International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I9P108

Volume 72 Issue 9, 96-103, September 2024
© 2024 Seventh Sense Research Group®

Review Article

A Comparative Study on 2D and 3D Floorplan Representations in VLSI Physical Design

Sony Snigdha Sahoo[1], Prafulla Kumar Behera[2]

[1,2] Department of Computer Science and Applications, Utkal University, Vani Vihar, India.

[1]Corresponding Author : sony28788@gmail.com

Received: 03 April 2024 Revised: 16 July 2024 Accepted: 02 September 2024 Published: 28 September 2024

Abstract *- An important step in designing a chip layout is floorplanning. The location, shape, and size of modules in a chip are represented in the form of a floorplan. A floorplan shows the relative locations of electronic modules on a chip by dividing its core into rectangles.The quality of the chip implementation mostly depends on the floorplan's goodness. Effective placement of the modules and timing and congestion-related issues are also dependent on how well organized the floorplan is. Floorplan representation is the output of the floorplanning phase and serves as an intermediary between floorplanning and other subsequent phases. Thus, choosing an appropriate floorplan representation is critical for any further implementation. This survey paper discusses and compares the features of various 2D floorplan representations such as mosaic floorplan, bounded slice line grid, corner block list, sequence pair, O-tree, binary slicing tree, B* tree,and their respective 3D counterparts.*

Keywords *- 2D and 3D floorplan, B* Tree, Bounded slice line grid, Corner block list, Floorplan representation, O-tree, VLSI design.*

1. Introduction

In the standard design cycle of integrated circuits, the physical design phase is of utmost importance for geometrically representing a circuit and its interconnections in a layout. One of the vital components of physical designing is floorplanning, which determines the locations, shapes, and sizes of the chip's modules so that there is no overlap among the modules. Estimating the area of the chip, delay,and congestion in wiring are also some of the elementary purposes of floorplanning.The output of the floorplanning phase is a floorplan that presents a layout for optimizing various cost metrics.As an efficient representation of floorplan is desired for managing circuit design complexity, both two-dimensional (2D) and three-dimensional (3D) floorplan representations have been proposed over time. While 2D representations provide a topological view without any reference to depth or height, 3D floorplans include a lot of intricate detailing, thereby providing substantial improvement in wirelength, bandwidth, and overall performance.3D ICs are usually formed via vertically stacking several 2D silicon dies coupledvia TSVs (Through Silicon Vias). Intra-block wire latency can be decreased by swapping out multiple 2D layouts for a single 3D block with a cubic block for each component and heights in the z position. The evolution of different floorplan representations for depicting VLSI floorplans can be attributed to several factors. Each representation method aims to address specific challenges and requirements in the design and optimization of VLSI circuits. Some of the key reasons for the development of various floorplan representations:

- New representations are needed to handle the complexity arising from the large number of components and their interconnections.
- Different representations offer different ways to optimize key objectives such as area, wire length, timing, power consumption, and manufacturability, among which some might be better suited for minimizing area. In contrast, others might be more effective for reducing interconnect delays or power usage.
- The efficiency of floorplanning algorithms' time and space complexity can depend heavily on the representation used. Some representations enable more straightforward and faster computation of potential solutions, while others might provide more flexibility but at the cost of increased computational complexity.
- Different representations offer varying levels of flexibility in handling specific design constraints and requirements. For example, some methods might be better at accommodating hierarchical designs or handling non-rectangular blocks.
- Advances in fabrication technology and design tools have necessitated new representations. For instance, the shift from 2D to 3D ICs has required new floorplanning methods to effectively manage the additional dimension.
- Some representations are easier to understand and implement, making them more accessible for designers. Ease

of visualization and manipulation can also be important factors in choosing a representation.

- The compatibility of floorplan representations with other design automation tools, such as placement, routing, and verification tools, can influence the choice of representation. Seamless integration with existing tools and workflows is often a critical consideration.

Keeping in view the significance of floorplan representation, our contributions in this paper include the following;

- Comparative evaluation of the characteristics, solution space, run time, and viability of several 2D and 3D representations.
- A concise outline of each representation's advantages and drawbacks.

The remaining text is structured as follows: a brief overview of different floorplan structures is provided in section 2. In Section 3 provides a brief discussion on the various 2D floorplan respresentations, and their respective 3D counterparts have been briefed in Section 4. Section 5 presents a comparison among different floorplan respresentations. The future scope of this work has been presented in Section 6.

