A Non-Slicing 3-D Floorplan Representation for Monolithic 3-D IC Design

Shantonu Das and Dae Hyun Kim
School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA
Email: shantonu.das@wsu.edu, daehyun@eecs.wsu.edu

***Abstract*—In this paper, we propose a non-slicing 3-D floorplan representation to design block-level monolithic 3-D ICs. The new 3-D floorplan representation applied to simulated annealing-based optimization achieves smaller volume, shorter wire length, and lower dynamic power consumption than the Sequence Triple, Sequence Quintuple, and Slicing Tree 3-D floorplanning representations.**

I. Introduction

Monolithic three-dimensional integration stacks multiple ultra-thin silicon tiers (dies) and connects transistors in different tiers through monolithic inter-layer vias (MIVs). MIVs are similar to through-silicon vias (TSVs) because both of them are fabricated in silicon, made of conducting material, and used for inter-tier electrical connections. However, the typical width of an MIV is around 0.1um, whereas that of a TSV is several micrometers. Thus, monolithic 3-D integration provides the most fine-grained 3-D integration, thereby enabling shorter wire length, lower power consumption, and higher performance than TSV-based 3-D integration [1]–[4].

One of the advantages of the monolithic 3-D integration is that functional blocks (either logic or memory) can be designed in multiple tiers. As shown in [5], redesigning a small block in a TSV-based 3-D integrated circuit (IC) provides almost no benefit or even has worse characteristics (longer wire length, higher power consumption, or lower performance) than its two-dimensional (2-D) counterpart because of the area overhead caused by TSV insertion. Thus, redesigning only large blocks can benefit from the TSV-based 3-D integration. On the contrary, redesigning a small block by monolithic 3-D integration still provides benefits because MIV insertion causes almost no area overhead [5]. Therefore, block-level 3-D IC design using monolithic 3-D integration enables the use of both 2-D and 3-D blocks. Design of block-level monolithic 3-D IC layouts, however, requires sophisticated algorithms for 3-D floorplanning to effectively pack the 2-D and 3-D blocks and minimize the wire length and power consumption.

In this paper, we propose a non-slicing 3-D floorplan representation, so-called single matrix multiple sequences (SMMS) representation, to design low-power block-level monolithic 3-D IC layouts. SMMS handles x-, y-, and z-coordinates separately, thereby providing larger solution space than other 3-D floorplan representations. In addition, SMMS does not require feasibility checks because each solution corresponds to a feasible floorplan. SMMS can also be applied to any-dimensional floorplanning problems. Simulation results in Section IV show that SMMS can produce high-quality 3-D floorplans.

II. Preliminaries

In this section, we explain several terminologies used for floorplanning, show a motivation for the development of a new 3-D floorplan representation, and review several 3-D floorplan representations proposed in the literature. Table I summarizes some properties of the 3-D floorplan representations.

A. Terminologies

Floorplan representations are evaluated based on the solution space size, evaluation time, feasibility, and representation dependencies. A solution is a floorplan and the solution space size of a floorplan representation is the total number of floorplans that can be represented by the floorplan representation. In general, a floorplan representation that has a larger solution space is better than other floorplan representations that have smaller solution space. However, some floorplan representations have many redundancies, which means that several solutions correspond to the same floorplan.

A solution is *infeasible* if it cannot be converted into a legal floorplan. For example, if a solution contains a condition such as "A is to the left of B and B is to the left of A", it is infeasible. If a floorplan representation generates infeasible solutions, it should perform a feasibility check for each solution, otherwise it will waste runtime to evaluate infeasible solutions.

A floorplan representation has *dependencies* if deciding one of the coordinates of a block is dependent on at least one of the other coordinates of the block. For example, some floorplan representations forbid having both x- and z-directional relations at the same time between two blocks. Thus, if two blocks have a relation (e.g., A is to the left of B) along an axis, they cannot have other relations (e.g., A is above B) along all the other axes. If there exist dependencies in a floorplan representation, the size of the solution space of the floorplan representation decreases.

B. Motivation

Most of the 3-D floorplan representations and algorithms focus on the minimization of the total volume of a given design. However, minimization of the total wire length and dynamic power consumption (sum of weighted wire lengths) is also crucial in the design of block-level 3-D ICs. Unfortunately, volume minimization does not necessarily lead to wire length and power minimization. Figure 1 shows an example in which a net connects two blocks b_1 and b_2 and a pin p_1. The half-perimeter wire length (HPWL) of the floorplan in

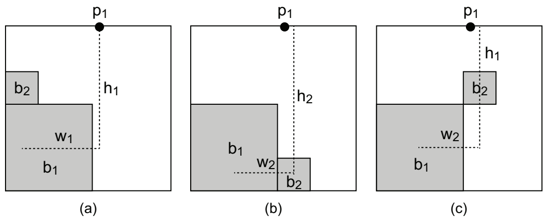

Fig. 1. Wire length of a 2-D floorplan. $w_2 < w_1$ and $h_1 < h_2$. (a) Wire length = $w_1 + h_1$. (b) Wire length = $w_2 + h_2$. (c) Wire length = $w_2 + h_1$.

