Investigating Modern Layout Representations for Improved 3D Design Automation

Robert Fischbach
fischbach@ieee.org

Jens Lienig
jens@ieee.org

Johann Knechtel
knechtel@ifte.de

Dresden University of Technology
Institute of Electromechanical and Electronic Design
www.ifte.de, Dresden, Germany

ABSTRACT

The current trend towards 3D integration requires new layout representations specifically designed to take 3D-specific constraints into account and to facilitate efficient design algorithms. We observe that it is difficult to compare and evaluate these layout-specific data structures. In this paper, we first present a detailed investigation of modern layout representations while analyzing their solution space and their characteristics, such as redundancy and reachability. Our investigation reveals their potential for 3D applications but also shows open challenges to be considered for (future) representations. Thus, we also provide guidelines for designing efficient layout representations. Finally, we release our investigation methodology as open-source tool, thus providing interested researchers with the opportunity to conduct reasonable evaluations on their own.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Layout*; E.1 [**Data**]: Data Structures; G.3 [**Probability and Statistics**]: Probabilistic Algorithms

General Terms

Algorithms, Design, Theory

Keywords

3D design, 3D layout representations, data structures, physical design

1. INTRODUCTION

Physical design automation of electronic systems is based on an abstract model of the corresponding design problem which is computationally represented as a data structure. These *layout representations* store information about layout elements and, if designed properly, provide helpful features (*operations*), such as direct access to adjacent elements.

Design optimization is performed in the realm of the data structure's solution space by applying (stochastic) optimization algorithms. Efficient algorithms require a solution space that minimizes redundancy, excludes invalid solutions, and includes best solutions. Importantly, an efficient layout representation must also allow fast execution of operations, such as the exchange of *modules* (i.e., functionally grouped sub-circuits), transformation from the abstract representation to the real geometry, and consideration of layout constraints.

Various efficient layout representations have been presented for physical design of conventional (2D) integrated circuits (e.g., [19]). However, the current trend towards 3D integration creates a multitude of new design challenges. This process requires new and efficient layout representations that take into account the characteristics of nanoscale 3D designs. Layout representations that support efficient consideration of vertical constraints, e.g., inter-layer thermal relationships, are especially important, yet lacking in today's design systems. A summary of current 3D integration technologies and related layout representations is given in [8].

The first 3D physical design flows utilized traditional, proven layout representations, for instance, one slicing tree per active device layer. However, the disadvantage of these approaches is that the tight linking between layers is neglected and, for example, successful thermal-driven design is often not achieved. In order to exploit the potential of modern 3D integration technologies, contemporary 3D layout representations must fully support vertical constraints.

In this paper, we first present a detailed overview of modern 3D layout representations with regard to their main characteristics (Sec. 2 and 3). Rigorous evaluation and comparison of these layout representations is one of our key contributions in this paper. Here, we present a novel evaluation methodology based on the most important characteristics of the respective layout representation, such as redundancy and reachability of different solutions. Second, we discuss experimental results, revealing the applicability of modern layout representations with regard to cost minimization and reachability (Sec. 4). Third, we summarize our observations and provide guidelines for the development and application of (future) 3D layout representations (Sec. 5). Furthermore, our evaluation tool and its source code are online available as open-source [1].

2. LAYOUT REPRESENTATIONS

Runtime complexity and the size of the solution space are common criteria in publications that investigate various layout representations. However, a detailed comparison based only on these two (main) criteria is insufficient for application-specific decisions. Additional criteria are required to compare several layout representations in depth.

Chan et al. [4] investigated the importance of floorplan representations for physical design. They pointed out that cost evaluation is the most time-consuming computation during optimization, diminishing the relevance of efficient permutation and transformation operations. Consequently, this highlights the importance of cost-evaluation-related criteria such as flexibility, size of solution space, and a possible correlation with given objectives.

Table 1: Overview of common 3D layout representations. The number of modules is denoted as n, the number of layers as l, and unavailable data as *n.g.*.