2. Floorplan Structure

Floorplans may be classified as either slicing floorplans or non-slicing floorplans [1]. If a floorplan can be generated by repeatedly cutting it in either direction, i.e. vertically or horizontally, it is said to be slicing; this is not the case with non-slicing floorplans. Figure 1 demonstrates the structure of the 2D slicing and non-slicing floorplan, respectively, while Figure 2 depicts their 3D counterparts. Another general form of non-slicing floorplan, namely the mosaic floorplan, is a simplified version of the 2D floorplan[16]. This floorplan class is not supposed to have any unoccupied rooms or crossing cuts.The mosaic floorplans can be depicted using horizontal and vertical constraint graphs. Two representations that effectively portray the very same feature are the corner block list[18] and the twin binary sequence[12]. For each mosaic floorplan, two binary trees are built in a twin binary sequence format, and from these binary trees, two binary strings are created that allow for a unique reconstruction of the floorplan.

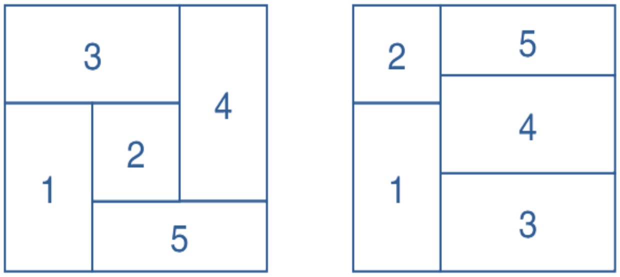

Fig. 1 2D Slicing and non-slicing representation[1]

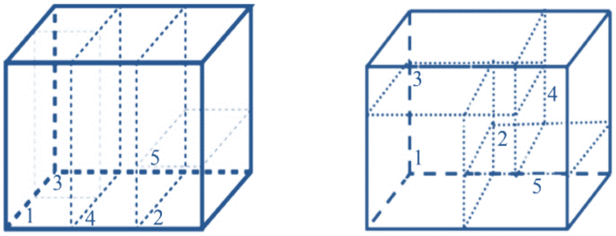

Fig. 2 3D Slicing and Non-Slicing Representation[23]

3. 2D Floorplan Representations

3.1. Representations for Slicing Floorplan

Two of the most frequently used representations for depicting a slicing floorplan are the polish expression and the slicing tree.

3.1.1. Slicing Tree

VLSI floorplans can be effectively represented using slicing trees, which allows circuit modules to have flexibility in their shape and orientation, enabling the identification of extremely compact slicing floorplans. However,when working with hard modules, they result in poor space utilization. As a result, a great deal of current research has concentrated on developing different non-slicing floorplan representations that can aid in better space usage. A binary slicing tree can well describe the hierarchical structure of a slicing floorplan. The slicing floorplan and its respective slicing tree are displayed in Figure 3. A slicing tree's internal nodes are denoted by the letters V or H, which stand for vertical and horizontal cuts, respectively. Each of the leaves of the tree corresponds to a module. For a particular floorplan, multiple slicing tree representations may exist because the cuts are not related to any dimensional information.

3.1.2. Polish Expression

The polish expression representation attained by the post-order traversal of a slicing tree can be used as a floorplan representation [20]. Figure 4 depicts the polish expression for the slicing tree presented in Figure 3.As there may be more than one polish expression for representing a given slicing floorplan, it is not a desirable slicing tree representation.

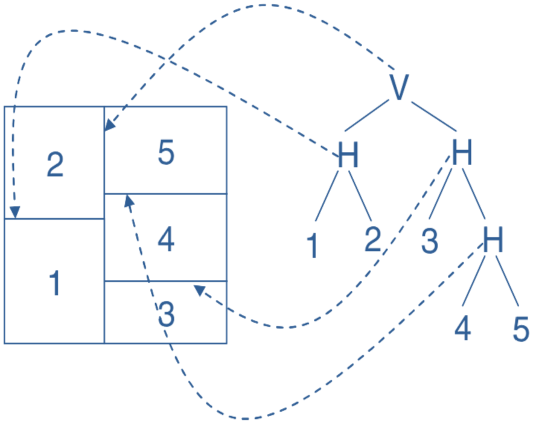

Fig. 3 Floorplan and its slicing tree[1]