TABLE I
COMPARISON OF 3-D FLOORPLAN REPRESENTATIONS. "SPACE" IS THE SIZE OF THE SOLUTION SPACE, "EVAL. TIME" IS THE EVALUATION TIME, "FEAS." IS WHETHER EACH ALGORITHM NEEDS FEASIBILITY CHECKS FOR SOLUTION PERTURBATIONS, AND "DEP." IS THE DEPENDENCY AMONG THE X-, Y-, AND Z-COORDINATES.

Representation	Space	Eval. time	Feas.	Dep.
Seq. Triple [6]	$O((n!)^3)$	$O(n^2)$	No	Yes
Seq. Quintuple [6]	$O((n!)^5)$	$O(n^2)$	No	Yes
3D CBL [7]	$O(n!48^n)$	$O(n)$	No	Yes
Slicing Tree [8]	$O(6^n(n!)^2)$	$O(n)$	Yes	Yes
3D-subTCG [9]	$O((n!)^3)$	$O(n^2)$	Yes	Yes
Tree + Seq^2 [10]	$O((n+1)^n(n!)^2)$	$O(n^2)$	No	Yes
DTS [11]	$O(n!(n+1)^{2n})$	$O(n^2)$	Yes	Yes
T-Tree [12]	$O(n!\frac{3^{3n}}{2^{2n}n^{1.5}})$	$O(n^2)$	Yes	Yes
SMMS (This work)	$O(7^{\frac{n(n-1)}{2}}(n!)^3)$	$O(n^2)$	**No**	**No**

Figure 1(a) is $w_1 + h_1$. In this floorplan, the relation between b_1 and b_2 is "b_2 is above b_1", which can be formulated in all the floorplan representations. Similarly, the HPWL of the floorplan in Figure 1(b) is $w_2 + h_2$. The relation between b_1 and b_2 in the figure is "b_2 is to the right of b_1". The HPWL of the floorplan shown in Figure 1(c) is $w_2 + h_1$, which is the shortest among the three floorplans. The relation between b_1 and b_2 in this case is "b_2 is above and to the right of b_1". However, most of the 3-D floorplan representations do not formulate this relation because they constrain only the x-, y-, or z-coordinate for a pair of blocks. For example, the Sequence Triple representation proposed in [6] determines a relation between a pair of blocks along only the x-, y-, or z-axis, but not along two or three axes at the same time. In addition, some floorplan representations require a feasibility check after perturbing a solution because some of their solutions are physically infeasible, especially due to cyclic relations such as "b_1 is to the left of b_2 and b_2 is to the left of b_1". Although some of the 3-D floorplan representations could be extended to process x-, y-, and z-coordinates separately, they might still need feasibility checks. The 3-D floorplan representation we propose in this paper handles the x-, y-, and z-coordinates separately and does not require feasibility checks.

C. 3-D Floorplan Representations

Sequence Triple (ST) proposed in [6] is an extension of the Sequence Pair representation [13]. ST uses three sequences of blocks and the relation between a pair of blocks is determined by their relative locations in the three sequences. The total number of combinations of the relative locations between two blocks in the three sequences is eight, but there are only six relations along the three axes, so two of the eight combinations are redundantly mapped into two of the six relations.

Sequence Quintuple (SQ) proposed in [6] uses five sequences of blocks and each solution corresponds to a unique 3-D floorplan. The first two sequences and the next two sequences are used as sequence pairs to determine the x- and y-directional relations between two blocks, respectively. The fifth sequence is used to determine the z-directional relation between two blocks. However, the fifth sequence is effective between two blocks only when there is no x- and y-directional relation between them.

3D Corner Block List (CBL) proposed in [7] uses a three-element triplet (S, L, T) to represent 3-D floorplans. S has a list of blocks, L has a list of orientations, and T has a list of junction information. On the other hand, 3-D slicing floorplan proposed in [8] uses a binary tree to construct 3-D slicing floorplans. Both of them have dependencies among x-, y-, and z-coordinates. 3-D Transitive Closure subGraph (3D-subTCG) proposed in [9] uses three transitive graphs to determine the relation between two blocks along the three axes. However, a 3D-subTCG should satisfy several feasibility conditions. For example, each transitive graph in the 3D-subTCG should be acyclic; otherwise, physical implementation of the graph will fail. Thus, 3D-subTCG requires feasibility checks for some solution perturbations.