Layout Representation	Pub.	Complexity	Solution Space	Main Characteristics
Sequence Triple [Quintuple] [18]	2000	$O(n^2)$	$O((n!)^3)$ $[O((n!)^5)]$	Three [five] sequences (*locii*)
Bounded-Sliceline Grid Array [7]	2001	$O(n^2)$	*depends on grid*	Multiple Bounded-Sliceline Grids
Multi-layer Slicing Tree Structure [2]	2004	$O(n)$	$O((n!\ 2^{3n}/n^{1.5})^l)$	Multiple Slicing Trees
Multi-layer Sequence Pair [14]	2004	$O(n \log \log n)$	$O((n!)^{2l})$	Multiple Sequence Pairs
Combined Bucket and 2D Array [6]	2004	*n.g.*	*n.g.*	Array of Transitive Closure Graphs and bucket structure
3D Sub-Transitive Closure Graph [21]	2004	$O(n^2)$	$O((n!)^3)$	Three transitive graphs
T-Tree [20]	2004	$O(n^2)$	$O(n!\ 3^{3n}/2^{2n}n^{1.5})$	Ternary tree; nodes: modules, branches: neighbor information
3D Bounded-Sliceplane Grid [17]	2005	$O(n^2)$	*depends on grid*	3D grid structure
3D Corner Block List [12]	2005	$O(n)$	$O(n!\ 3^{n-1}\ 2^{4n-4})$	Sequence of modules, list of orientations, list of tri-branches
3D Slicing Tree [5]	2005	$O(n)$	$O(n!\ 2^{3n}/n^{1.5})$	Binary tree; inner nodes: slices, leaves: modules
Sequence Quadruple [10]	2006	$O(n^2)$	$O((n!)^4)$	Four sequences (*locii*)
O-Sequence [13]	2006	$O(n)$	*n.g.*	Sequence of modules and symbols
Layered Transitive Closure Graph [11]	2006	$O(n^2)$	$O((n!)^{2l})$	Two Transitive Closure Graphs and layer information
Double Tree and Sequence [9]	2007	$O(n^2)$	$O(n!\ n^{2(n-1)})$	Two rooted trees (x(y)-tree), sequence (z-order)
Labeled Tree and Dual Sequences [15]	2008	$O(n^{4/3}\ \log n)$	$O((n!)^2\ n^{n-1})$	Sequence of modules, number sequence and tree
Twin Quaternary Tree [16]	2009	$O(n)$	$O((4n)!^2/(3n+1)!^2)$	Two quaternary trees; nodes: modules, branches: neighbor relations

In Tables 1 and 2, we list several criteria (and review their fulfillment by several layout representations) that are important to evaluate the practical relevance of modern 3D layout representations. Among others, this informations is helpful in selecting the most suitable layout representation dependent on application-specific constraints and operations.

Sections 2.1 - 2.8 describe the used evaluation criteria of Tables 1 and 2 in detail.

2.1 Runtime Complexity

Discussing runtime complexity of layout representations requires to determine the complexity of operations depending on the number of modules. In general, permutation and transformation operations are considered in such investigations. Despite its importance (see [4]), cost evaluation is mostly neglected. Consequently, cost evaluation with a competitive runtime complexity is required in such investigations. Layout representations with inherent cost-correlating properties would enable runtime reduction compared to a data-structure-independent evaluation.

In general, the comparison of runtime complexities strongly depends on the problem size. For small problem sizes, conclusions can be inaccurate. Nevertheless, it is a useful characteristic to rate scalability. In summary, the usage of practically relevant problem sizes (as given in most benchmark sets) is required for reasonable comparisons.

2.2 Size of the Solution Space

The size of the solution space refers to the number of possible solutions encoded by a specific layout representation.

Rotations of modules are sometimes neglected in the reported number of possible solutions. Depending on symmetry, complexity is increased by a factor of up to 24^n (n being the number of modules) if rotations are considered. Mirrored modules would increase complexity further by a factor of 6^n. Please note that it is an application-specific decision if rotated and/or mirrored modules should be considered for the solution space size.

Considering the solution space size alone (or in combination with runtime complexity) is insufficient for a detailed comparison of layout representations. Additional criteria, such as the ones listed below, are required.

2.3 Level of Abstraction

Some layout representations directly model geometric relations between modules (similar to constraint graphs), others use abstractions. The less abstract a layout representation, the easier the implementation of both geometric constraints and geometric operations. However, in such cases it is difficult to realize an evenly distributed search over the solution space. As illustrated in Table 1, more abstract layout representations provide a reduced number of possible solutions (and limit the solution space to the most relevant solutions).

2.4 Layout Classification in Floorplanning

A common criterion for investigating 3D layout representations is the layout classification provided by 3D floorplanning. It can be distinguished into several categories, which are based on the corresponding floorplanning problem.

Slicing: Packing description by a recursive dissection of a cuboid into two sub-cuboids. A well-known example for a slicing layout representation is the 3D Slicing Tree [5].

Mosaic: Extension of T-junctions known from 2D mosaic layout representations into the third dimension. Modules with a common edge or surface form such a 3D T-junction as described for the Twin Quaternary Tree [16]. A sufficient amount of dummy modules enables the representation of general packings.

Compacted: Arbitrary arrangement of modules which are compacted to the left, bottom, and/or front.

General: Representation of all possible geometrical relations between modules. General layouts without any compaction are unusual for floorplanning representations but may be necessary for other design problems.