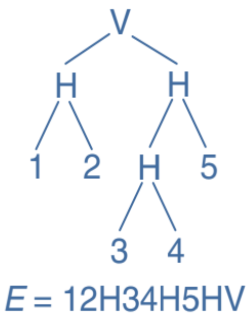

Fig. 4 Slicing tree with its polish expression[1]

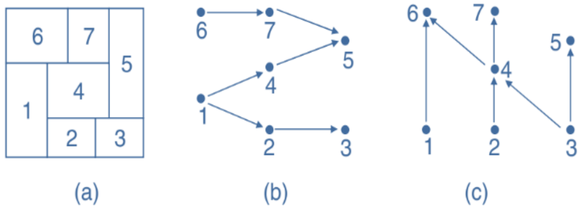

Fig. 5 Floorplan and respective HCG and VCG[1]

Additionally, a normalized polish expression has also been created; it is the outcome of skewed slicing trees in which there are no successive operators of the same type [2]. It enables the construction of a unique slicing floorplan.

3.2. Representations for Non-Slicing Floorplan

A more general form of floorplanning is the non-slicing floorplan, as shown in Figure 1. It is not possible to model a non-slicing floorplan with a slicing tree. Instead, Horizontal Constraint Graphs (HCG) and Vertical Constraint Graphs (VCG) were initially proposed for modeling non-slicing floorplans.

While the vertical constraint graph defines how the modules are placed vertically, the horizontal constraint graph establishes the same horizontally, as demonstrated in Figure 5 [20]. Other non-slicing floorplan representations include bounded slice line grid, sequence pair, Transition Closure Graph, TCG-S, Corner Sequence, O-Tree, B* Tree, etc.

3.2.1. Bounded Slice Line Grid

A BSG structure is made up of horizontal and vertical unit segments that cross each other, forming rooms. Figure 6 illustrates a BSG having dimension P x Q, BSG_{PxQ}.Whenever the BSG structure is used for depicting a placement, the number of modules must be less than or the same as P x Q. A consecutive pair of horizontal and vertical units form a rectangular space called a room. Each module is placed in a separate room.BSG uses the longest path algorithm to figure out the position of modules. It has proved useful in the placement of hard modules.

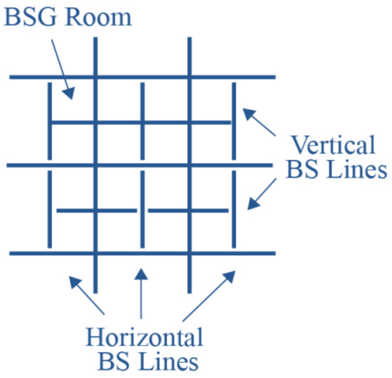

Fig. 6 Bounded slice line grid[10]

3.2.2. Sequence Pair

Another slicing technique for solution space has been proposed in [7], in which a set of module name sequences or sequence pairs represents each packing.Geometric relations between each pair of blocks are represented by two permutations.For instance, the following arrangement in a pair shows the following.

$(..M_1..M_2..,..M_1..M_2..) \rightarrow M_1$ is left of M_2,
$(..M_1..M_2..,..M_2..M_1..) \rightarrow M_1$ is above M_2,
$(..M_2..M_1..,..M_1..M_2..) \rightarrow M_1$ is below M_2,
$(..M_2..M_1..,..M_2..M_1..) \rightarrow M_1$ is right of M_2[5].

However, this technique results in a very long runtime.

3.2.3. Transitive Closure Graph(TCG)

A Directed Acyclic Graph's transitive closure is represented by the graph $G_0 = (V, E_0)$ where an edge $E_0 = \{(n_i, n_j)|$there exists a path from n_i to n_j in G$\}$.It makes use of horizontal and vertical closure graphs to describe the geometric relationships among each pair of modules,as shown in Figure 7[8]. It is the first floorplan representation where perturbation can be done on graphs directly, along with a guarantee of a feasible solution in each perturbation. A representation, termed TCG-S, which merges the features of TCG and SP, has been proposed in [11].This representation uses vertical and horizontal transitive closure graphs and a packing sequence to represent a placement. It has proved superior as it combines faster packing and perturbation from SP with transparent geometric relations among the modules from TCG, leading to faster convergence.

