat the same process for the right vertex of e (or the top vertex if e is y-directional) (lines 19–23).

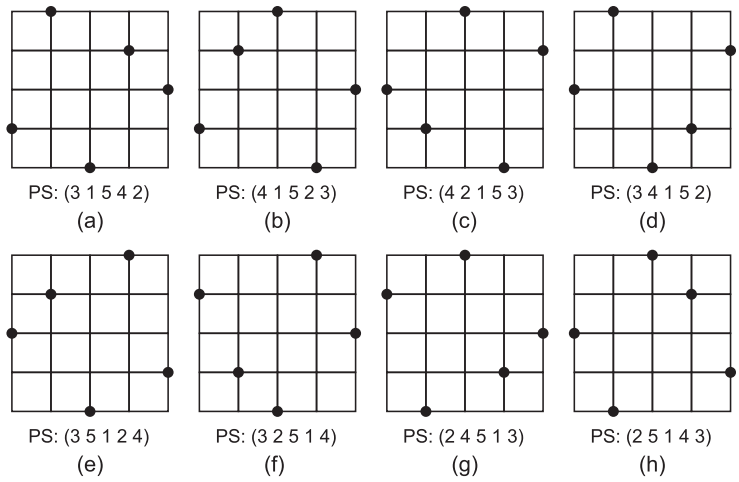

Fig. 7. Congruence of eight PSs.

If we have visited all the edges connected to the left and right (or the bottom and top) vertices of e, we can find the z-directional edges required to connect to the pin and the edges along the z-axis at the vertices. From the z-directional edges, we obtain the number of MIVs (line 24). If the total number of MIVs currently used in G_3 is less than or equal to the minimum number of MIVs used in the best graphs found until now, we move on to the next edge in E_2' (line 27). Otherwise, the current graph uses more MIVs than the best graphs, so we do not need to proceed to the next edge. Once the recursive function call ends (line 28), we re-adjust the number of MIVs used in G_3 (line 29) and try using the edge above e (lines 30 and 31).

In Figure 6(c), for example, $e_x(0, 0, 0)$ is in Tier 0 and its left vertex is a pin vertex, so both the bottommost and topmost tiers for the vertex are Tier 0. Then, we move on to $e_y(0, 1)$ in E_2' and try using $e_y(0, 1, 0)$ in G_3 in Figure 6(d). The *used* variables of all the edges connected to the bottom vertex of $e_y(0, 1)$ are *true* at this point and all the edges are placed in Tier 0. Thus, we do not need to add any z-directional edges above the vertex. Then, we process the next edge $e_x(1, 1)$ in Figure 6(e). The right vertex of $e_x(1, 1)$ is connected to the pin located at $(2, 1)$ in G_2, which corresponds to the pin located at $(2, 1, 1)$ in G_3, so the bottommost and topmost tiers at the vertex are Tier 0 and Tier 1, respectively. Thus, we use $e_z(2, 1, 0)$ in G_3, which is inserting an MIV into the location. When we also try using $e_y(1, 1, 0)$ in Figure 6(f), we finally construct a 3D graph connecting all the pins and the total number of MIVs is 3. Similarly, the total numbers of MIVs in the 3D graphs in Figure 6(g) through (i) are all 3. However, the 3D graph in Figure 6(j) uses two MIVs. At this time, T contains all the 3D graphs found in Figure 6(f) through (i), so we delete all of them from T and add the 3D graph found in Figure 6(j) to T. There are four more 3D graphs using two MIVs as shown in Figure 6(k) through (n). Thus, when the algorithm finishes, T contains all the five 3D graphs, which become 3D POSTs for the given pin locations and 2D POST.

3.2 Congruence of 3D POSTs

The runtime of the algorithm shown in Algorithm 1 is still long and there are numerous 3D POSTs in the DB, so it is crucial to reduce the runtime and the DB size. In this section, we show congruent properties of the 3D POSTs, which are used to skip generating and storing some 3D POSTs.

3.2.1 Congruence of PSs. As mentioned in the work of Chu and Wong [2], two PSs are congruent if rotating one of them leads to the other. For example, Figure 7(a) shows the PS (31542). If we

rotate it counterclockwise by 90, 180, and 270 degrees, we obtain PSs (41523), (42153), and (34152) as shown in Figure 7(b), (c), and (d), respectively. If two PSs are congruent to each other, the POSTs constructed for one of them can be used for the other. Thus, we do not need to generate 2D POSTs for some PSs. In addition to the rotation, reflection also generates congruent PSs. Reflecting the pin locations in Figure 7(a) over the y-directional line results in PS (35124) shown in Figure 7(e). Now, rotating the PS counterclockwise by 90, 180, and 270 degrees leads to PSs (32514), (24513), and (25143) shown in Figure 7(f), (g), and (h), respectively.

Rotating and reflecting a PS has the same effect as generating PSs by $\Gamma_{(s,r)}$. For example, generating the PS in Figure 7(a) is the same as generating the PS $\Gamma_{(y_+,x_+)}$. The PSs obtained by rotating the PS $\Gamma_{(y_+,x_+)}$ by 90, 180, and 270 degrees are the same as the PSs $\Gamma_{(x_+,y_-)}$, $\Gamma_{(y_-,x_-)}$, and $\Gamma_{(x_-,y_+)}$, respectively. Similarly, reflecting $\Gamma_{(y_+,x_+)}$ over the y-directional line is the same as generating the PS $\Gamma_{(y_+,x_-)}$. Then, rotating $\Gamma_{(y_+,x_-)}$ by 90, 180, and 270 degrees is the same as obtaining the PSs $\Gamma_{(x_-,y_-)}$, $\Gamma_{(y_-,x_+)}$, and $\Gamma_{(x_+,y_+)}$, respectively. Appendix A shows operations defined for PSs, their properties, and a table for the congruence rules.

If multiple PSs are congruent, we store POSTs for only one (called a *base position sequence*) of them. Then, we can obtain POSTs for the other PSs by properly transforming the POSTs stored for their base PS. We use the following rule to determine base PSs. When pin locations are given, we find eight PSs $\Gamma_{(y_\pm,x_\pm)}$ and $\Gamma_{(x_\pm,y_\pm)}$ for them and choose the smallest PS for its base PS. In Figure 7, for example, (24513) in Figure 7(g) is the smallest number, so (24513) becomes the base PS for all the PSs in Figure 7.

3.2.2 Congruence of 3D POSTs. Suppose pin locations are given in the 3D space. Then, we can characterize the pin locations by two sequences: a PS and a **Tier Sequence (TS)**. The PS is based on the projection of the pins onto the xy plane, and the TS is based on the z-coordinates of the pins. Figure 8 shows an example. In Figure 8(a), the z-coordinates of the pins corresponding to the PS elements 3, 1, 5, 4, 2 are 0, 1, 0, 1, 0, respectively. Thus, the TS for the pin locations is (01010).

If we rotate the two tiers in Figure 8(a) counterclockwise by 90, 180, and 270 degrees around the z-axis, we obtain the PSs and TSs shown in Figure 8(b), (c), and (d), respectively. In addition, if we reflect the two tiers in Figure 8(a) over the yz plane, we obtain the PS and TS in Figure 8(e). Rotating the two tiers in Figure 8(e) counterclockwise by 90, 180, and 270 degrees around the z-axis leads to the PSs and TSs in Figure 8(f), (g), and (h), respectively. Moreover, reflecting the two tiers in Figure 8(a) and (e) over the xy plane generates the PSs and TSs in Figure 8(i) and (m), respectively. Rotating them counterclockwise by 90, 180, and 270 degrees around the z-axis generates the PSs and TSs in Figure 8(j), (k), and (l), and Figure 8(n), (o), and (p), respectively.

To find a congruence between two sets of PSs and TSs, we define a *3D position sequence* $\Lambda_{(s,r,w)}$, which consists of a pair of sequences. The first sequence is the 2D PS $(a_1 \ldots a_n)$ obtained from $\Gamma_{(s,r)}$. The second sequence is the TS along the w-direction ($w \in \{z_+, z_-\}$) as defined previously. Then, the 3D PS for the pins in Figure 8(a) is denoted by $\Lambda_{(y_+,x_+,z_+)}$. Similarly, 3D PSs for the pins in Figure 8(b), (c), (d), (e), (f), (g), and (h) are $\Lambda_{(x_+,y_-,z_+)}$, $\Lambda_{(y_-,x_-,z_+)}$, $\Lambda_{(x_-,y_+,z_+)}$, $\Lambda_{(y_+,x_-,z_+)}$, $\Lambda_{(x_-,y_-,z_+)}$, $\Lambda_{(y_-,x_+,z_+)}$, and $\Lambda_{(x_+,y_+,z_+)}$, respectively. Since the reflection over the xy plane reverses the TS, 3D PSs for the pins in Figure 8(i), (j), (k), (l), (m), (n), (o), and (p) are $\Lambda_{(y_+,x_+,z_-)}$, $\Lambda_{(x_+,y_-,z_-)}$, $\Lambda_{(y_-,x_-,z_-)}$, $\Lambda_{(x_-,y_+,z_-)}$, $\Lambda_{(y_+,x_-,z_-)}$, $\Lambda_{(x_-,y_-,z_-)}$, $\Lambda_{(y_-,x_+,z_-)}$, and $\Lambda_{(x_+,y_+,z_-)}$, respectively. If two sets of pin locations are congruent, we can use the 3D POSTs belonging to one of them for the other by properly transforming the 3D POSTs.