The 3-D floorplan representation proposed in [10] uses a labeled tree, a permutation sequence, and a number sequence. This single-tree dual-sequence representation always tries to minimize the volume, so it cannot effectively minimize the wire length if the minimum wire length is obtained from a non-smallest-volume floorplan.

Double tree and sequence (DTS) based 3-D floorplanning proposed in [11] uses an x-tree and a y-tree to determine the x- and y-directional relations between two blocks. DTS also uses a sequence to determine the z-directional relation between two blocks when they overlap in a plane. Thus, the z-coordinates of the blocks are dependent on the x- and y-coordinates of the blocks.

T-Tree proposed in [12] uses a tree structure in which a node of a block has three child nodes of blocks by which the relation between each child node (block) and its parent node (block) is uniquely determined. However, some of the solution perturbations such as move and swap operations need feasibility checks. The 3-D floorplanning algorithms in [14]–[16] use the sequence pair representation for each tier, so they do not handle 3-D blocks.

The 3-D floorplan representation (SMMS) we propose does not require feasibility checks because each solution corresponds to a floorplan. In addition, the x-, y-, and z-coordinates of each block are determined separately, so the representation can minimize the volume, wire length, and power effectively.

III. SINGLE MATRIX MULTIPLE SEQUENCES 3-D FLOORPLAN REPRESENTATION

In this section, we propose a new floorplan representation for multitier block-level monolithic 3-D ICs that can handle 3-D blocks. Table II shows notations used in this paper.

TABLE II
NOTATIONS USED IN THIS PAPER

b_i	Block i
(x_i, y_i, z_i)	The coordinate of the bottom-left-front corner of b_i
l_{x_i}	The x-directional length of b_i
l_{y_i}	The y-directional length of b_i
l_{z_i}	The z-directional length of b_i

A. Single Matrix Multiple Sequences

Two blocks in a 3-D floorplan always have at least one of the X, Y, and Z relations as follows:

Definition 1: If $\boldsymbol{b_i X b_j}$ holds, $x_i + l_{x_i} \leq x_j$ is satisfied. Similarly, $y_i + l_{y_i} \leq y_j$ and $z_i + l_{z_i} \leq z_j$ are satisfied if $\boldsymbol{b_i Y b_j}$ and $\boldsymbol{b_i Z b_j}$ hold, respectively.

We also define a relation matrix to store relations among the blocks as follows:

Definition 2: A **relation matrix** M_R is an $n \times n$ matrix where n is the total number of blocks. The element $m_{i,j}$ at the i-th row and the j-th column shows the relation between block b_i and block b_j. $m_{i,j}$ can be X, Y, Z, XY, YZ, ZX, or XYZ. If $m_{i,j}$ is X, either $b_i X b_j$ or $b_j X b_i$ holds. Y and Z are defined in a similar way. If $m_{i,j}$ is XY, either $b_i X b_j$ or $b_j X b_i$ holds and either $b_i Y b_j$ or $b_j Y b_i$ holds at the same time. YZ, ZX, and XYZ are defined similarly.

To represent the relations among the blocks, we use the relation matrix defined above. Since $m_{i,j}$ has at least one relation, any pair of two blocks always has at least one relation, which is used to avoid an overlap between the two blocks. However, the elements in the relation matrix do not show the orders of the blocks. For example, if $m_{i,j}$ is Z, either $b_i Z b_j$ or $b_j Z b_i$ holds, but it does not determine which one is chosen. To determine the order, we use sequences of blocks. The following defines a sequence of blocks:

Definition 3: A **sequence** of blocks is an ordered list $S =< b_{i_1}, ..., b_{i_n} >$ of a set of n blocks $B = \{b_1, ..., b_n\}$ where each block appears only once in S.

For example, $S_1 =< b_5, b_3, b_4, b_1, b_2 >$ is a sequence of blocks for $B = \{b_1, b_2, b_3, b_4, b_5\}$, but $S_2 =< b_2, b_3, b_1, b_5 >$ and $S_3 =< b_1, b_4, b_3, b_5, b_2, b_1 >$ are not sequences for B because S_2 does not contain b_4 and S_3 has b_1 twice.

The single-matrix multiple-sequences (SMMS) 3-D floorplan representation we propose has three sequences as follows:

Definition 4: An $\boldsymbol{X}$ **sequence** $\boldsymbol{S_X}$ for a set of n blocks $B = \{b_1, b_2, ..., b_n\}$ is a sequence of B having relations among the blocks along the x-axis. If block b_i appears before b_j in S_X, either $b_i + l_{x_i} \leq b_j$ holds or they have no x-directional relation. Y and Z sequences are defined similarly. If block b_i appears before b_j in S_Y, either $y_i + l_{y_i} \leq y_j$ holds or they have no y-directional relation. If block b_i appears before b_j in S_Z, either $z_i + l_{z_i} \leq z_j$ holds or they have no z-directional relation.