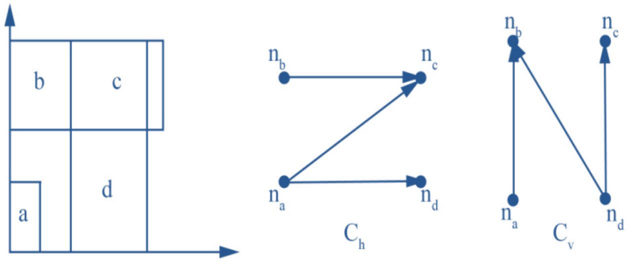

Fig. 7 A placement and its corresponding Transitive Closure Graph[10]

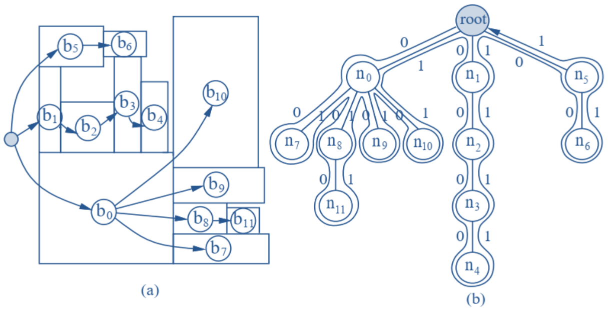

Fig. 8 O-Tree for a given admissible placement[20]

3.2.4. Corner Sequence

The corner sequence consists of two tuples that indicate the corners where the modules are inserted and the module packing sequence.The CS representation, <(S_1, D_1)(S_2, D_2),…,(S_m, D_m)>, describes a compacted placement as a function of a packing sequence “S” of “m” modules and matching bends D produced by the modules[12].

3.2.5. O-Tree

An O-tree is a tree structure that is ordered.For every node, it can have an arbitrary number of branches. This irregular structure of the branches leads to complicated tree operations and incurs a lot of overhead in sequence encoding [13]. The structure of an O-tree is shown in Figure 8 and is encoded by the two tuple (001001101011000011110011,b_0,b_7,b_8,b_{11},b_9,b_{10},b_1,b_2,b_3,b_4,b_5,b_6) obtained by DFS on the O-tree[14].Compaction operation is included in the O-tree structure,and it takes linear time to turn an O-tree into a constraint graph.

3.2.6. B*-Tree

The basis of the B*-Tree is an ordered binary tree.Its construction is similar to that of a depth-first search. First, the subtree on the left is built recursively from the root, and then the right subtree[20]. Figure 9 shows a B*-Tree along with its admissible placement. It can be seen in the figure that B*-Tree has a fixed number of branches. It can flexibly deal with soft, hard, rectilinear and preplaced modules.

3.2.7. Corner Block List

Another simple version of a mosaic structure is a corner block list.It represents each block as a room with just topological relationships between its members. In a mosaic structure, the topological relations are represented by CBL using a triplet (S, L, T), where S is the set of blocks, L is the orientation list, and T is the T-junction list. This chip has been partitioned into rectangular rooms, with one block allocated to each room according to (S, L, T) as shown in Figure 10. The Corner Block is the block that is placed into the placement's upper right corner (CB)[19]. The main benefit of corner block lists over O-trees is their ability to depict floorplans regardless of block sizes. This facilitates the convenient optimization of blocks with different width and height configurations.

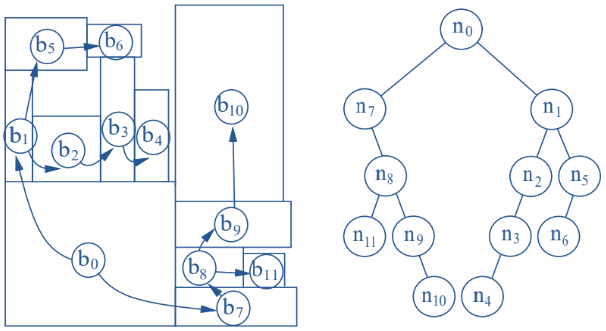

Fig. 9 B*-Tree for a given admissible placement[20]

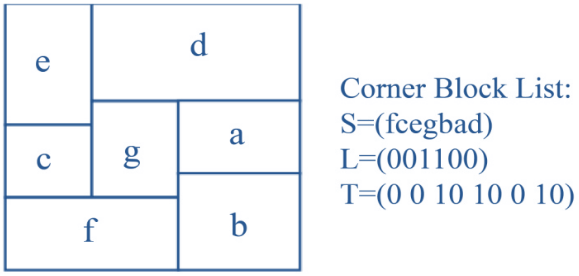

Fig. 10 Corner block list for a given non-slicing floorplan [21]