We also define a *3D base position sequence* as follows. Suppose a set of pin locations is given in the 3D space. Then, we find all the 16 3D PSs $\Lambda_{(y_\pm,x_\pm,z_\pm)}$ and $\Lambda_{(x_\pm,y_\pm,z_\pm)}$ for them and choose the smallest 3D PS for their 3D base PS. If multiple 3D PSs have the same 2D PS, the one with

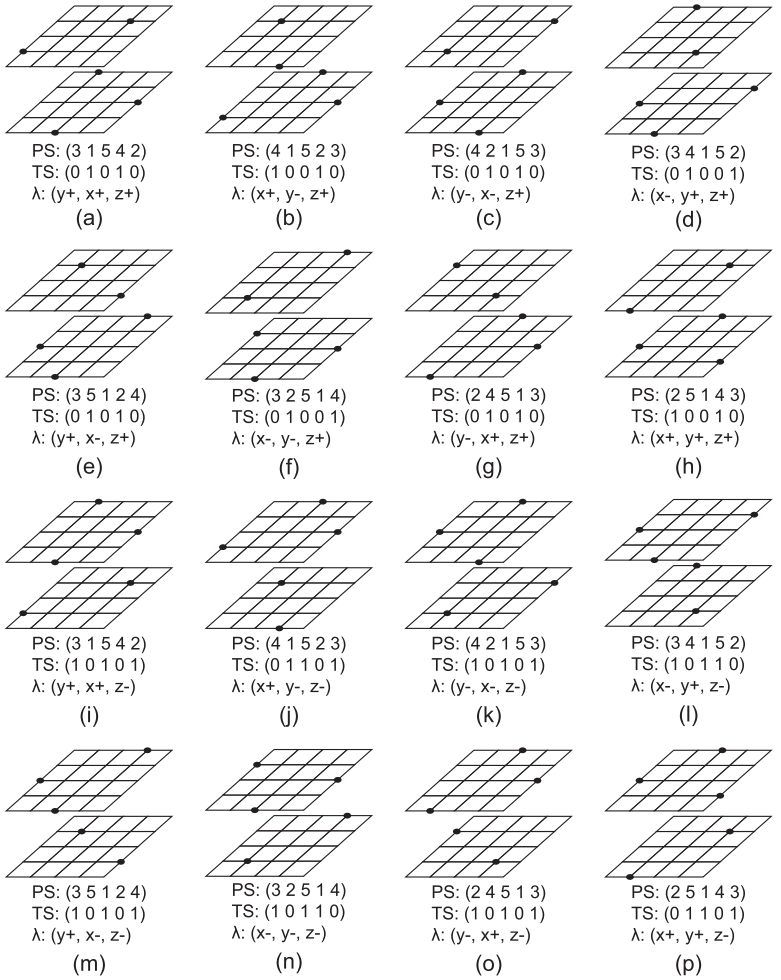

Fig. 8. Congruence of 16 PSs and TSs.

the smallest TS becomes the 3D base PS for them. In Figure 8, for example, the smallest 2D PS is (24513) in Figure 8(g) and (o). Between these two, the TS (01010) is smaller than the TS (10101), so the 3D PS of Figure 8(g) becomes the 3D base PS for all the 3D PSs in Figure 8. Appendix A also shows operations defined for 3D PSs, their properties, and a table for the congruence rules.

4 DB GENERATION

In this section, we present simulation results obtained from the construction of all 3D POSTs on the 3D Hanan grid. We implemented the proposed algorithm using C/C++ and ran the code in an Intel Core i5-6600K 3.3-GHz CPU system with 64 GB of memory. We used the 2D POST DB in previous work [9]. Table 1 shows some statistics of the construction of all 3D POSTs for two- to six-pin nets and two to four tiers.

Our first observation is that as the tier count goes up from 2 to 4, the total number of 3D POSTs increases exponentially. This is because the number of combinations of placing pins in different tiers increases exponentially as the tier count goes up. The recurrence relation for the number of

Table 1. Statistics of the Construction of all 3D POSTs for Two- to Six-Pin Nets for Two, Three, and Four Tiers

# pins (n)	# PS ($n!$)	# 2D POSTs	# tiers	# all 3D POSTs (A)	# gen. 3D POSTs (B)	r (B/A)	Con. time (C)	Con. eff. (B/C)	Table size
2	2	4	2	24	12	0.5	0.0 s	–	< 1 KB
			3	48	24	0.5	0.0001 s	–	< 1 KB
			4	80	40	0.5	0.0001 s	–	< 1 KB
3	6	16	2	224	84	0.375	0.0001 s	–	1 KB
			3	896	336	0.375	0.0003 s	–	2 KB
			4	2,352	888	0.378	0.0006 s	–	4 KB
4	24	284	2	20,056	5,372	0.268	0.0043 s	1,249,302	35 KB
			3	226,800	60,120	0.265	0.0457 s	1,315,536	313 KB
			4	1,396,944	367,424	0.263	0.3465 s	1,060,387	2 MB
5	120	4,260	2	719,864	125,360	0.174	0.1484 s	844,744	850 KB
			3	14,876,928	2,575,092	0.173	5.2478 s	490,699	16 MB
			4	142,195,680	24,482,354	0.172	95.77 s	255,637	167 MB
6	720	120,212	2	85,530,040	13,831,206	0.162	20.13 s	687,094	93 MB
			3	4,318,826,472	697,355,262	0.161	42.2 m	275,417	5.1 GB
			4	90,473,628,112	14,586,090,890	0.161	30.2 h	134,162	129 GB

"# PS" is the number of 2D position sequences for the projected pins. "# all 3D POSTs" is the total number of 3D POSTs (A), and "# gen. 3D POSTs" is the number of 3D POSTs (B) generated from the proposed algorithm. r is B/A. "Con. time" is the DB construction time (C). "Con. eff." is the construction efficiency measured by B/C (# 3D POSTS generated per second).

combinations is as follows:

$$f(n,t) = t^n - \sum_{i=1}^{t-1}\{(t-i+1)\cdot f(n,i)\}, \tag{2}$$

where $f(n,t)$ is the number of combinations of placing n pins in t consecutive tiers. A closed-form expression for $f(n,t)$ is as follows:

$$f(n,t) = t^n - 2\cdot(t-1)^n + (t-2)^n, \tag{3}$$

$$f(n,1) = 1. \tag{4}$$

Thus, as t increases, $f(n,t)$ goes up polynomially, and as the pin count goes up, the number of 2D POSTs increases exponentially as shown in the table. Thus, the total number of 3D POSTs increases extremely fast as the pin and tier counts go up. The number of generated 3D POSTs is approximately 16% of the total 3D POSTs. As explained in Section 3.2, using the congruence properties of PSs and 3D POSTs significantly reduces the number of 3D POSTs generated. Thus, we reduce the construction time and the DB size effectively.

The construction efficiency measured by the ratio between the number of generated 3D POSTs and the total construction time decreases almost exponentially as the pin count and the tier count go up. The algorithm can still construct approximately 130,000 3D POSTs per second for the six-pin four-tier case. However, there are almost 15 billion 3D POSTs to generate for the case, so the construction time is about 30 hours. The table size is approximately 135 GB, which can be easily handled in server computers.

Figure 9 shows two 3D POSTs constructed for the given six pins and the same 2D POST. The red edges are planar wires, and the blue edges are MIVs. The 3D POST in Figure 9(a) has five

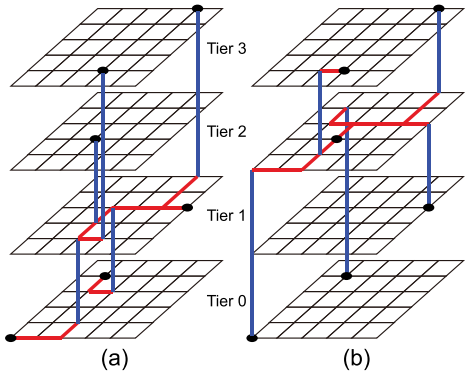

Fig. 9. Comparison of two 3D POSTs constructed for pin locations $(0, 0, 0)$, $(3, 1, 3)$, $(2, 2, 2)$, $(5, 3, 1)$, $(1, 4, 0)$, $(4, 5, 3)$. 3D PS $\Lambda_{(y_+, x_+, z_+)}$: $PS = (143625)$, $TS = (032103)$.

planar edges in Tier 0 and eight planar edges in Tier 1. However, the 3D POST in Figure 9(b) has 12 planar edges in Tier 2 and a planar edge in Tier 3. In addition, the planar coordinates of the MIVs in Tier 1 in Figure 9(a) are $(2, 1)$ and $(2, 3)$, whereas those in Tier 1 in Figure 9(b) are $(0, 0)$ and $(1, 4)$. Similarly, the planar coordinates of the MIVs in Tier 2 in Figure 9(a) are $(3, 1)$, $(2, 2)$, and $(4, 5)$, whereas those in Figure 9(b) are $(0, 0)$, $(1, 4)$, and $(5, 3)$. The planar coordinates of the MIVs in Tier 3 in Figure 9(a) are $(3, 1)$ and $(4, 5)$, whereas those in Figure 9(b) are $(2, 1)$ and $(4, 5)$. Thus, among the seven MIVs inserted in the two 3D POSTs, only one MIV located at $(4, 5, 3)$ is common and the other MIVs are located at quite different locations. We also found similar trends in many other 3D POSTs. Thus, we expect that the DB of the 3D POSTs can be used for 3D routing to evenly distribute planar wires and MIVs across the tiers.

5 APPLICATION I: 3D ROUTING TOPOLOGY GENERATION

In this section and the next section, we present two applications for the practical use of the ARD. The first application is constructing 3D routing topologies for the optimization of given metrics. The two metrics we optimize are the **Source-to-Critical-Sink Length (SCSL)** used for wirelength minimization and the **Source-to-Critical-Sink Delay (SCSD)** used for timing optimization. The second application is congestion-aware 3D global routing for the minimization of routing congestion in M3D IC layouts.

5.1 Motivation

Figure 10 shows a two-tier 3D placement result for a five-pin net to be routed. The 3D PS is $\Lambda_{(y_+, x_+, z_+)} = ((31542), (00011))$, and the POWV is (12211111). If we assume the length of each planar edge is l and the length of an MIV is l_m, then all 3D POSTs in the figure have the same length of $(10l + l_m)$.

Suppose the source of the net is p_5 and the critical sink (the sink that has the smallest slack) is p_4. Then, the SCSL is $(2l + l_m)$ in Figure 10(a), but that in Figure 10(b) is $(4l + l_m)$, so the former has a shorter SCSL. Similarly, suppose the source is p_2 and the critical sink is p_4. Then, all the topologies in Figure 10 have the same SCSL of $3l$. We can also compute the SCSD using the PI model for the edges and the Elmore delay model for the delay estimation. Suppose the output resistance of the source is R_D, the input capacitance of each sink is C_L, the RC of an edge are r and c, respectively, and the RC of an MIV are r_m and c_m, respectively. Then, if the source is p_5 and the critical sink is p_4, the SCSD in Figure 10(a) is smaller than that in Figure 10(b) by $r(5C_L + 9c + c_m)$. Moreover, the

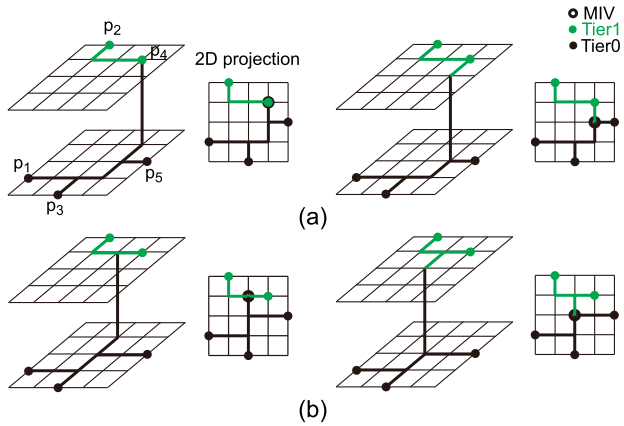

Fig. 10. Four MMRSMTs for the routing of a five-pin net in a two-tier design. 3D PS $\Lambda_{(y_+,x_+,z_+)}$: PS = (31542), TS = (00011), $POWV$ = (12211111).