Combining the relation matrix M_R and the three sequences S_X, S_Y, and S_Z produces a unique relation between any two blocks. For example, suppose $m_{i,j}$ is XY, $S_X =< ..., b_i, ..., b_j, ... >$, $S_Y =< ..., b_j, ..., b_i, ... >$, and $S_Z =< ..., b_i, ..., b_j, ... >$. In this case, b_i and b_j have two relations, one along the x-axis and the other along the y-axis. Since b_i appears before b_j in S_X, $x_i + l_{x_i} \leq x_j$ holds. Similarly, b_j appears before b_i in S_Y, $l_j + l_{y_j} \leq y_i$ holds. However, $m_{i,j}$ does not contain Z, so we ignore the z-directional relation between b_i and b_j in S_Z.

We call this floorplan representation the **Single Matrix Multiple Sequences (SMMS)** representation because it consists of a single relation matrix and multiple sequences. We can also apply SMMS to a 2-D floorplanning by having only two sequences S_X and S_Y and allowing $m_{i,j}$ to have only X, Y, and XY. In general, SMMS can be extended to a k-dimensional floorplanning if all the blocks have k orthogonal coordinates. In this case, the k-dimensional floorplan representation has k sequences $S_1, ..., S_k$ and a relation matrix M_R in which $m_{i,j}$ is an OR-ed value of $\{r_1, ..., r_k\}$ where r_p is the relation along the p-th axis.

B. Properties of SMMS

The SMMS representation has the following properties.

Property 1 (Symmetric): M_R is always symmetric, so we use only the upper triangular elements in M_R to evaluate an SMMS solution.

Property 2 (Acyclic): A floorplan has an x-directional cycle if there is a sub-sequence of blocks $S_C =< b_{i_1}, b_{i_2}, ..., b_{i_k} >$ such that $x_{i_1} + l_{x_{i_1}} \leq x_{i_2}, x_{i_2} + l_{x_{i_2}} \leq x_{i_3}, ..., x_{i_k} + l_{x_{i_k}} \leq x_{i_1}$, which is physically infeasible. y- and z-directional cycles are defined in a similar way. If a floorplan solution does not have any cycle, it is called *acyclic*. All SMMS solutions are acyclic because each block appears exactly once in each sequence.

Property 3 (Solution space): There are $n(n-1)/2$ block pairs for given n blocks and each block pair has a relation among $\{X, Y, Z, XY, YZ, ZX, XYZ\}$, so there are total $7^{n(n-1)/2}$ combinations of the relations. In addition, each sequence has n elements, so the total number of combinations of the blocks in each sequence is $n!$. Since there are three sequences in an SMMS solution, the total number of combinations of the blocks in the three sequences is $(n!)^3$. Thus, the total number of SMMS solutions for 3-D floorplanning is $7^{\frac{n(n-1)}{2}} \cdot (n!)^3$ for n blocks.

Property 4 (P-admissible): First, the solution space of SMMS is finite as shown in Property 3. Second, every solution of SMMS is feasible. Third, realization of an SMMS solution takes polynomial time as shown in the next section. Fourth, there exists an SMMS solution corresponding to an optimal solution because it is always possible to convert a given floorplan to an SMMS solution. Thus, the SMMS representation is P-admissible.

C. From an SMMS Solution to a 3-D Floorplan

Algorithm 1 shows a function to evaluate the x-coordinates of the blocks in B. We evaluate the y- and z- coordinates in a similar way using S_Y and S_Z instead of S_X, respectively. The algorithm first constructs a directed graph G and inserts a head node and a tail node into G. Then, for each block b_j, it inserts a node n_j corresponding to b_j and creates an edge from the head node to n_j and an edge from n_j to the tail node. Then, we check $m_{j,k}$ for each block pair and insert an edge from b_j to b_k if $m_{j,k}$ has X and b_j appears before b_k in S_X or from b_k to b_j if $m_{j,k}$ has X and b_k appears before b_j

Algorithm 1: Evaluation of the x-coordinates for an SMMS representation.

Input: $M_R, S_X = < b_{i_1}, ..., b_{i_n} >$ for $B = \{b_1, ..., b_n\}$
Output: The x-coordinates of all the blocks in B

Declare a directed graph G;
G.insert_node (head); // n_h is the head node.
G.insert_node (tail); // n_t is the tail node.
for $j = 1$ to n **do**
  G.insert_node (b_j); // node n_j is for b_j
  G.insert_edge (n_h, n_j); // $e_{h,j}$ = $n_h \to n_j$
  G.insert_edge (n_j, n_t); // $e_{j,t}$ = $n_j \to n_t$
end for
for $j = 1$ to n **do**
  // for block b_{i_j} in S_X
  for $k = j + 1$ to n **do**
    // for block b_{i_k} in S_X
    if m_{i_j,i_k} in M_R has X **then**
      G.insert_edge (b_{i_j}, b_{i_k});
    end if
  end for
end for
G.recursive_traversal (tail);