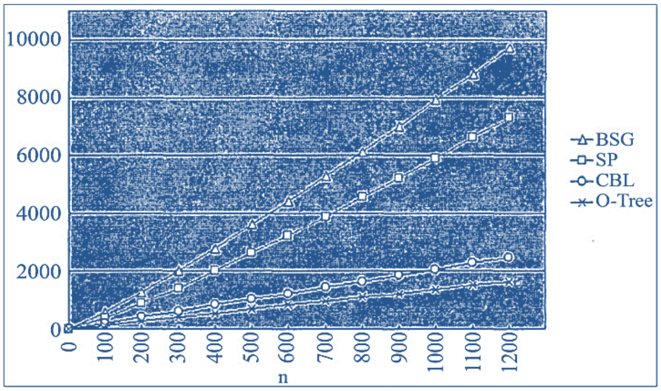

Fig. 11 Comparison among combination of four modes namely BSG, SP, CBL, O-Tree normalized by n![19]

4. Various 3D FloorplanRepresentations

A number of 3D floorplanning representations are trivial extensions of their 2D counterparts. Sequence-Triple, Sequence-Quadruple and Sequence-Quintuple are 3D extensions of Sequence-Pair representation for 2D floorplans. TCG and BSG have been extended into 3D-sub Transitive Closure Graph (TCG) and 3D Bounded Slice-line Grid (BSG),respectively. 3D CBL (Corner Block List) is a renewed from 2D CBL for encoding 3D packing topologies.

4.1. Sequence Triple (ST)

A modified version of the Sequence-Pair representation is Sequence Triple [21]. It uses three block sequences, and their positions in the three sequences determine the relationship between each block. The amount of information on 3D packing that ST can transmit is limited.

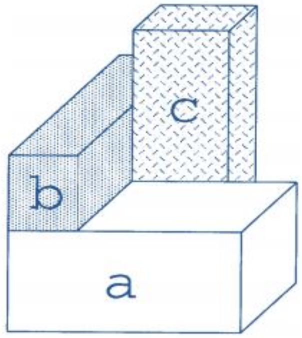

Fig. 12 A given seq-triple(bac,acb,abc) with respective topology in terms of 3D packing[7]

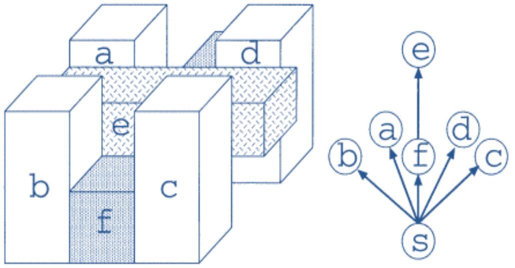

Fig. 13 3D packing, whose topology can be addressed by seq-quintuple(ebafcd, abfdce, fcbeda, bceadf, abcdfe) but not by seq-triple [7]

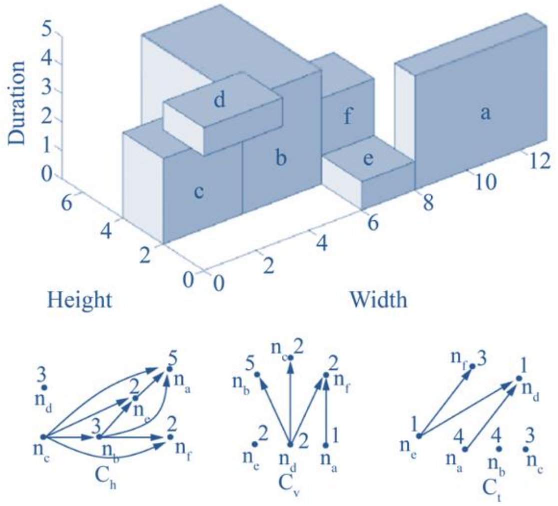

Fig. 14 A placement and its respective 3d-TCG[18]

4.2. Sequence Quintuple (SQ)

Sequence Quintuple [21] employs five sequences of blocks, and each possible solution stands for a distinct 3-D floorplan. The x-directional and y-directional relationships between a pair of blocks are established with the first and second sequences, respectively. At the same time,a z-directional relationship between two blocks can only be established by the fifth sequence and is only applicable when there are no x-directional or y-directional relationships between a given set of blocks.