Table 2. Comparison of the SCSL and SCSD for the Topologies in Figure 10

	source:$p_5 \rightsquigarrow$ sink:p_4		source:$p_2 \rightsquigarrow$ sink:p_4	
	SCSL	SCSD	SCSL	SCSD
(a) left	$\boldsymbol{2l+l_m}$	$\boldsymbol{T_D + r(6C_L+13c+2c_m) + r_m(2C_L+3c+0.5c_m)}$	$\boldsymbol{3l}$	$T_D + r(12C_L+25.5c+3c_m)$
(a) right	$\boldsymbol{2l+l_m}$	$\boldsymbol{T_D + r(6C_L+13c+c_m) + r_m(2C_L+4c+0.5c_m)}$	$\boldsymbol{3l}$	$T_D + r(12C_L+25.5c+3c_m)$
(b) left	$4l+l_m$	$T_D + r(11C_L+22c+3c_m) + r_m(2C_L+3c+0.5c_m)$	$\boldsymbol{3l}$	$\boldsymbol{T_D + r(9C_L+18.5c+2c_m)}$
(b) right	$4l+l_m$	$T_D + r(11C_L+22c+2c_m) + r_m(2C_L+4c+0.5c_m)$	$\boldsymbol{3l}$	$\boldsymbol{T_D + r(9C_L+18.5c+2c_m)}$

$T_D = R_D(4C_L+10c+c_m)$.

two topologies in Figure 10(a) have the same 2D projection with different MIV locations. Thus, the topology on the left has less SCSD than the one on the right if ($\frac{r}{c} < \frac{r_m}{c_m}$), whereas the right one has less SCSD if ($\frac{r}{c} > \frac{r_m}{c_m}$). In addition, if the source is p_2 and the critical sink is p_4, the SCSD of the topologies in Figure 10(b) is smaller than that in Figure 10(a) by $r(3C_L + 7c + c_m)$ as shown in Table 2. In conclusion, the effectiveness of a particular topology for a specific metric is dependent on the locations of the source and sinks and can be maximized only after examining all the MMRSMTs.

5.2 Simulation Methodology

To show the effectiveness of the use of the ARD, we compare two 3D routing topology generation approaches. The first is a so-called BF approach that selects one MMRSMT for each 3D net. If a 3D net is given, we select the first MMRSMT found in the ARD for the pin locations of the 3D net. The second approach is using the ARD for which we search the ARD for a given 3D net, find all MMRSMTs, and select the best one for a given metric (SCSL or SCSD). We used the ISPD 2005 and 2006 benchmarks [16, 17] and ePlace-3D [15] to generate 3D placement results in two, three, and four tiers. For the SCSD computation, we assume that the output resistance of a source (driver) is 100Ω, the wire RC per unit length are 2Ω/unit and $0.4fF$/unit, respectively, and the load capacitance of a sink pin is $5fF$ [12]. The MIV height, resistance, and capacitance are 140 nm, 4Ω, and $1fF$, respectively [7, 18]. For each 3D net, we set the source pin to the driver node of the net from the benchmark suites, randomly selected a pin for the critical sink, constructed two 3D routing topologies, one by the BF approach and the other by the ARD, and compared their SCSLs and SCSDs.

5.3 Simulation Results

Table 3 shows the comparison of the SCSLs of the 3D routing topologies constructed by the BF- and ARD-based approaches for all the 3D nets of the benchmarks with net degree 4 and 6. Notice that all 2D POSTs for each two- or three-pin net have the same SCSL regardless of the selection of the source and critical-sink pins. Therefore, the 3D routing topologies constructed for these nets by BF and ARD have the same SCSL. The BF and ARD have different SCSLs for approximately 20.70% to 23.54% of the 3D nets (N_L/N). Nonetheless, the average SCSL differences (L_T/N_L) between BF and ARD are approximately 103, 92, and 86 for the two-, three-, and four-tier designs, respectively. Moreover, we compute the sum of the ratios of each SCSL difference to its average to obtain the average difference ratio, which are 0.09, 0.11, and 0.14, respectively, for the two-, three-, and four-tier designs. The maximum SCSL differences are also very large (840 to 7,080), so the comparison shows that ARD can effectively minimize the SCSL for each 3D net.

We also observe that the average SCSL difference (L_T/N_L) between BF and ARD generally goes down as the tier count goes up from 2 to 4. This is because stacking more tiers helps reduce the wirelength of each net. For example, the uniform-scaling-based 3D placement [1, 13] ideally reduces the wirelength of a net by $1/\sqrt{t}$, where t is the number of tiers. As a result, the average SCSL difference also goes down as the tier count increases. However, the maximum SCSL difference is dependent not only on the 3D placement result but also on whether the 3D routing topologies constructed by BF can minimize the SCSLs by accident. Thus, the maximum SCSL difference does not go down even if the tier count goes up as shown in the table.

We observe similar trends in the SCSD simulation results. First of all, BF and ARD have different SCSDs for approximately 53.28% to 57.40% of the 3D nets (N_D/N). The reason that N_D/N is greater than N_L/N is that two 3D routing topologies with the same SCSL can have different SCSDs as shown in Figure 10 and Table 2. The average SCSD differences (D_T/N_D) between BF and ARD are approximately 222ps, 195ps, and 122ps for the two-, three-, and four-tier designs, respectively. The maximum SCSD differences are also large as shown in the table. Moreover, the SCSD average difference ratios are 0.14, 0.15, and 0.17, respectively, for the two-, three-, and four-tier designs. Although several interconnect optimization techniques such as buffer insertion would help reduce the SCSD, finding a good 3D routing topology would still be one of the most important interconnect optimization techniques. As shown previously, the ARD can provide multiple 3D routing topologies optimal for different metrics such as SCSL and SCSD.

Note that in some cases, minimizing the SCSL (or SCSD) of a net may not be compatible with constructing its RSMT. As the papers related to this work [2, 12] aimed at minimizing the planar wirelength (and then minimizing # vertical edges), we do not construct minimum-SCSL (or minimum-SCSD) topologies either. Instead, we find topologies minimizing the SCSL (or SCSD) among the MMRSMTs for a given net.

6 APPLICATION II: CONGESTION-AWARE 3D ROUTING

In this section, we present the use of ARD for the minimization of routing congestion in M3D IC layouts. As shown in Figure 9, MMRSMTs might use very different 3D routing topologies. Thus, we can minimize routing congestion by selecting a good MMRSMT after an exhaustive search of all MMRSMTs for each 3D net.

6.1 Simulation Methodology

We used ePlace-3D [15] for 3D placement and bin-based 3D global routing for the congestion-aware 3D routing. Each x- or y-directional edge $e_{c,i,j,k}$ ($c \in \{x, y\}$) has a predetermined maximum capacity $m_{c,i,j,k}$ and the # nets $s_{c,i,j,k}$ crossing the edge. Similarly, each bin $bin_{z,i,j,k}$ has a

Table 3. Comparison of the SCSL and SCSD of 3D Routing Topologies Constructed by BF- and ARD-Based Routing

# T	Bench.	SCSL					SCSD				
		Avg. Diff.	# d. nets (%)	Avg. Diff. (d.)	Max. Diff.	Avg. Diff. ratio	Avg. Diff. (ps)	# d. nets (%)	Avg. Diff. (d.) (ps)	Max. Diff. (ps)	Avg. Diff. ratio
		$\frac{L_T}{N}$	$\frac{N_L}{N} \times 100$	$\frac{L_T}{N_L}$		$\frac{R_L}{N_L}$	$\frac{D_T}{N}$	$\frac{N_D}{N} \times 100$	$\frac{D_T}{N_D}$		$\frac{R_D}{N_D}$
2	adaptec1	11.72	19.89	58.95	960	0.13	33.94	58.09	58.43	3,698	0.16
	adaptec2	20.63	18.78	109.85	2,400	0.15	69.78	56.35	123.83	8,530.1	0.16
	adaptec3	19.31	15.52	124.45	2,400	0.03	218.37	37.36	584.56	14,452.8	0.09
	adaptec4	52.91	32.91	160.78	3,600	0.06	233.47	43.13	541.3	15,454.8	0.13
	adaptec5	36.8	25.37	145.02	4,560	0.10	243.94	45.81	532.45	52,503.8	0.13
	bigblue1	16.2	23.97	67.6	1,080	0.15	33.25	58.49	56.85	2,964.4	0.18
	bigblue2	61.71	34.29	180	1,320	0.07	711.74	77.14	922.63	10,225.8	0.18
	bigblue3	19.83	21.66	91.59	7,080	0.09	172.77	58.86	293.5	78,146.4	0.12
	newblue1	22.51	21.43	105.04	2,520	0.15	82.85	58.04	142.76	7,469.3	0.15
	newblue2	16.28	21.07	77.26	1,440	0.14	37.79	58.44	64.67	4,499.6	0.17
	newblue4	13.1	25.03	52.34	1,680	0.11	55.15	58.8	93.79	6,011	0.16
	newblue5	32.73	22.51	145.38	3,240	0.09	153.74	42.91	358.31	14,339	0.11
	newblue6	31.24	27.3	114.42	1,440	0.04	271.05	50.79	533.63	12,109.6	0.10
Geo. mean		24.02	23.31	103.06		0.09	118.07	53.28	221.60		0.14
3	adaptec1	16.98	25.2	67.38	1,680	0.09	77.2	64.77	119.2	8,446	0.14
	adaptec2	30.1	25.29	119.01	5,040.3	0.08	220.92	56.54	390.73	32,344.4	0.13
	adaptec3	20	26.3	76.03	2,160	0.08	153.44	55.06	278.67	21,513.4	0.15
	adaptec4	11.81	20.98	56.28	1,680	0.11	60.12	44.86	134.01	6,831.3	0.15
	adaptec5	16.42	22.27	73.74	2,400	0.13	66.78	58.05	115.05	18,701	0.18
	bigblue1	15.15	21.77	69.59	960	0.13	43.38	58	74.8	3,217.8	0.18
	bigblue2	20.01	17.79	112.45	1,680	0.20	64.85	56.98	113.81	4,254.6	0.16
	bigblue3	41.59	23.75	175.09	6,600	0.11	524.81	60.61	865.87	87,378.8	0.15
	newblue1	16.86	23.65	71.31	1,680	0.10	68.88	57.39	120.01	5,297.7	0.16
	newblue2	19.53	23.28	83.88	4,800	0.12	74.37	60.22	123.5	21,886.9	0.16
	newblue4	35.56	25.49	139.52	3,960	0.10	140.29	54.93	255.39	13,062.6	0.17
	newblue5	26.21	25.75	101.8	2,640	0.08	218.55	61.56	354.99	17,204.6	0.13
	newblue6	31.11	26.23	118.57	1,560	0.10	172.25	59.69	288.6	14,646.8	0.16
Geo. mean		21.71	23.54	92.23		0.11	111.66	57.40	194.55		0.15
4	adaptec1	15.8	20.7	76.34	2,280	0.14	68.95	60.15	114.64	8,235.6	0.17
	adaptec2	31.03	22.71	136.67	2,640	0.15	84.2	61.79	136.26	11,941.9	0.15
	adaptec3	18.27	20.58	88.76	3,840	0.14	93.78	53.5	175.29	27,445.7	0.18
	adaptec4	18.22	18.9	96.4	1,920	0.15	71.88	54	133.1	9,576.5	0.17
	adaptec5	14.04	21.43	65.5	2,160	0.12	47.64	54.77	86.98	7,849.3	0.17
	bigblue1	12.57	21.85	57.52	1,080	0.15	22.72	60.3	37.68	1,831.4	0.18
	bigblue2	9.56	12.73	75.05	840	0.28	17.21	53.46	32.19	1,272.1	0.20
	bigblue3	21.37	21.3	100.35	3,480	0.13	168.48	59.02	285.45	33,586.7	0.16
	newblue1	16.93	21.94	77.16	2,160	0.12	69.15	54.65	126.52	5,406.8	0.19
	newblue2	19	24.08	78.93	6,960	0.10	113.97	57.07	199.71	83,711.1	0.15
	newblue4	17.07	21.95	77.79	2,280	0.14	57.21	56.27	101.67	6,777.1	0.17
	newblue5	22.61	22.57	100.14	2,400	0.13	104.25	56.96	183.03	10,346.5	0.16
	newblue6	24.5	21.02	116.55	2,640	0.14	144.71	57.07	253.55	15,780.2	0.17
Geo. mean		17.79	20.70	85.97		0.14	69.21	56.79	121.87		0.17