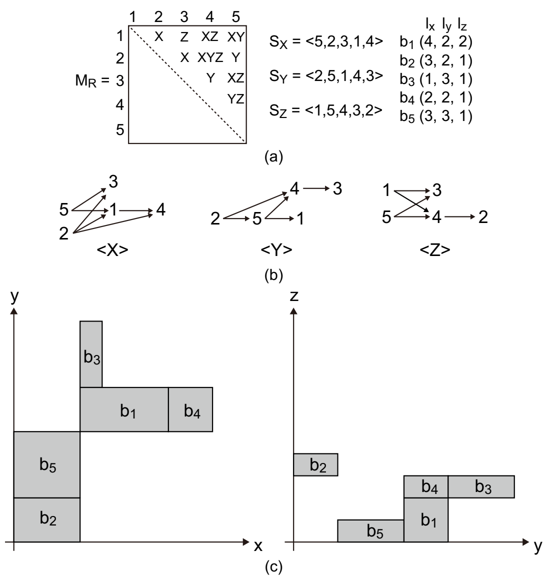

Fig. 2. An example of the SMMS representation. (a) An SMMS solution. (b) Its constraint graphs. (c) The 3-D floorplan corresponding to the SMMS solution in (a).

in S_X. We call G a *constraint graph*. The recursive_traversal function finds the longest length from the head node to each block node by recursive traversal starting from the tail node. Figure 2 shows an example. The evaluation time of a constraint graph is $O(n^2)$.

D. From a 3-D Floorplan to an SMMS Solution

We translate a given 3-D floorplan into an SMMS solution as follows. We first prepare three empty directed graphs, G_X, G_Y, and G_Z for x-, y-, and z-directional relations, respectively. Then, for each pair of blocks b_i and b_j, we update M_R, G_X, G_Y, and G_Z based on their relative locations. Once G_X, G_Y, and G_Z are constructed, we iteratively find the nodes that have no incoming edges in each graph and insert them into the end of the sequence corresponding to the graph. Whenever we add blocks to each graph, we remove the outgoing edges of the blocks. This procedure constructs a sequence of blocks from each graph.

E. Solution Perturbation

We use the following perturbations for the simulated annealing-based optimization using the SMMS representation.

- Change an element in the relation matrix: Choose two blocks randomly and change their relationship in the relation matrix.
- Swap two blocks in a sequence: Choose two blocks b_i and b_j and a sequence S_t randomly. Then, swap the locations of the two blocks in S_t.
- Swap two blocks in two sequences: Choose two blocks b_i and b_j and two sequences S_t and S_p randomly. Then, swap the locations of the two blocks in S_t and S_p.
- Swap two blocks in all the sequences: Choose two blocks b_i and b_j randomly. Then, swap the locations of the two blocks in S_X, S_Y, and S_Z.
- Resize a block in 2-D: Choose a block b_i randomly and resize it in 2-D. Thus, its z-directional length does not change, but its x- and y-directional lengths change.
- Resize a block in 3-D: Choose a block b_i randomly and resize it in 3-D. We first change its z-directional length because it should be an integer. Then, we change its x- and y-directional lengths.

F. Reduction of the Evaluation Time

The time complexity of the evaluation of each constraint graph is $O(n^2)$. However, we reduce the evaluation time as follows. First, changing an element in the relation matrix in the solution perturbation requires selective evaluation of the constraint graphs. For instance, if $m_{i,j}$ is XY and changed to XYZ, we do not need to re-evaluate the x- and y-directional constraint graphs because only the z-coordinates of the blocks are affected by adding the Z relation between b_i and b_j. Second, the perturbation methods except resizing requires reconstruction of the constraint graphs. However, once we construct a constraint graph, we can incrementally update the constraint graph whenever we perturb the current solution. Changing an element $m_{i,j}$ in the relation matrix adds or removes maximum three edges between b_i and b_j. Thus, the runtime for updating the constraint graphs for changing $m_{i,j}$ is $O(1)$. Swapping two blocks b_i and b_j in a sequence in which b_i appears before b_j requires 1) reversing the edges from b_i to the blocks between b_i and b_j, 2) reversing the edges from the blocks between b_i and b_j to b_j, and 3) reversing the edge from b_i to b_j, all in the constraint graph corresponding to the sequence. The complexity of updating a constraint graph for swapping two blocks is $O(n)$ in the worst case.

IV. Simulation Results

In this section, we present 3-D floorplanning simulation results and detailed analysis.