4.3. 3D Corner Block List (CBL)

A 3D CBL has been introduced as a representation for 3D floorplans using a triplet of three elements (S, L, T), where S represents a block list, L represents an orientationset, and T represents information on the junction. It is different from 2D-CBL in the sense that here, the plane is divided into cubic rooms, and thus, dependencies exist among the coordinates along x, y, and z-axes[21].

4.4. 3D Transitive Closure SubGraph

In [2], the 3-D Transitive Closure subGraph (3D-sub-TCG) determines the relationship between two blocks along the three axes using three transitive graphs, horizontal, vertical and temporal transitive closure graphsas depicted in fig.18, for modelling the temporal as well as the spatial relations among the modules. However, several feasibility conditions must be met for a 3D-subTCG to be considered viable.

4.5. Single-Tree Dual-Sequence Representation

In [15], a 3-D floorplan representation termed a single-tree dual-sequence is suggested that relies on a permutation sequence, a number sequence and a labelled tree. This representation aims at optimizing volume. However,this approach may not be efficient in minimizing wirelength as sometimes the floorplan configuration, which results in the smallest wire length,need not necessarily have the smallest volume.

4.6. Double Tree and Sequence

In a study referenced as [9], a floorplanning strategy named Double Tree and Sequence (DTS) was presented, which operates in a 3-dimensional space. For determining the positional relationships between a given set of blocks in the x and y dimensions, DTS employs two x- and y-tree. Furthermore, DTS utilizes a sequencing technique to establish the positional relationship between any two blocks that overlap in the z-direction plane. As a result, the blocks' z-coordinates are determined based on their coordinates along x and y-axes.

4.7. T-Tree

T-Tree, a tree-based datastructure, has been used as a 3D floorplanning technique. Each block node in the T-Tree has three child nodes that determine the relationship between the child and parent nodes. Nonetheless, feasibility checks are to be carried out for some solution perturbations, like swap and move operations.T-tree representation outperforms 3D-sub-TCG in terms of volume optimization and packing efficiency.

4.8. Single Matrix Multiple Sequences (SMMS)

In comparison to other 3-D floorplan representations, this one handles the x, y, and z coordinates independently, offering a greater solution space. Here, each probable solution represents a feasible floorplan, thereby requiring no additional feasibility checks. SMMS can be generalized for any dimensional floorplanning problems[3].

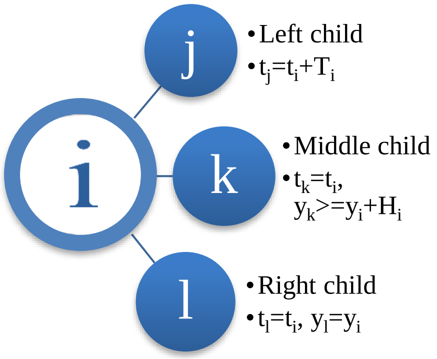

Fig. 15 Structure of a T-tree[14]

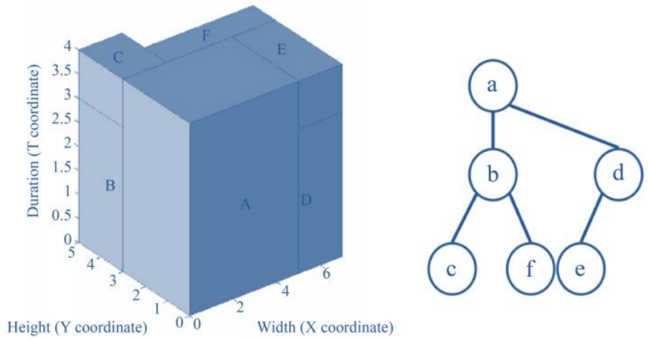

Fig. 16 A compacted placement and its corresponding T-tree[14]

Table 1. Comparison among 2D floorplan representations based on runtime and space[14]

Representation	Solution Space	Runtime
Polish Expression	$O(n!2^{3n}/n^{1.5})$	O(n)
BSG	$O(n!C(n^2,n)$	$O(n^2)$
SP	$(n!)^2$	$O(n^2)$
TCG	$(n!)^2$	$O(n^2)$
TCG-S	$(n!)^2$	$O(n^2)$
CS	$O((n!)^2)$	O(n)
CBL	$\leq O(n!2^{3n})$	O(n)
O-Tree	$O(n!2^{3n}/n^{1.5})$	O(n)
B*-Tree	$O(n!2^{3n}/n^{1.5})$	O(n)