N, # four- to six-pin 3D nets; L_T (D_T), the sum of the SCSL (or SCSD) differences; R_L (R_D), the sum of the ratios of each SCSL (or SCSD) difference to its average; N_L (N_D), # 3D nets with nonzero SCSL (or SCSD) differences. "# T" denotes the number of tiers, "# d. nets (%)" denotes how many of the 3D nets have nonzero SCSL (or SCSD) differences. "Max. Diff." denotes the maximum SCSL (or SCSD) differences.

predetermined maximum MIV capacity $m_{bin,i,j,k}$ and the # MIVs $s_{bin,i,j,k}$ located on that bin. We compute the overflow $OF_{c,i,j,k}$ (or $OF_{bin,i,j,k}$) of edge $e_{c,i,j,k}$ (or bin $bin_{z,i,j,k}$) as follows:

$$OF_{w,i,j,k} = \begin{cases} (s_{w,i,j,k} - m_{w,i,j,k}), & \text{if } s_{w,i,j,k} > m_{w,i,j,k} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $w \in \{c, bin\}$. The objective is to minimize the total overflow in the following equation while routing all the 2D and 3D nets sequentially.

$$OF_{Combined} = \alpha \cdot OF_{c,i,j,k} + \beta \cdot OF_{bin,i,j,k} \tag{6}$$

We chose α and β suitably. We estimate the maximum MIV capacity of a bin by deducting the total area occupied by all instances of that bin from the total bin area and dividing the resulting area by the MIV pitch area. Moreover, MIV violations occur when the number of MIVs placed in a bin exceeds its maximum MIV capacity.

We route all the 2D and 3D nets of each design using two routing methodologies similar to the BF- and ARD-based routing methodologies used in Section 5. We route each net as follows:

- *2D nets (≤ 8 pins)*: BF uses the FLUTE DB, so it finds only one RSMT for a 2D net. ARD uses the ARSMT DB, so it finds all RSMTs for a 2D net and selects the best one minimizing the overflow.
- *2D nets (> 8 pins)*: Both BF and ARD use the net-breaking technique proposed in FLUTE [2]. The net-breaking decomposes a high-degree net into multiple low-degree nets, uses the FLUTE DB to find an RSMT for each low-degree net, and inserts some additional (Steiner) points to connect the low-degree nets. Thus, BF and ARD use only one RSMT for a 2D net in this case.
- *3D nets* (≤ 6 pins): BF uses the ARD, but it finds only one MMRSMT in the DB and uses it for a 3D net. ARD also uses the ARD and finds all MMRSMTs for a 3D net and selects the best one minimizing the overflow.
- *3D nets* (> 6 pins): Both BF and ARD use a net-breaking technique shown in the following.

However, we made an exception for the six-pin-four-tier case and used the net-breaking technique shown in the following for both the BF and ARD due to memory limitations. Moreover, we also routed all the nets of each design using MLOARST construction algorithms [8] to assess the efficacy of the ARD-based routing approach.

Note that global routing is a coarse-level bin-based routing step focusing on constructing routing topologies for a given design on a single routing layer under maximum capacity constraints. On the contrary, detailed routing that includes track assignment is a more fine-grained routing step based on the routing topologies obtained in the global routing step. In brief, since a single routing layer is often used for global routing purposes in the literature, we also consider similar conventions and problem definitions in this research.

6.2 3D Net-Breaking Techniques

Suppose N pins of a 3D net, $P = \{p_1, p_2, \ldots, p_N\}$, are given and its 3D PS is $((s_1 s_2 \ldots s_N), (t_1 t_2 \ldots t_N))$. If a group of the pins belongs to an octant and the others belong to its opposite octant, we can break the pins into two groups, construct an MMRSMT for each group, and connect the two MMRSMTs using an additional point. Figure 11(a) illustrates the octant pairs geometrically opposite in the 3D space. For example, if $P_{G1} = \{p_1, \ldots, p_r\}$ belongs to the octant $x_+ y_+ z_+$ and $P_{G2} = \{p_{r+1}, \ldots, p_N\}$ belongs to the opposite octant $x_- y_- z_-$, we can find $p_i \in P_{G1}$ and $p_j \in P_{G2}$ closest to the origin. Then, we can insert a point p_h in the hexahedron constructed with p_i and p_j as the two endpoints of the hexahedron. Then, the union of the two MMRSMTs constructed for $P_{G1} \cup \{p_h\}$ and $P_{G2} \cup \{p_h\}$ is

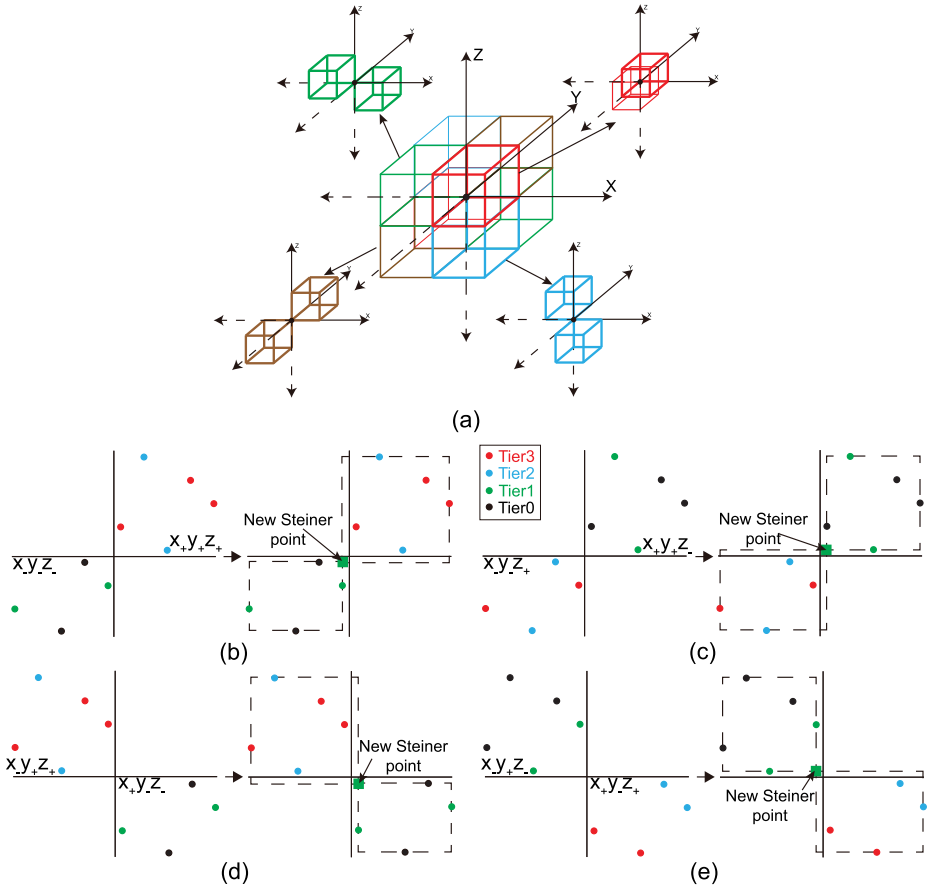

Fig. 11. 3D optimal net-breaking techniques. (a) Four octant pairs opposite to each other in 3D space. The octant pairs in brown, green, cyan, and red are detailed through (b) $x_-y_-z_- \rightarrow x_+y_+z_+$, (c) $x_-y_-z_+ \rightarrow x_+y_+z_-$, (d) $x_+y_-z_- \rightarrow x_-y_+z_+$, and (e) $x_+y_-z_+ \rightarrow x_-y_+z_-$, respectively.

an MMRSMT for P. Figure 11(b) through (e) demonstrate the four octant pairs, $(x_-y_-z_- \rightarrow x_+y_+z_+)$, $(x_-y_-z_+ \rightarrow x_+y_+z_-)$, $(x_+y_-z_- \rightarrow x_-y_+z_+)$, and $(x_+y_-z_+ \rightarrow x_-y_+z_-)$, that can be used for 3D optimal net breaking.

Let S_{G1} and S_{G2} be the PS values of P_{G1} and P_{G2}, respectively. Similarly, let T_{G1} and T_{G2} be the TS values of P_{G1} and P_{G2}, respectively. Then, the following inequalities show the conditions for the 3D optimal net breaking:

$$\max(S_{G1}) \leq \min(S_{G2}) \ \ \& \ \ \max(T_{G1}) \leq \min(T_{G2}), \tag{7}$$

$$\max(S_{G1}) \leq \min(S_{G2}) \ \ \& \ \ \min(T_{G1}) \geq \max(T_{G2}), \tag{8}$$

$$\min(S_{G1}) \geq \max(S_{G2}) \ \ \& \ \ \max(T_{G1}) \leq \min(T_{G2}), \tag{9}$$

$$\min(S_{G1}) \geq \max(S_{G2}) \ \ \& \ \ \min(T_{G1}) \geq \max(T_{G2}), \tag{10}$$

where $\max(A)$ and $\min(A)$ find the maximum and minimum elements in A, respectively, and Inequalities (7) through (10) correspond to the cases shown in Figure 11(b) through (e), respectively. The figures also show new Steiner points (■) inserted for the 3D optimal net breaking. Note that in

Figure 11(b) through (e), dots in red (●), cyan (●), green (●), and black (●) indicate nodes on Tiers 3, 2, 1, and 0, respectively.