TABLE III

COMPARISON OF THE 3-D FLOORPLANNING ALGORITHMS. SEQT: SEQUENCE TRIPLE. SEQQ: SEQUENCE QUINTUPLE. SLIT: SLICING TREE. SMMS IS THE PROPOSED ALGORITHM. V: 3-D FLOORPLAN VOLUME. L: WIRE LENGTH. P: DYNAMIC POWER CONSUMPTION (SUM OF THE WEIGHTED WIRE LENGTHS). THE NUMBERS IN THE PARENTHESES SHOW THE VALUES SCALED TO THE VALUES OF THE SMMS DESIGNS.

Benchmark	# tiers	SeqT			SeqQ			SliT			SMMS		
		V	L	P	V	L	P	V	L	P	V	L	P
n100_500	2	21,226 (1.17)	67,219 (1.34)	32,456 (1.33)	27,047 (1.49)	92,508 (1.84)	44,770 (1.83)	28,189 (1.55)	72,470 (1.44)	35,333 (1.45)	18,146 (1.00)	50,347 (1.00)	24,433 (1.00)
	3	22,608 (1.59)	61,691 (1.10)	30,230 (1.17)	25,874 (1.82)	91,290 (1.62)	44,449 (1.72)	16,085 (1.13)	58,919 (1.05)	28,393 (1.10)	14,253 (1.00)	56,215 (1.00)	25,860 (1.00)
	4	22,287 (1.29)	56,300 (1.58)	27,596 (1.61)	23,488 (1.35)	86,126 (2.42)	41,935 (2.44)	18,692 (1.08)	53,068 (1.49)	26,031 (1.51)	17,341 (1.00)	35,625 (1.00)	17,190 (1.00)
	5	19,191 (0.99)	57,502 (1.85)	22,542 (1.50)	25,153 (1.30)	91,065 (2.94)	44,592 (2.97)	17,582 (0.91)	40,427 (1.31)	19,715 (1.31)	19,400 (1.00)	30,937 (1.00)	15,027 (1.00)
	Avg.	(1.24)	(1.26)	(1.39)	(1.47)	(2.15)	(2.19)	(1.25)	(1.31)	(1.33)	(1.00)	(1.00)	(1.00)
n200_600	2	34,282 (1.01)	122,622 (1.45)	58,241 (1.43)	50,690 (1.49)	160,530 (1.89)	74,928 (1.85)	34,803 (1.02)	87,989 (1.04)	44,206 (1.09)	34,062 (1.00)	84,758 (1.00)	40,599 (1.00)
	3	38,231 (1.16)	120,814 (1.76)	57,455 (1.76)	47,701 (1.45)	170,572 (2.49)	81,138 (2.48)	33,468 (1.03)	69,422 (1.03)	33,076 (1.03)	32,846 (1.00)	68,618 (1.00)	32,689 (1.00)
	4	41,018 (1.21)	121,331 (2.03)	57,788 (2.08)	62,836 (1.85)	161,978 (2.71)	76,639 (2.76)	33,017 (0.97)	61,190 (1.02)	28,958 (1.04)	33,981 (1.00)	59,743 (1.00)	27,784 (1.00)
	5	39,696 (1.11)	121,101 (2.01)	57,759 (2.02)	52,191 (1.47)	146,513 (2.43)	69,285 (2.42)	36,051 (1.01)	61,965 (1.03)	29,409 (1.03)	35,604 (1.00)	60,201 (1.00)	28,619 (1.00)
	Avg.	(1.12)	(1.80)	(1.80)	(1.56)	(2.36)	(2.35)	(1.01)	(1.05)	(1.05)	(1.00)	(1.00)	(1.00)
n300_1000	2	37,489 (1.03)	170,558 (1.01)	83,681 (0.99)	86,868 (2.38)	376,977 (2.24)	186,771 (2.20)	48,263 (1.32)	182,309 (1.08)	97,721 (1.15)	36,522 (1.00)	168,538 (1.00)	84,748 (1.00)
	3	37,011 (0.99)	170,496 (0.99)	83,902 (0.99)	104,464 (2.81)	361,270 (2.11)	179,048 (2.12)	52,578 (1.41)	160,906 (0.94)	79,893 (0.94)	37,226 (1.00)	171,882 (1.00)	84,597 (1.00)
	4	38,040 (1.07)	173,986 (1.34)	84,961 (1.24)	94,159 (2.64)	409,077 (3.16)	203,014 (2.96)	46,623 (1.31)	130,784 (1.01)	65,339 (0.95)	35,642 (1.00)	129,570 (1.00)	68,624 (1.00)
	5	35,719 (0.99)	168,526 (1.01)	82,438 (1.01)	99,260 (2.76)	367,025 (2.20)	182,364 (2.23)	43,696 (1.21)	168,831 (1.01)	82,149 (1.01)	35,989 (1.00)	167,137 (1.00)	81,738 (1.00)
	Avg.	(1.02)	(1.08)	(1.05)	(2.64)	(2.39)	(2.35)	(1.31)	(1.01)	(1.01)	(1.00)	(1.00)	(1.00)

TABLE IV

RUNTIME COMPARISON OF THE 3-D FLOORPLANNING ALGORITHMS. WE SHOW ONLY RELATIVE RUNTIME VALUES.