2.5 Completeness and Redundancy

A layout representation is *complete* if all possible solutions of a certain layout classification can be represented. For example, the 3D Slicing Tree can represent all possible slicing layouts, i.e., it is a complete slicing representation. However, most of the compacted layout representations are incomplete because completeness is often lost due to the transformation from 2D to 3D (e.g., T-Tree [20]). One reason is that *cyclic packings* (circular dependencies of relative positions) can occur in 3D layouts. A sequential encoding prevents such cycles. Complete and compacted 3D layout representations are rare, Sequence Quintuple [18] is one example at the expense of a very large solution space with many redundant solutions (see Sec. 4). In general, the closer a layout representation to completeness, the larger the solution space. However, the solution space is usually correlated to redundancy, i.e., there are abstract solutions resulting in the same geometric representation. A low fraction of redundancy and invalid solutions reduces complexity.

It is an interesting question whether a complete solution space is necessary. Some restricted layout representations, i.e., slicing and mosaic representations, are complete with regard to their layout classification, are redundancy-free, and

Table 2: Details on common 3D layout representations (to the best of our knowledge). The following characteristics are reviewed. Abstraction (topological representation (TR), room dissection (RD)), spatial resolution (discrete (D), continuous (C)), layout classification (slicing (S), mosaic (M), compacted (C)), operations (O), constraints (C), and features (F). Each characteristic may be not given (□), impossible (☒), impossible (unproven) (✗), possible (☑), or possible (unproven) (✓).

	Abstraction	Spatial Resolution (3rd Dim.)	Layout Classification	Completeness	Cyclic Packings	Placement to Representation	One-to-One Correspondence	Reachability	O: Shift/Move	O: Exchange/Swap	O: Rotation	O: Split (3D)	O: Merge (3D)	O: Local and Global	C: Precedence-/Layer Order	C: Vertical/Boundary/Range	C: Preplacement/Fixed	C: Fixed Outline	C: Number of Circuit Layers	C: Module Layer Assignment	F: Multi-Layer Modules	F: Soft Blocks	F: Rectilinear Blocks	F: Cost Evaluation Support
Sequence Triple [18]	TR	C	C	☒	□	□	✗	$\leq 3n-3$	□	☑	☑	□	□	□	✗	✗	✗	✗	✗	✗	☑	✓	□	✓
Sequence Quintuple [18]	TR	C	C	☑	☑	□	✗	$\leq 5n-5$	□	☑	☑	□	□	□	✗	✗	✗	✗	✗	✗	☑	✓	□	✓
Bounded-Sliceline Grid Array [7]	TR	D	C	✓	✓	□	☒	☑	✓	✓	✓	✓	✓	✓	✓	□	□	□	☑	☑	✓	□	✓	✓
Multi-layer Slicing Tree Structure [2]	RD	D	S	✓	✗	□	☑	✓	✗	☑	☑	✗	✗	✓	✓	✗	✗	✗	☑	☑	✗	✓	✗	□
Multi-layer Sequence Pair [14]	TR	D	C	✓	✗	☑	☑	☑	□	✓	✓	✗	✗	☑	✓	✗	✗	✗	☑	☑	✗	✓	□	✓
Combined Bucket and 2D Array [6]	TR	D	C	✓	□	□	□	✓	☑	☑	☑	✓	✓	☑	✓	✓	□	□	☑	☑	✓	□	□	☑
3D Sub-Transitive Closure Graph [21]	TR	C	C	☑	☑	□	□	✓	☑	☑	☑	✓	✓	✓	☑	☑	□	☑	□	□	☑	□	□	☑
T-Tree [20]	TR	C	C	✗	☒	☑	✗	$\leq 4n-3$	☑	☑	☑	□	□	✓	☑	✓	✓	☑	□	□	☑	□	□	✓
3D Bounded-Sliceplane Grid [17]	TR	C	C	□	□	□	☒	☑	☑	☑	☑	☑	☑	✓	☑	☑	✓	□	✓	□	☑	□	✓	✓
3D Corner Block List [12]	RD	C	M	✓	□	☑	□	✓	□	☑	☑	✓	✓	✗	□	□	□	□	□	□	☑	☑	□	□
3D Slicing Tree [5]	RD	C	S	☑	☒	□	✓	$\leq 10n-6$	□	☑	☑	☑	☑	☑	□	☑	□	□	□	✓	☑	☑	□	□
Sequence Quadruple [10]	TR	C	C	☒	□	□	✗	$\leq 4n-4$	□	☑	☑	□	□	□	✗	✗	✗	✗	✗	✗	☑	✓	□	✓
O-Sequence [13]	RD	C	M	□	✗	✓	☑	□	□	☑	☑	✓	✓	□	□	□	□	□	□	□	☑	☑	□	□
Layered Transitive Closure Graph [11]	TR	D	C	□	✓	□	□	✓	☑	☑	☑	✓	✓	☑	✓	☑	□	□	☑	☑	✓	☑	□	☑
Double Tree and Sequence [9]	TR	C	C	✓	✓	□	☒	☑	☑	☑	☑	✓	✓	☑	☑	✓	□	□	□	□	☑	□	□	✓
Labeled Tree and Dual Sequences [15]	TR	C	C	✓	☑	☑	☒	☑	□	☑	☑	□	□	□	□	□	□	□	□	□	☑	□	□	□
Twin Quaternary Tree [16]	TR	C	M	✓	☑	☑	✗	✓	□	✓	☑	□	□	□	□	□	✗	✓	✗	✗	☑	□	□	✓