Notice that a 3D net cannot be optimally broken if we cannot find P_{G1} and P_{G2} satisfying any of the inequality pairs in (7) through (10). In this case, we first project all the pins of the 3D net onto the xy plane and perform 2D optimal net breaking for the projected pins. If this is successful, there will be two groups of the projected pins, so we construct an MMRSMT for each pin group and connect the two MMRSMTs. If this is not successful, however, we use the 2D net-breaking heuristics for the projected pins [2], construct an MMRSMT for each pin group, and connect all the MMRSMTs.

6.3 Simulation Results

Table 4 compares the planar overflows and the number of MIV violations of the congestion-aware 3D global routing by the BF- and ARD-based routing. The number of global nets shows the total number of nets routed optimally (by 2D RSMTs and MMRSMTs) or nonoptimally, whereas the number of routed nets shows the total number of nets routed optimally. For all the designs, most of the nets (around 94% on average) are routed optimally because they are low-degree nets. Table 5 shows the details of the # "k-Pin" nets for each "k."

For the two-tier designs, the ratios of the total number of planar overflows and MIV violations between BF and ARD are 0.25 and 0.47 on average, respectively, which demonstrate that the use of ARD can reduce the routing congestion effectively. ARDs are effective because they minimize planar overflow while distributing MIVs according to available whitespace. For each net, the minimum planar wirelength is used and then MIVs are minimized. The average planar overflows (# planar overflows per edge capacity) and the maximum planar overflows of ARD are also lower than those of BF by 48% to 93% and 11% to 74%, respectively. Furthermore, the # MIV violations of ARD is 20% to 68% fewer than their BF counterparts. Note that a higher number of tiers means more bins accepting MIVs, which boosts the overall capacity of MIVs. Consequently, we expect fewer MIV violations. Nevertheless, the # MIVs will also rise due to the increasing number of 3D wires. Thus, with more tier inclusion, the # MIV violations varies either way (increasing or decreasing).

The runtime of the BF approach is less than 1 second for small benchmarks and maximum 2 seconds for the largest design. However, ARD takes 4 to 30 seconds for the routing of all the nets. The runtime overhead is negligible, but the overflow reduction is significantly large, which shows the effectiveness of the ARD for the routing congestion minimization. We observe similar trends in the three- and four-tier designs. The planar overflow ratio between the BF and ARD designs is 0.24 for the three-tier designs and 0.27 for the four-tier designs on average provided 49% less MIV violations on average for both the cases. ARD still achieves 11% to 83% lower maximum planar overflows with 38% to 63% fewer MIV violations for the three-tier designs and 29% to 69% lower maximum planar overflows with 38% to 68% fewer MIV violations for the four-tier designs.

Table 6 compares the planar overflows and the number of MIV violations of congestion-aware 3D global routing by the MLOARST construction algorithms (denoted by MR) and ARD-based routing. For the two-, three-, and four-tier designs, the ratios of the total number of planar overflows between MR and ARD are 0.26, 0.25, and 0.25 on average, respectively, and are 0.18, 0.16, and 0.16 on average, respectively, for the number of MIV violations. ARD's average and maximum planar overflows are consistently lower with significantly fewer # MIV violations compared to MR. This is because even with the minimized planar wirelength and MIV count, the MR only generates one routing topology demonstrating the effectiveness of ARD with multiple topologies to reduce routing congestion. However, only for the four-tier bigblue2 design, the ARD shows slightly more planar and maximum overflow than MR. This is because nets were sequentially routed in the ARD.

Table 4. Comparison of the Planar Edge Overflows and the # MIV Violations of Congestion-Aware 3D Global Routing by BF- and ARD-Based Routing

# T	Bench.	# Bins per tier	# Nets		# Planar overflows		Avg. overflow		Max. overflow		# MIV violations		RT (s)
			Global	Routed	BF	ARD	BF	ARD	BF	ARD	BF	ARD	
2	adaptec1	140×140	152.4k	142.03k (0.93)	4872	1485 (0.3)	0.06	0.02	171	58	685	356 (0.52)	4.12
	adaptec2	184×184	165.04k	153.46k (0.93)	20005	5470 (0.27)	0.15	0.04	239	61	885	486 (0.55)	5.03
	adaptec3	289×287	303.61k	286.45k (0.94)	44437	3000 (0.07)	0.13	0.01	240	70	980	498 (0.51)	10.62
	adaptec4	282×285	322.19k	308.03k (0.96)	11476	1183 (0.1)	0.04	0	145	62	1947	621 (0.32)	8.68
	adaptec5	289×287	517.4k	487.8k (0.94)	90861	18256 (0.2)	0.27	0.06	338	143	1544	697 (0.45)	17.4
	bigblue1	140×140	187.19k	174.48k (0.93)	23006	9196 (0.4)	0.3	0.12	223	198	962	451 (0.47)	5.14
	bigblue2	233×232	334.71k	318.89k (0.95)	0	0 (0)	0	0	0	0	729	580 (0.8)	8.42
	bigblue3	344×343	601.03k	569.08k (0.95)	516833	220542 (0.43)	1.1	0.47	1360	498	3204	1564 (0.49)	18.29
	newblue1	148×148	195.92k	187.13k (0.96)	10160	1962 (0.19)	0.12	0.02	149	68	686	232 (0.34)	4.55
	newblue2	286×344	353.82k	332.82k (0.94)	519503	103129 (0.2)	1.32	0.26	797	627	3149	1563 (0.5)	8.68
	newblue4	226×225	432.65k	410.11k (0.95)	31189	10346 (0.33)	0.15	0.05	218	95	2914	1392 (0.48)	10.89
	newblue5	309×314	678.91k	622.5k (0.92)	125248	64738 (0.52)	0.32	0.17	531	301	1900	1078 (0.57)	29.92
	newblue6	343×340	836.87k	785.75k (0.94)	43011	20794 (0.48)	0.09	0.04	263	173	881	297 (0.34)	30.47
Geo. mean			0.94		0.25						0.47		
3	adaptec1	116×116	151.89k	141.28k (0.93)	6008	1440 (0.24)	0.08	0.02	131	62	1140	682 (0.6)	4.07
	adaptec2	152×152	164.93k	153.43k (0.93)	15361	2190 (0.14)	0.11	0.02	136	53	1402	864 (0.62)	5.99
	adaptec3	236×235	303.25k	285.45k (0.94)	105416	21717 (0.21)	0.32	0.07	641	244	1951	794 (0.41)	9.31
	adaptec4	231×233	318.46k	304.27k (0.96)	9605	697 (0.07)	0.03	0	110	50	1812	678 (0.37)	7.44
	adaptec5	236×235	515.16k	484.16k (0.94)	106255	23917 (0.23)	0.32	0.07	350	150	3050	1835 (0.6)	16.6
	bigblue1	116×116	182.98k	170.35k (0.93)	22690	9905 (0.44)	0.28	0.12	223	131	1485	720 (0.48)	5.08
	bigblue2	190×189	333.12k	317.12k (0.95)	18	3 (0.17)	0	0	12	2	951	591 (0.62)	7.71
	bigblue3	281×280	600.61k	567.58k (0.95)	540436	132427 (0.25)	1.15	0.28	880	309	8479	5248 (0.62)	33.26
	newblue1	123×123	194.36k	185.79k (0.96)	14850	3612 (0.24)	0.16	0.04	156	89	998	432 (0.43)	3.58
	newblue2	234×281	355.91k	334.71k (0.94)	566695	125180 (0.22)	1.44	0.32	809	724	4253	2628 (0.62)	10.3
	newblue4	185×184	429.16k	406.88k (0.95)	30577	15436 (0.5)	0.15	0.08	248	145	3159	1257 (0.4)	10.59
	newblue5	258×257	694.69k	636.79k (0.92)	315083	123946 (0.39)	0.8	0.31	772	262	2481	1218 (0.49)	29.68
	newblue6	281×278	829.96k	778.91k (0.94)	28151	11063 (0.39)	0.06	0.02	164	118	2317	997 (0.43)	31.5
Geo. mean			0.94		0.24						0.51		
4	adaptec1	101×101	152.77k	142.04k (0.93)	4230	1058 (0.25)	0.05	0.01	187	81	1724	956 (0.55)	3.73
	adaptec2	133×133	162.43k	150.92k (0.93)	12140	6572 (0.54)	0.09	0.05	287	165	1947	1202 (0.62)	5.04
	adaptec3	204×203	302.21k	283.98k (0.94)	95171	19696 (0.21)	0.29	0.06	470	307	3448	1765 (0.51)	7.9
	adaptec4	204×203	315.78k	301.58k (0.96)	18230	2388 (0.13)	0.06	0.01	164	76	1791	605 (0.34)	7.07
	adaptec5	204×203	515.64k	483.84k (0.94)	92752	14116 (0.15)	0.28	0.04	409	127	3957	2210 (0.56)	14.91
	bigblue1	101×101	182.69k	169.35k (0.93)	20833	5945 (0.29)	0.26	0.07	271	117	2212	1169 (0.53)	4.25
	bigblue2	165×164	331.8k	315.75k (0.95)	22	7 (0.32)	0	0	15	6	1400	856 (0.61)	7.13
	bigblue3	254×253	591.64k	559.22k (0.95)	448410	170041 (0.38)	0.88	0.33	1880	670	7140	4310 (0.6)	16.5
	newblue1	112×112	192.63k	183.94k (0.95)	13648	2495 (0.18)	0.14	0.03	196	71	1256	401 (0.32)	3.57
	newblue2	211×254	355.85k	334.26k (0.94)	548915	100936 (0.18)	1.29	0.24	927	655	6712	4145 (0.62)	9.67
	newblue4	160×159	429.9k	406.98k (0.95)	34992	15425 (0.44)	0.17	0.08	208	133	3727	1864 (0.5)	9.54
	newblue5	224×223	682.7k	625.42k (0.92)	150013	79262 (0.53)	0.38	0.2	531	359	4268	2537 (0.59)	24.95
	newblue6	243×242	829.69k	778.15k (0.94)	31221	7641 (0.24)	0.07	0.02	259	88	2719	1188 (0.44)	25.49
Geo. mean			0.94		0.27						0.51		

"# T" denotes the number of tiers, "# global nets" is the total number of nets routed optimally (2D RSMTs and MMRSMTs) or nonoptimally, "# routed nets" is the total number of nets routed optimally, and "RT" denotes the runtime which is the global routing time of the ARD-based routing.

Routing topologies were chosen to minimize the total overflow that ultimately depends on the net routing order. Still, the # MIV violations are significantly fewer for ARD compared to MR.