Benchmark	SeqT	SeqQ	SliT	SMMS
n100_500	4.15	4.20	0.08	1.00
n200_600	3.96	4.45	0.09	1.00
n300_1000	4.82	5.07	0.12	1.00

A. Benchmarks and Simulation Settings

We generated three benchmarks to compare 3-D floorplan representations. The name of each benchmark is nA_B where A is the number of blocks and B is the number of nets. The three benchmarks are n100_500, n200_600, and n300_1000. Each block has a fixed volume. All the blocks are resizable in 3-D and the range of the planar aspect ratio (y-directional length / x-directional length) of the blocks is $[0.7, 1.3]$. For example, if the volume of a block is V and its z-directional length is t, its planar area is V/t. In this case, the minimum and maximum values of the width of the block are $\sqrt{V/(1.3t)}$ and $\sqrt{V/(0.7t)}$, respectively. When we generated the benchmarks, we also randomly generated access frequencies for all the nets to obtain and compare dynamic power consumption. We compare the volume, the total wire length, and the total dynamic power consumption. We use the half-perimeter wire length (HPWL) for the wire length computation and the weighted HPWL for the dynamic power consumption. The weighting factors are the access frequencies for the nets.

We implemented four 3-D floorplanning representations, Sequence Triple (SeqT), Sequence Quintuple (SeqQ), Slicing Tree (SliT), and the proposed algorithm (SMMS). We applied them to the simulated annealing algorithm with the following objective function:

$$C = \alpha \cdot V + \beta \cdot L + \gamma \cdot P \qquad (1)$$

where V is the volume, L is the wire length, P is the dynamic power consumption, and α, β, and γ are weighting factors for the volume, wire length, and power consumption, respectively. We ran each algorithm ten times for each benchmark and obtained average values. For a fair comparison, we used the same simulated annealing parameters (initial and final temperatures, cooling rate, etc.) for all the algorithms.

B. Comparison of Volume, Wire Length, and Power

Table III compares the four 3-D floorplanning algorithms for the three benchmarks. We also vary the number of tiers to compare the quality of the algorithms for different floorplanning configurations. For n100_500, the SMMS algorithm achieves 24% to 47% smaller volume, 26% to 115% shorter wire length, and 33% to 119% lower dynamic power consumption on average than the other algorithms. However, there are a few cases for which some of the other algorithms achieve smaller volume than the SMMS algorithm. For example, the volumes of the five-tier floorplan designed by the Sequence Triple and the Slicing Tree algorithms are 1% and 9% smaller than the SMMS designs, respectively. In those cases, however, the wire length and the power consumption of the floorplans designed by the two algorithms are 31% to 85% worse than those designed by the SMMS algorithm.

For the n200_600 benchmark, the SMMS representation still achieves the best volume, wire length, and power consumption on average. The Slicing Tree has 1% larger volume, 3% longer wire length, and 3% higher power consumption than SMMS. The Sequence Triple and Sequence Quintuple representations have 12% and 56% large volume, 80% and 236% longer wire length, and 80% and 235% higher power consumption than SMMS, respectively. We find similar trends for n300_1000 although the volume, wire length, and power differences between the Sequence Triple and SMMS and between the Slicing Tree and SMMS go down. The reason that the Sequence Quintuple shows the worst values is because Sequence Quintuple requires longer runtime or more perturbations to generate high-quality floorplans. Thus, if the same number of perturbations is applied, the Sequence Quintuple representation is expected to show the worst values.

A result to note is that building a block-level 3-D IC layout in multiple tiers does not necessarily increase the quality of the layout as the tier count goes up. For instance, SMMS obtains the smallest average volume, the shortest wire length, and the lowest power consumption for n300_1000 when the tier count is four. The solution set of the five-tier floorplanning includes the solution set of the four-tier floorplanning, so ideally the quality of the five-tier designs should be better than that of the four-tier designs. However, we use the simulated annealing algorithm, which is a stochastic algorithm. Thus, applying different constraints (max. four tiers vs. max. five tiers) does not necessarily lead to a better solution although the solution set of the latter includes that of the former.