Table 2. Comparison of 3D Floorplan Representations based on Runtime and Space

Representation	Solution Space	Runtime
Sequence Triple[6]	$O((n!)^3)$	$O(n^2)$
Sequence Quintuple[6]	$O((n!)^5)$	$O(n^2)$
3D CBL[7]	$O(n!48^n)$	O(n)
Slicing Tree[8]	$O(6^n(n!)^2)$	O(n)
3D-subTCG[9]	$O((n!)^3)$	$O(n^2)$
Tree+seq²[10]	$O((n+1)^n(n!)^2)$	$O(n^2)$
DTS[11]	$O(n!(n+1)^{2n})$	$O(n^2)$
T-Tree[12]	$O(n!\frac{3^{3n}}{2^{2n}n^{1.5}})$	$O(n^2)$
SMMS	$O(7^{\frac{n(n-1)}{2}(n!)^3})$	$O(n^2)$

Table 3. Comparison among floorplan representations based on features

Representation	Advantages	Limitations
Polish Expression	Selects the floorplan that best fits among many floorplans represented by a particular polish expression.	Does not consider placement constraints. Not P-admissible
BSG	Efficiently manages general floor layout, which includes soft blocks, L- and T-shaped structures, and non-slicing structures. P-admissible representation	Has an excessively large number of redundancies and a solution space leading to longer runtime in searching for a good solution
SP	Adequately deals with the high-performance analog layout. P-admissible representation	Encoding module sequences is necessary for tree operations and computing costs using constraint graphs.
TCG	Memory usage is smaller, supports incremental updates during operations, and geometric relations among modules are transparent. P*-admissible representation	TCG evaluates its packing cost using constraint graphs.
TCG-S	Faster packing time, transparent geometric relation among modules, Supports incremental update, faster convergence. P*-admissible representation	
CS	Permits packing with incremental updates and generates generic packing with worst-case linear time. P-admissible representation	
CBL	Needs fewer bits than SP and BSG for encoding, can represent floorplans independent of block sizes	It can represent only mosaic floorplans. Not P-admissible
O-Tree	Smaller solution space and fewer bits for encoding than SP and BSG. It can represent only LB-compact placement.	Irregular, unpredictable no. of branches, Incurs high operation complexity, limited positions for new insertion. Not P-admissible.
B*-Tree	Handles various types of modules flexibly, with Smaller solution space and encoding cost	Placement must be left or bottom compacted. Not P-admissible

5. Comparison among Various Floorplan Representations

A floorplan must be based on criteria like properties of the representation, size of the solution space along with the solution structure established by the floorplan, and operations that can be carried out on the representation. All of these criteria must be considered while evaluating floorplans. The omission of any may lead to pitfalls. Among the representations, general floorplans can be represented by sequence pair, TCG, TCG-S, & ACG, while O-tree, B*-tree, and corner sequence can well depict compacted floorplans. CBL and TBS models portray the mosaic floorplans with exactly one module in each room. Normalized Polish expressions are restricted to slicing floorplans only. Comparisons are presented in Tables 1 and 2.

6. Conclusion and Future Scope

This paper has briefly compared various 2D and 3D floorplan representations in terms of runtime, solution space, and features. The advantages and limitations have also been discussed. It can serve as a manual for the creation of better representations in the future.

References

[1] Tung-Chieh Chen, and Yao-Wen Chang, “Floorplanning,” *Electronic Design Automation*, pp. 575-634, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[2] D.F. Wong, and C.L. Liu, “A New Algorithm for Floorplan Design,” *23rd ACM/IEEE Design Automation Conference*, Las Vegas, NV, USA, pp. 101-107, 1986. [CrossRef] [Google Scholar] [Publisher Link]

[3] Shantonu Das, and Dae Hyun Kim, “A Non-Slicing 3-D Floorplan Representation for Monolithic 3-D IC Design,” *20th International Symposium on Quality Electronic Design*, Santa Clara, CA, USA, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Evangeline Fung Y. Young, Chris Chong Nuen Chu, and Cien Shen, “Twin Binary Sequences: A Nonredundant Representation for General Nonslicing Floorplan,” *Proceedings of the 2002 International Symposium on Physical Design*, vol. 22, no. 4, pp. 457-469, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[5] Hiroshi Murata, and Ernest Kuh, “Sequence-Pair Based Placement Methods for Hard/Soft/Preplaced Modules,” *Proceedings of the 1998 International Symposium on Physical Design*, pp. 167-172, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[6] H. Murata et al., “VLSI Module Placement based on Rectangle-Packing by the Sequence Pair” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518-1524, 1996. [CrossRef] [Google Scholar] [Publisher Link]