Moreover, Table 7 shows a detailed comparison of the # 3D nets, the total **Half-Perimeter Wirelength (HPWL)**, the MIV distribution on different tiers, and the total # MIVs for the

Table 5. Details of the Benchmarks of the Two-, Three-, and Four-Tier Designs of Bin-Based 3D Global Routing Showing # 2D and 3D "k-Pin" Nets Separately for Each "k"

# T	Bench.	# 2D "k-Pin" nets								# 3D "k-Pin" nets					
		"2"	"3"	"4"	"5"	"6"	"7"	"8"	">8"	"2"	"3"	"4"	"5"	"6"	">6"
2	atec1	89184	24407	10223	5082	3799	2757	2106	9249	1789	1275	722	442	239	1125
	atec2	104314	20381	8413	5316	3618	2626	1932	9488	3946	1286	861	434	334	2089
	atec3	195104	42372	19505	10853	6517	3780	2818	17110	4226	755	203	270	49	44
	atec4	213085	49957	18287	8239	4860	3221	2287	14125	7345	431	229	60	24	42
	atec5	327515	72655	33177	18227	12631	8932	6494	28096	6043	990	532	409	194	1507
	bbl1	110922	27018	12647	7805	5135	3115	1940	10510	3587	1054	529	421	310	2196
	bbl2	231956	47586	20382	7417	4118	2738	2283	15672	1862	508	16	11	8	151
	bbl3	402736	74093	34575	17821	9578	6074	4062	24519	12197	3714	1744	1484	997	7437
	nbl1	129017	27026	11312	5955	3970	2644	1921	7811	3158	1118	492	302	214	981
	nbl2	207175	66225	22466	9872	5299	3253	2403	10465	7511	5084	1560	1157	837	10509
	nbl4	266001	65972	28115	15282	9581	6651	5106	20473	9754	1842	799	615	392	2065
	nbl5	458137	74959	33350	17980	12238	8828	7385	56168	6570	1876	698	367	112	239
	nbl6	533494	119181	56056	29843	19174	13514	9784	51033	3604	472	308	222	100	86
3	atec1	88329	23710	10065	5045	3531	2634	2031	8578	2701	1482	802	521	431	2028
	atec2	104077	19635	8449	5370	3755	2750	1995	10142	4312	1715	691	452	233	1352
	atec3	190546	41207	18665	10036	6086	3667	2530	14550	7677	2113	1361	1002	561	3251
	atec4	210026	48639	17956	7884	4744	3113	2168	13684	8177	868	419	178	94	509
	atec5	321653	69308	31426	17324	11126	8465	5970	24957	10363	3652	2187	1685	1305	6040
	bbl1	107293	26056	12120	7604	5128	3078	1954	10051	4608	1206	564	401	335	2582
	bbl2	229986	46813	20232	7419	4164	2656	2140	15283	2422	842	214	149	81	721
	bbl3	390458	70610	32805	16696	9424	5700	3617	21587	24089	6880	3421	2363	1520	11442
	nbl1	126880	26436	11299	5940	3866	2804	2001	7864	4521	1228	424	210	178	706
	nbl2	207022	65552	21991	9791	5258	3240	2329	9618	8104	6533	2354	1563	972	11583
	nbl4	264174	64030	28169	14898	9198	6671	5099	20880	9581	2859	969	852	380	1401
	nbl5	465807	75718	33335	18024	11977	8803	7080	52534	9638	3390	1477	1040	501	5369
	nbl6	522828	116044	54583	28454	18401	13120	9687	48767	10808	2488	1249	765	479	2285
4	atec1	87345	23750	9938	4830	3554	2647	2011	8084	3809	1971	1083	684	435	2627
	atec2	99656	18864	7801	4923	3533	2608	2019	8929	6683	2547	970	857	502	2536
	atec3	187629	39227	18017	9764	5758	3314	2337	13553	10372	3596	1817	1390	769	4671
	atec4	208665	48111	17620	7759	4570	3030	2323	12916	7034	1398	593	292	189	1283
	atec5	319516	68162	30759	16862	11007	8025	5758	23679	12626	4647	2955	2045	1510	8090
	bbl1	105669	24802	11318	6785	4613	2556	1699	8978	6207	2455	1303	1085	863	4359
	bbl2	227085	46475	20141	7313	3979	2488	2202	14403	3472	1525	562	343	163	1646
	bbl3	386850	70763	32351	16457	9254	5644	3645	20507	21385	6311	3369	2075	1306	11725
	nbl1	124625	25587	10888	5713	3983	2624	1882	7583	5515	1618	828	404	273	1104
	nbl2	201414	64133	21131	9120	5158	3148	2128	8659	13750	8288	3169	1868	1123	12759
	nbl4	262404	63259	26482	14085	8777	6408	5046	19106	11636	4096	2260	1640	889	3808
	nbl5	454105	72875	32208	17421	11748	8482	6938	49661	13111	4447	1978	1435	692	7597
	nbl6	521761	114194	53849	28462	18355	12798	9656	46655	11647	3677	1783	1215	757	4879

"atec," "bbl," and "nbl" denote "adaptec," "bigblue," and "newblue," respectively.

two-, three-, and four-tier designs for all the benchmarks. Except for adaptec4 and bigblue3, the number of 3D nets increases as the tiers climb. The total HPWL is computed from the HPWL of the 2D nets plus the 2D projection of the 3D nets. The HPWL decreases from the two-tier design to the four-tier design for all the benchmarks except for adaptec2, bigblue3, newblue2, and newblue5, which is purely dependent on the placement locations of the instances. We obtained the placement results from the 3D placer [15] that showed a similar trend to the HPWL as well.

Table 6. Comparison of the Planar Edge Overflows and the # MIV Violations of Congestion-Aware 3D Global Routing by MR- and ARD-Based Routing

Benchmark	# Planar overflows						Avg. overflow					
	2 tier		3 tier		4 tier		2 tier		3 tier		4 tier	
	MR	ARD	MR	ARD	MR	ARD	MR	ARD	MR	ARD	MR	ARD
adaptec1	4784	1485 (0.31)	5067	1440 (0.28)	7862	1058 (0.13)	0.06	0.02	0.06	0.02	0.1	0.01
adaptec2	10001	5470 (0.55)	13656	2190 (0.16)	11685	6572 (0.56)	0.07	0.04	0.1	0.02	0.08	0.05
adaptec3	63903	3000 (0.05)	106728	21717 (0.2)	93848	19696 (0.21)	0.19	0.01	0.32	0.07	0.28	0.06
adaptec4	14890	1183 (0.08)	13344	697 (0.05)	12913	2388 (0.18)	0.05	0	0.04	0	0.04	0.01
adaptec5	94377	18256 (0.19)	100210	23917 (0.24)	87057	14116 (0.16)	0.29	0.06	0.3	0.07	0.26	0.04
bigblue1	17935	9196 (0.51)	18675	9905 (0.53)	16869	5945 (0.35)	0.23	0.12	0.23	0.12	0.21	0.07
bigblue2	0	0	25	3 (0.12)	0	7	0	0	0	0	0	0
bigblue3	535776	220542 (0.41)	585782	132427 (0.23)	435857	170041 (0.39)	1.14	0.47	1.25	0.28	0.85	0.33
newblue1	7536	1962 (0.26)	8722	3612 (0.41)	5441	2495 (0.46)	0.09	0.02	0.1	0.04	0.05	0.03
newblue2	518172	103129 (0.2)	555603	125180 (0.23)	490522	100936 (0.21)	1.32	0.26	1.41	0.32	1.15	0.24
newblue4	34098	10346 (0.3)	32655	15436 (0.47)	29504	15425 (0.52)	0.17	0.05	0.16	0.08	0.15	0.08
newblue5	132664	64738 (0.49)	246577	123946 (0.5)	1316436	79262 (0.06)	0.34	0.17	0.62	0.31	3.31	0.2
newblue6	41979	20794 (0.5)	24846	11063 (0.45)	26456	7641 (0.29)	0.09	0.04	0.05	0.02	0.06	0.02
Geo. mean	0.26		0.25		0.25							

Benchmark	# MIV violations						Max. overflow					
	2 tier		3 tier		4 tier		2 tier		3 tier		4 tier	
	MR	ARD	MR	ARD	MR	ARD	MR	ARD	MR	ARD	MR	ARD
adaptec1	1975	356 (0.18)	3202	682 (0.21)	4887	956 (0.2)	173	58	74	62	255	81
adaptec2	2857	486 (0.17)	3195	864 (0.27)	5479	1202 (0.22)	183	61	142	53	288	165
adaptec3	2392	498 (0.21)	7215	794 (0.11)	10097	1765 (0.17)	467	70	617	244	482	307
adaptec4	3222	621 (0.19)	5082	678 (0.13)	5674	605 (0.11)	142	62	106	50	136	76
adaptec5	3908	697 (0.18)	10661	1835 (0.17)	13064	2210 (0.17)	359	143	359	150	359	127
bigblue1	3053	451 (0.15)	3881	720 (0.19)	6360	1169 (0.18)	225	198	241	131	206	117
bigblue2	1846	580 (0.31)	3557	591 (0.17)	5769	856 (0.15)	0	0	14	2	0	6
bigblue3	8581	1564 (0.18)	15853	5248 (0.33)	18712	4310 (0.23)	1035	498	1042	309	1450	670
newblue1	2270	232 (0.1)	2790	432 (0.15)	4310	401 (0.09)	107	68	90	89	81	71
newblue2	12278	1563 (0.13)	18429	2628 (0.14)	22293	4145 (0.19)	797	627	827	724	794	655
newblue4	6186	1392 (0.23)	7221	1257 (0.17)	9288	1864 (0.2)	217	95	249	145	215	133
newblue5	4845	1078 (0.22)	11272	1218 (0.11)	16465	2537 (0.15)	553	301	761	262	563	359
newblue6	2173	297 (0.14)	9231	997 (0.11)	11076	1188 (0.11)	283	173	161	118	238	88
Geo. mean	0.18		0.16		0.16							

Note that all the routed designs have a minimum planar wirelength. Furthermore, the number of MIVs increases from two-tier to three-tier designs and from three-tier to four-tier designs except for the adaptec4 and bigblue3 benchmarks. The reason is that the three-tier designs have more 3D nets than the four-tier designs for these benchmarks discussed earlier. Ideally, the # MIVs should exceed the # 3D nets. Moreover, in most cases, two-tier nets dominate the set of 3D nets. The MIV distribution on each tier is therefore related to the 3D net distribution on that tier to its lower neighboring tier.

In conclusion, controlling the MIV density is crucial for the MIV violations, which are not trivial, and should be considered after minimizing the planar wirelength for the congestion-aware 3D global routing. In addition to that, by integrating our ARD into the ePlace-3D, we anticipate that the 3D placement engine will be able to manage the # 3D nets, the overall HPWL, and the vertical interconnects better.