C. Runtime

Table IV compares the average runtimes of the four algorithms for the three benchmarks. The evaluation complexity of the Slicing Tree representation is $O(n)$ where n is the number of blocks as shown in Table I, but that of the other three algorithms is $O(n^2)$. Thus. the Slicing Tree has the shortest runtime and almost ten times as fast as the SMMS algorithm and 40 times as fast as the Sequence Triple and the Sequence Quintuple algorithms. In addition, the runtime of the SMMS algorithm is approximately four times as short as the Sequence Triple and the Sequence Quintuple algorithms although they have the same evaluation complexity theoretically. The reason is that we use the evaluation time reduction technique explained in Section III-F. The technique helps reduce the complexity from $O(n^2)$ to $O(n)$. However, notice that we can apply a similar evaluation time reduction technique to the Sequence Triple and the Sequence Quintuple algorithms.

V. Conclusion

In this paper, we proposed a 3-D floorplan representation that supports independent x-, y-, and z-directional relations without any feasibility check. The new representation uses a relation matrix and multiple sequences to determine the relation between a pair of blocks. In addition, the representation can be extended to any-dimensional floorplanning problems. The simulation results show that the proposed representation constantly produces high-quality 3-D floorplans.

Acknowledgement

This work was supported by the Defense Advanced Research Projects Agency (DARPA) Young Faculty Award under Grant D16AP00119.

References

[1] C.-H. Shen, J.-M. Shieh, T.-T. Wu, W.-H. Huang, C.-C. Yang, C.-J. Wan, C.-D. Lin, H.-H. Wang, B.-Y. Chen, G.-W. Huang, Y.-C. Lien, S. Wong, C. Wang, Y.-C. Lai, C.-F. Chen, M.-F. Chang, C. Hu, and F.-L. Yang, "Monolithic 3D Chip Integrated with 500ps NVM, 3ps Logic Circuits and SRAM," in *Proc. IEEE Int. Electron Devices Meeting*, Dec. 2013, pp. 9.3.1–9.3.4.

[2] Y.-J. Lee and S. K. Lim, "Ultrahigh Density Logic Designs Using Monolithic 3-D Integration," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 12, Dec. 2013, pp. 1892–1905.

[3] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Power-Performance Study of Block-Level Monolithic 3D-ICs Considering Inter-Tier Performance Variations," in *Proc. ACM Design Automation Conf.*, Jun. 2014, pp. 1–6.

[4] ——, "Design and CAD Methodologies for Low Power Gate-level Monolithic 3D ICs," in *Proc. Int. Symp. on Low Power Electronics and Design*, Aug. 2014, pp. 171–176.

[5] D. H. Kim, S. Mukhopadhyay, and S. K. Lim, "TSV-Aware Interconnect Distribution Models for Prediction of Delay and Power Consumption of 3-D Stacked ICs," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 9, Sep. 2014, pp. 1384–1395.

[6] H. Yamazaki, K. Sakanushi, S. Nakatake, and Y. Kajitani, "The 3D-Packing by Meta Data Structure and Packing Heuristics," in *IEICE Transactions on Fundamentals*, vol. E83-A, no. 4, Apr. 2000, pp. 639–645.

[7] Y. Ma, X. Hong, S. Dong, and C.-K. Cheng, "3D CBL: An Efficient Algorithm for General 3D Packing Problems," in *Proc. IEEE Int. Midwest Symp. on Circuits and Systems*, Aug. 2005, pp. 1079–1082.

[8] L. Cheng, L. Deng, and M. D. F. Wong, "Floorplanning for 3-D VLSI Design," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 2005, pp. 405–411.

[9] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "Temporal Floorplanning Using the Three-Dimensional Transitive Closure subGraph," in *ACM Trans. on Design Automation of Electronics Systems*, vol. 12, no. 4, Sep. 2007, pp. 37:1–37:34.

[10] R. Wang, E. F. Y. Young, Y. Zhu, F. C. Graham, R. Graham, and C.-K. Cheng, "3-D Floorplanning Using Labeled Tree and Dual Sequences," in *Proc. Int. Symp. on Physical Design*, Apr. 2008, pp. 54–59.

[11] K. Fujiyoshi, H. Kawai, and K. Ishihara, "A Tree Based Novel Representation for 3D-Block Packing," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 5, May 2009, pp. 759–764.

[12] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "T-Trees: A Tree-Based Representation for Temporal and Three-Dimensional Floorplanning," in *ACM Trans. on Design Automation of Electronics Systems*, vol. 14, no. 4, Aug. 2009, pp. 51:1–51:28.

[13] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, Dec. 1996, pp. 1518–1524.

[14] M.-C. Tsai, T.-C. Wang, and T. Hwang, "Through-Silicon Via Planning in 3-D Floorplanning," in *IEEE Trans. on VLSI Systems*, vol. 19, no. 8, Aug. 2011, pp. 1448–1457.

[15] D. H. Kim, R. O. Topaloglu, and S. K. Lim, "Block-Level 3D IC Design with Through-Silicon-Via Planning," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 2012, pp. 335–340.

[16] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "High-Density Integration of Functional Modules Using Monolithic 3D-IC Technology," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 2013, pp. 681–686.