[7] Hiroyuki Yamazaki et al., “The 3D-Packing by Meta Data Structure and Packing Heuristics,” *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 83, no. 4, pp. 639-645, 2000. [Google Scholar] [Publisher Link]

[8] Jaiming Lin, and Yao-Wen Chang, “TCG: A Transitive Closure Graph based Representation for Nonslicing Floorplans,” *Proceedings of the 38th Annual Design Automation Conference*, Las Vegas Nevada USA, pp. 764-769, 2001.[CrossRef] [Google Scholar] [Publisher Link]

[9] Kunihiro Fujiyoshi, Hidenori Kawai, and Keisuke Ishihara, “A Tree Based Novel Representation for 3D-Block Packing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 5, pp. 759-764, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[10] M. Kang, and W.W.M. Dai, “General Floorplanning with L-shaped, T-shaped and Soft Blocks based on Bounded Slicing Grid Structure,” *Proceedings of ASP-DAC '97: Asia and South Pacific Design Automation Conference*, Chiba, Japan, pp. 265-270, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[11] Jai-Ming Lin, Yao-Wen Chang, “TCG-S: Orthogonal Coupling of P*-Admissible Representations for General Floorplans,” *Proceedings of the 39th Annual Design Automation Conference*, New Orleans Louisiana, USA, pp. 842-847, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[12] Jai-Ming Lin, Yao-Wen Chang, and Shih-Ping Lin, “Corner Sequence - a P-admissible Floorplan Representation with a Worst Case Linear-Time Packing Scheme,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*,vol. 11, no. 4, pp. 679-686, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[13] Pei-Ning Guo, Chung-Kuan Cheng, and T. Yoshimura, “An O-Tree Representation of Non-slicing Floorplan and Its Applications,” *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*, New Orleans, LA, USA, pp. 268-273, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[14] Pinghung Yuh, Chia-Lin Yang, and Yao-Wen Chang, “T-trees: A Tree-based Representation for Temporal and Three-Dimensional Floorplanning,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 14, no. 4, pp. 1-28, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[15] Renshen Wang et al., “3-D Floorplanning Using Labeled Tree and Dual Sequences,” *Proceedings of the 2008 International Symposium on Physical Design*, pp. 54-59, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[16] S. Nakatake et al., “*Bounded-Slicing Structure for Module Placement*,” Technical Report of the Institute of Electronics, Information and Communication Engineers of Japan, vol. 94, pp. 19-24, 1994. [Google Scholar]

[17] Song Chen et al., “Evaluating a Bounded Slice-Line Grid Assignment in O(nlogn) Time,” *2003 IEEE International Symposium on Circuits and Systems (ISCAS)*, Bangkok, Thailand, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[18] X. Hong et al., “Corner Block List Representation and its Application to Floorplan Optimization,” *IEEE Transaction on Circuits and Systems II: Express Briefs*, vol. 51, no. 5, pp. 228-233, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[19] Xianlong Hong et al., “Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan,” *IEEE/ACM International Conference on Computer Aided Design, ICCAD – 2000, IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140)*, San Jose, CA, USA, pp. 8-12, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[20] Yun-Chih Chang et al., “B*-Trees: A New Representation for Non-Slicing Floorplans,” *Proceedings 37th Design Automation Conference*, Los Angeles, CA, USA, pp. 458-463, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[21] Yuchun Ma et al., “3D CBL: An Efficient Algorithm for General 3D Packing Problems,” *48th Midwest Symposium on Circuits and Systems*,Covington, KY, USA, vol. 2, pp. 1079-1082, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[22] Pinghung Yuh, Chia-Lin Yang, and Yao-Wen Chang, “Temporal Floorplanning using the Three-Dimensional Transitive Closure SubGraph,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 4, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[23] P. Sivaranjani, and A. Senthilkumar, “3D VLSI Non-Slicing Floorplanning Using Modified Corner List Representation,” *Indian Journal of Science and Technology*, vol. 8, no. 35, pp. 1-6, 2015. [CrossRef] [Google Scholar] [Publisher Link]