Table 7. Comparison of the # 3D Nets, HPWL, the MIV Distribution on Different Tiers, and the Total # MIVs for the Two-, Three-, and Four-Tier Designs for All Benchmarks of Congestion-Aware 3D Global Routing

Benchmark	# 3D nets			HPWL		
	2 tier	3 tier	4 tier	2 tier	3 tier	4 tier
adaptec1	4467	5937	7968	63.74	58.41	55.63
adaptec2	6861	7403	11520	78.25	81.41	66.92
adaptec3	5503	12714	17938	153.46	148.83	133.92
adaptec4	8089	9736	9506	136.93	123.29	116.78
adaptec5	8168	19192	23751	241.89	233.62	216.19
bigblue1	5901	7114	11911	80.34	76.15	68.69
bigblue2	2405	3708	6065	110	97.95	91.18
bigblue3	20136	38273	34251	334.42	483.1	317.25
newblue1	5284	6561	8637	61.97	56.63	52.99
newblue2	16149	19526	28027	186.99	217.63	237.50
newblue4	13402	14641	20521	177.93	166.34	157.74
newblue5	9623	16046	21647	319.97	332.30	283.11
newblue6	4706	15789	19077	373.24	338.53	317.19

Benchmark	# MIVs							
	2 tier	3 tier			4 tier			
	T1 (Total)	T1	T2	Total	T1	T2	T3	Total
adaptec1	4695	3428	3484	6912 (1.47)	3429	4171	2781	10381 (2.21)
adaptec2	7144	4577	3833	8410 (1.18)	3274	7380	4149	14803 (2.07)
adaptec3	5523	10826	4108	14934 (2.70)	9735	9849	3854	23438 (4.24)
adaptec4	8114	8684	4503	13187 (1.63)	7821	3791	1305	12917 (1.59)
adaptec5	8250	17815	5466	23281 (2.82)	18339	6779	2787	27905 (3.38)
bigblue1	6153	5117	2749	7866 (1.28)	5987	6520	2223	14730 (2.39)
bigblue2	2419	2798	1547	4345 (1.80)	4322	1873	818	7013 (2.90)
bigblue3	20861	23503	21512	45015 (2.16)	13380	18799	11851	44030 (2.11)
newblue1	5499	2023	4741	6764 (1.23)	1888	4834	3528	10250 (1.86)
newblue2	16807	13539	11873	25412 (1.51)	15604	15145	7079	37828 (2.25)
newblue4	13834	9722	6425	16147 (1.17)	11455	7708	4483	23646 (1.71)
newblue5	9823	9651	8074	17725 (1.80)	11856	6881	7421	26158 (2.66)
newblue6	4771	12626	6789	19415 (4.07)	15011	4956	2431	22398 (4.69)
Geo. mean $\left(\frac{\text{\# MIVs (X-tier)}}{\text{\# MIVs (2-tier)}}\right)$		1.77			2.48			

The HPWL includes both 2D and 3D nets and measures in meters.

6.4 Proposed Approach: Five Tiers and Above

Suppose we have an ARD constructed for two, three, and four tiers. For more than four tiers, we can construct 3D routing topologies using the ARD as follows (considering a five-tier case):

First, we begin the routing topology construction by projecting all the pins in the tiers above Tier 3 onto Tier 3. Now, all the pins are located in four tiers, so we can use the ARD to find all MMRSMTs for the (projected) pin locations. Then, we move the projected pins back to their original tiers. When we expand them, we insert vertical edges to connect the projected pins and the MMRSMTs. Of course, we can project the pins in many different ways (e.g., project the pins in the tiers below Tier 1 onto Tier 1), which will help generate many different routing topologies.

One of the problems of this methodology is that it does not use planar wires in the tiers above Tier 3, which might cause routing congestion in Tier 0 through Tier 3. We can solve this problem in many different ways like the following:

- *Reassign planar edges after the vertical expansion step*: Moving a planar edge to a neighboring tier and adjusting the locations of the relevant vertical edges could result in a new 3D routing topology without any overhead. For example, if an end point of a planar edge in Tier k is connected to a vertical edge connecting Tier k and Tier (k+1), we can move the planar edge to Tier (k+1) without any overhead.
- *Run the ARD construction algorithm (Algorithm 1) for each 2D POST for a given net*: Since Algorithm 1 can be applied to any number of tiers, this methodology can find all MMRSMTs for the given pin locations (as long as the number of pins is less than 10). One problem of this methodology is the runtime overhead. Although the runtime of Algorithm 1 for a single 3D net could be small, running it for many 3D nets might need a long runtime. Thus, we can apply this methodology only to timing-critical 3D nets and the other methodologies to most of the other 3D nets.

7 CONCLUSION

Routing of 3D nets in the design of M3D IC layouts that uses tiny MIVs requires distributing 2D wires evenly, minimizing the planar wirelength and the number of MIVs simultaneously in each tier. In this article, we proposed an algorithm for building a DB of 3D POSTs that helps generate MMRSMTs swiftly to route M3D ICs optimizing 2D and 3D interconnects. The DB size is manageable for up to four-tier six-pin 3D POSTs. We applied the DB to construct timing-driven 3D routing topologies minimizing SCSLs and SCSDs. We also proposed a 3D optimal net-breaking technique and performed congestion-aware global routing on 3D designs minimizing the congestion cost. Both the applications evidenced the efficacy of using the ARD. We anticipate that the proposed algorithm and the DB of the 3D POSTs will help various VLSI CAD algorithms effectively optimize 3D IC layouts and serve as a baseline for the algorithms for better M3D IC routing.

APPENDIX

A PS ALGEBRA

A.1 Definition

A *sequence* of size n is an n-tuple, $(a_1, \ldots, a_n)$. A *2D position sequence (PS)* of size n is a sequence $(a_1, \ldots, a_n)$ of natural numbers such that $1 \leq a_k \leq n$ and $a_i \neq a_j$ if $i \neq j$. A *tier sequence (TS)* of size n is a sequence $(a_1, \ldots, a_n)$ such that $a_k \in \{0, 1, \ldots, t-1\}$ for $1 \leq k \leq n$ and there exist at least one a_k for each $i \in \{0, t-1\}$ such that $a_k = i$ (t is the number of tiers and is given). A *3D position sequence* of size n, $\Lambda = (\Gamma_1, \Gamma_2)$, is a pair of a 2D PS Γ_1 of size n and a TS Γ_2 of size n. A PS generally means a 2D PS in this article. We define $\Gamma_{(y_\pm, x_\pm)}$ and $\Gamma_{(x_\pm, y_\pm)}$ as described in Section 2.1.3. We also define $\Lambda_{(y_\pm, x_\pm, z_\pm)}$ and $\Lambda_{(x_\pm, y_\pm, z_\pm)}$ as described in Section 3.2.2.

A.2 Operations and Functions

We define the addition operation for two sequences $\Gamma_1 = (a_1, \ldots, a_n)$ and $\Gamma_2 = (b_1, \ldots, b_n)$ as follows:

$$\Gamma_1 + \Gamma_2 = (a_1 + b_1, \ldots, a_n + b_n). \tag{11}$$

We also define the inversion operation for a sequence $\Gamma = (a_1, \ldots, a_n)$ as follows:

$$\overline{\Gamma} = (a_n, \ldots, a_1). \tag{12}$$

Notice that $\Gamma_1 + \Gamma_2 = \Gamma_2 + \Gamma_1$ and $\overline{\overline{\Gamma}} = \Gamma$. The following function maps a_i in $\Gamma = (a_1, \ldots, a_n)$ to i:

$$\phi(a_i) = i. \tag{13}$$

Table 8. Congruence Rules

		To							
		(y_+, x_+)	(y_+, x_-)	(y_-, x_+)	(y_-, x_-)	(x_+, y_+)	(x_+, y_-)	(x_-, y_+)	(x_-, y_-)
From	(y_+, x_+)	-	$(\overline{a_1}, \ldots, \overline{a_n})$	$(a_n, \ldots, a_1)$	$(\overline{a_n}, \ldots, \overline{a_1})$	$(b_1, \ldots, b_n)$	$(\overline{b_1}, \ldots, \overline{b_n})$	$(b_n, \ldots, b_1)$	$(\overline{b_n}, \ldots, \overline{b_1})$
	(y_+, x_-)	$(\overline{a_1}, \ldots, \overline{a_n})$	-	$(\overline{a_n}, \ldots, \overline{a_1})$	$(a_n, \ldots, a_1)$	$(b_n, \ldots, b_1)$	$(\overline{b_n}, \ldots, \overline{b_1})$	$(b_1, \ldots, b_n)$	$(\overline{b_1}, \ldots, \overline{b_n})$
	(y_-, x_+)	$(a_n, \ldots, a_1)$	$(\overline{a_n}, \ldots, \overline{a_1})$	-	$(\overline{a_1}, \ldots, \overline{a_n})$	$(\overline{b_1}, \ldots, \overline{b_n})$	$(b_1, \ldots, b_n)$	$(\overline{b_n}, \ldots, \overline{b_1})$	$(b_n, \ldots, b_1)$
	(y_-, x_-)	$(\overline{a_n}, \ldots, \overline{a_1})$	$(a_n, \ldots, a_1)$	$(\overline{a_1}, \ldots, \overline{a_n})$	-	$(\overline{b_n}, \ldots, \overline{b_1})$	$(b_n, \ldots, b_1)$	$(\overline{b_1}, \ldots, \overline{b_n})$	$(b_1, \ldots, b_n)$
	(x_+, y_+)	$(b_1, \ldots, b_n)$	$(\overline{b_1}, \ldots, \overline{b_n})$	$(b_n, \ldots, b_1)$	$(\overline{b_n}, \ldots, \overline{b_1})$	-	$(\overline{a_1}, \ldots, \overline{a_n})$	$(a_n, \ldots, a_1)$	$(\overline{a_n}, \ldots, \overline{a_1})$
	(x_+, y_-)	$(b_n, \ldots, b_1)$	$(\overline{b_n}, \ldots, \overline{b_1})$	$(b_1, \ldots, b_n)$	$(\overline{b_1}, \ldots, \overline{b_n})$	$(\overline{a_1}, \ldots, \overline{a_n})$	-	$(\overline{a_n}, \ldots, \overline{a_1})$	$(a_n, \ldots, a_1)$
	(x_-, y_+)	$(\overline{b_1}, \ldots, \overline{b_n})$	$(b_1, \ldots, b_n)$	$(\overline{b_n}, \ldots, \overline{b_1})$	$(b_n, \ldots, b_1)$	$(a_n, \ldots, a_1)$	$(\overline{a_n}, \ldots, \overline{a_1})$	-	$(\overline{a_1}, \ldots, \overline{a_n})$
	(x_-, y_-)	$(\overline{b_n}, \ldots, \overline{b_1})$	$(b_n, \ldots, b_1)$	$(\overline{b_1}, \ldots, \overline{b_n})$	$(b_1, \ldots, b_n)$	$(\overline{a_n}, \ldots, \overline{a_1})$	$(a_n, \ldots, a_1)$	$(\overline{a_1}, \ldots, \overline{a_n})$	-

From: $\Gamma_{(s,r)} = (a_1, \ldots, a_n)$. $\phi(\Gamma_{(s,r)}) = (b_1, \ldots, b_n)$. $\overline{k} = (n+1) - k$.

Applying ϕ to a PS Γ results in a new sequence as follows:

$$\phi(\Gamma) = (\phi(1), \ldots, \phi(n)). \tag{14}$$

For example, suppose $\Gamma = (31542)$. Then, $\phi(a_1) = \phi(3) = 1$, $\phi(a_2) = \phi(1) = 2$, $\phi(a_3) = \phi(5) = 3$, $\phi(a_4) = \phi(4) = 4$, and $\phi(a_5) = \phi(2) = 5$. Then, $\phi(\Gamma) = (25143)$. If $i \neq j$, then $a_i \neq a_j$ because Γ is a PS. In addition, $1 \leq \phi(a_i) \leq n$, so $\phi(\Gamma)$ is also a PS. For a 3D PS $\Lambda = (\Gamma_1, \Gamma_2)$, we denote its 2D PS and TS by PS(Λ) and TS(Λ), respectively.

A.3 Properties of 2D PSs

PSs have the following properties:

$$\Gamma_{(s,x_+)} + \Gamma_{(s,x_-)} = (n+1, \ldots, n+1), \tag{15}$$

$$\Gamma_{(s,y_+)} + \Gamma_{(s,y_-)} = (n+1, \ldots, n+1), \tag{16}$$

$$\overline{\Gamma_{(x_+,r)}} = \Gamma_{(x_-,r)}, \tag{17}$$

$$\overline{\Gamma_{(y_+,r)}} = \Gamma_{(y_-,r)}. \tag{18}$$

For the pins in Figure 2(a), for example, $\Gamma_{(y_+,x_+)} + \Gamma_{(y_+,x_-)} = (66666)$ and $\overline{\Gamma_{(y_+,x_+)}} = \overline{(31542)} = (24513) = \Gamma_{(y_-,x_+)}$. PSs also have the following properties:

$$\phi(\Gamma_{(s,r)}) = \Gamma_{(r,s)}. \tag{19}$$

For example, $\phi(\Gamma_{(y_+,x_+)} = (31542)) = (25143) = \Gamma_{(x_+,y_+)}$, $\phi(\Gamma_{(y_+,x_-)} = (35124)) = (34152) = \Gamma_{(x_-,y_+)}$, $\phi(\Gamma_{(y_-,x_+)} = (24513)) = (41523) = \Gamma_{(x_+,y_-)}$, and $\phi(\Gamma_{(y_-,x_-)} = (42153)) = (32514) = \Gamma_{(x_-,y_-)}$. Using the properties shown previously, we can generate seven PSs from a PS as shown in Table 8.

A.4 Properties of 3D PSs

First of all, the 2D PSs of 3D PSs have the properties shown in (15) through (19). The TSs of 3D PSs have the following properties:

$$\mathrm{TS}(\Lambda_{(s,r,z_+)}) + \mathrm{TS}(\Lambda_{(s,r,z_-)}) = (t-1, \ldots, t-1), \tag{20}$$

$$\mathrm{TS}(\Lambda_{(s,x_+,w)}) = \mathrm{TS}(\Lambda_{(s,x_-,w)}), \tag{21}$$

$$\mathrm{TS}(\Lambda_{(s,y_+,w)}) = \mathrm{TS}(\Lambda_{(s,y_-,w)}), \tag{22}$$

$$\overline{\mathrm{TS}(\Lambda_{(x_+,r,w)})} = \mathrm{TS}(\Lambda_{(x_-,r,w)}), \tag{23}$$

$$\overline{\mathrm{TS}(\Lambda_{(y_+,r,w)})} = \mathrm{TS}(\Lambda_{(y_-,r,w)}). \tag{24}$$

For the pins in Figure 8, for example, $\mathrm{TS}(\Lambda_{(y_+,x_+,z_+)}) + \mathrm{TS}(\Lambda_{(y_+,x_+,z_-)}) = (11111)$, $\mathrm{TS}(\Lambda_{(y_+,x_+,z_-)}) = (10101) = \mathrm{TS}(\Lambda_{(y_+,x_-,z_-)})$, and $\overline{\mathrm{TS}(\Lambda_{(x_+,y_-,z_-)})} = \overline{(01101)} = (10110) = \mathrm{TS}(\Lambda_{(x_-,y_-,z_-)})$. Table 8

and the properties of TSs in (20) through (24) can be used for the congruence mapping of 2D and 3D PSs.

REFERENCES

[1] Yiting Chen and Dae Hyun Kim. 2017. A legalization algorithm for multi-tier gate-level monolithic three-dimensional integrated circuits. In *Proceedings of the 2017 18th International Symposium on Quality Electronic Design (ISQED'17)*. 277–282. https://doi.org/10.1109/ISQED.2017.7918328

[2] Chris Chu and Yiu-Chung Wong. 2008. FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 1 (2008), 70–83. https://doi.org/10.1109/TCAD.2007.907068

[3] Bhargava Gopireddy and Josep Torrellas. 2019. Designing vertical processors in monolithic 3D. In *Proceedings of the 46th International Symposium on Computer Architecture (ISCA'19)*. ACM, New York, NY, 643–656. https://doi.org/10.1145/3307650.3322233

[4] Dae Hyun Kim, Krit Athikulwongse, and Sung Kyu Lim. 2013. Study of through-silicon-via impact on the 3-D stacked IC layout. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 5 (2013), 862–874. https://doi.org/10.1109/TVLSI.2012.2201760

[5] Dae Hyun Kim, Rasit Onur Topaloglu, and Sung Kyu Lim. 2012. Block-level 3D IC design with through-silicon-via planning. In *Proceedings of the 17th Asia and South Pacific Design Automation Conference*. 335–340. https://doi.org/10.1109/ASPDAC.2012.6164969

[6] Bon Woong Ku, Kyungwook Chang, and Sung Kyu Lim. 2018. Compact-2D: A physical design methodology to build commercial-quality face-to-face-bonded 3D ICs. In *Proceedings of the 2018 International Symposium on Physical Design (ISPD'18)*. ACM, New York, NY, 90–97. https://doi.org/10.1145/3177540.3178244

[7] Young-Joon Lee and Sung Kyu Lim. 2013. Ultrahigh density logic designs using monolithic 3-D integration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 12 (2013), 1892–1905. https://doi.org/10.1109/TCAD.2013.2273986

[8] Chung-Wei Lin, Shih-Lun Huang, Kai-Chi Hsu, Meng-Xiang Lee, and Yao-Wen Chang. 2008. Multilayer obstacle-avoiding rectilinear Steiner tree construction based on spanning graphs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 11 (2008), 2007–2016. https://doi.org/10.1109/TCAD.2008.2006095

[9] Sheng-En David Lin and Dae Hyun Kim. 2018. Construction of all rectilinear Steiner minimum trees on the Hanan grid. In *Proceedings of the 2018 International Symposium on Physical Design (ISPD'18)*. ACM, New York, NY, 18–25. https://doi.org/10.1145/3177540.3178240

[10] Sheng-En David Lin and Dae Hyun Kim. 2019. Construction of all multilayer monolithic rectilinear Steiner minimum trees on the 3D Hanan grid for monolithic 3D IC routing. In *Proceedings of the 2019 International Symposium on Physical Design (ISPD'19)*. ACM, New York, NY, 57–64. https://doi.org/10.1145/3299902.3309749

[11] Sheng-En David Lin and Dae Hyun Kim. 2019. Wire length characteristics of multi-tier gate-level monolithic 3D ICs. *IEEE Transactions on Emerging Topics in Computing* 7, 2 (2019), 301–310. https://doi.org/10.1109/TETC.2016.2630064

[12] Sheng-En David Lin and Dae Hyun Kim. 2020. Construction of all rectilinear Steiner minimum trees on the Hanan grid and its applications to VLSI design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 6 (2020), 1165–1176. https://doi.org/10.1109/TCAD.2019.2917896

[13] Sheng-En David Lin, Partha Pratim Pande, and Dae Hyun Kim. 2016. Optimization of dynamic power consumption in multi-tier gate-level monolithic 3D ICs. In *Proceedings of the 2016 17th International Symposium on Quality Electronic Design (ISQED'16)*. 29–34. https://doi.org/10.1109/ISQED.2016.7479152

[14] Chih-Hung Liu, Chun-Xun Lin, I-Che Chen, D. T. Lee, and Ting-Chi Wang. 2014. Efficient multilayer obstacle-avoiding rectilinear Steiner tree construction based on geometric reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 12 (2014), 1928–1941. https://doi.org/10.1109/TCAD.2014.2363390

[15] Jingwei Lu, Hao Zhuang, Ilgweon Kang, Pengwen Chen, and Chung-Kuan Cheng. 2016. EPlace-3D: Electrostatics based placement for 3D-ICs. In *Proceedings of the 2016 International Symposium on Physical Design (ISPD'16)*. ACM, New York, NY, 11–18. https://doi.org/10.1145/2872334.2872361

[16] Gi-Joon Nam. 2006. ISPD 2006 placement contest: Benchmark suite and results. In *Proceedings of the 2006 International Symposium on Physical Design (ISPD'06)*. ACM, New York, NY, 167. https://doi.org/10.1145/1123008.1123042

[17] Gi-Joon Nam, Charles J. Alpert, Paul Villarrubia, Bruce Winter, and Mehmet Yildiz. 2005. The ISPD2005 placement contest and benchmark suite. In *Proceedings of the 2005 International Symposium on Physical Design (ISPD'05)*. ACM, New York, NY, 216–220. https://doi.org/10.1145/1055137.1055182

[18] Shreepad Panth, Kambiz Samadi, Yang Du, and Sung Kyu Lim. 2013. High-density integration of functional modules using monolithic 3D-IC technology. In *Proceedings of the 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC'13)*. 681–686. https://doi.org/10.1109/ASPDAC.2013.6509679

[19] Shreepad Panth, Kambiz Samadi, Yang Du, and Sung Kyu Lim. 2014. Design and CAD methodologies for low power gate-level monolithic 3D ICs. In *Proceedings of the 2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED'14)*. 171–176. https://doi.org/10.1145/2627369.2627642

[20] Shreepad Panth, Kambiz Samadi, Yang Du, and Sung Kyu Lim. 2015. Placement-driven partitioning for congestion mitigation in monolithic 3D IC designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 4 (2015), 540–553. https://doi.org/10.1109/TCAD.2014.2387827

[21] Shreepad Panth, Kambiz Samadi, Yang Du, and Sung Kyu Lim. 2017. Shrunk-2-D: A physical design methodology to build commercial-quality monolithic 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36, 10 (2017), 1716–1724. https://doi.org/10.1109/TCAD.2017.2648839

[22] Sandeep Kumar Samal, Deepak Nayak, Motoi Ichihashi, Srinivasa Banna, and Sung Kyu Lim. 2016. Monolithic 3D IC vs. TSV-based 3D IC in 14nm FinFET technology. In *Proceedings of the 2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S'16)*. 1–2. https://doi.org/10.1109/S3S.2016.7804405

Received 4 March 2023; revised 11 July 2023; accepted 15 September 2023