exclude invalid solutions. That is, a *one-to-one correspondence* between abstract solutions and geometric packings exists which could be valuable for specific applications. A layout representation featuring a one-to-one correspondence is not necessarily complete but guarantees valid solutions. Practically relevant problem sets (in addition to standard benchmarks) are required to further evaluate restricted layout representations with regard to their industrial applicability.

2.6 Supported Operations (O)

Classical operations on modules (e.g., movement, exchange, or rotation) are supported by almost all layout representations. However, 3D-specific operations like splitting or merging modules are rarely supported.

Based on analyzing their impact on solution quality, differentiating operations into *global operations* (higher impact) and *local operations* (lower impact) is useful. Depending on the balance between global and local operations, the *reachability* (number of required operations to gain a specific solution) may vary. High reachability often corresponds with efficient optimization approaches with regard to speed and quality (see Sec. 4).

2.7 Supported Constraints (C)

In addition to classical constraints, such as symmetry or distance constraints, several new constraints must be accounted for to fully benefit from 3D integration. For example, modules in a 3D layout may occupy several layers. Hence, this needs to be accounted for which is an inherent feature of "real" 3D layout representations. On the other hand, using a fixed number of layers and/or direct layer assignments of modules is supported by layered layout representations, the so-called *2.5D layout representations*.

The comparison in Table 2 is limited to constraints that are inherent in the reviewed layout representation.

2.8 Cost Evaluation Support

A *cost function* contains various *cost terms*, such as wirelength and layout area, and is used to determine the quality of a layout solution. Cost evaluation is typically the most time-consuming operation during layout optimization. Complex cost criteria such as temperature distributions are runtime-intensive compared to geometrical ones (e.g., footprint area, aspect ratio). Efficient layout representations should inherently support cost evaluation. As illustrated in Table 2, some layout representations already provide mesh-like information as for example needed for thermal simulation (e.g., Combined Bucket and 2D Array [6], Bounded-Sliceline Grid Array [7], and 3D Bounded-Sliceplane Grid [17]). Efficient determination of neighbor/adjacency information between modules without generating the complete packing is also valuable for many design decisions. In any case, incremental cost evaluation is preferred over global evaluation in each optimization step (specifically in the inner loop of optimization).

A tremendous speedup is expected if a layout representation would reveal cost-correlating properties, such as topology characteristics. Unfortunately, such correlations have not been discovered yet.

3. SOLUTION SPACE INVESTIGATION

In general, the solution space is a set of all possible solutions to a problem. For the 3D layout problem, this set contains all valid layout configurations. These configurations are associated with some costs and are modeled by using a specific layout representation.

An investigation of any solution space with regard to its most important characteristics, such as redundancy and reachability of different solutions, provides important insights into the related layout representation.

3.1 Requirements

The following requirements must be met in order to enable sampling of the solution space of a layout representation.

Enumeration of solutions. Some layout representations allow consecutive numbering of their abstract solutions. Thus, every abstract solution can be unambiguously sorted.

In order to determine the number of possible solutions, the layout representation is analyzed w.r.t. computability of its different parts. For example, a 3D Slicing Tree uses a

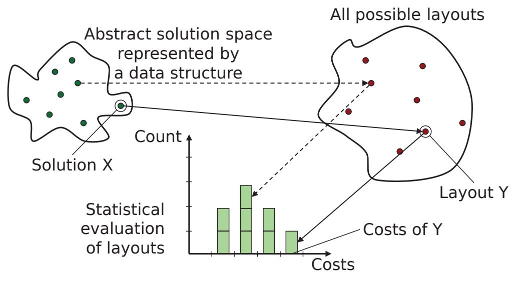
All possible layouts
Abstract solution space represented by a data structure
Solution X
Count
Layout Y
Statistical evaluation of layouts
Costs of Y
Costs

Figure 1: General data sampling approach. An abstract solution X is transformed into the corresponding layout Y. A statistical analysis of the generated layout data can reveal a large variety of useful information (e.g., cost distributions).

binary tree, variations of slicing operations (assigned to inner nodes), permutations of modules (assigned to leaf nodes), and rotation of modules. Hence, the computability of each part is determined. The number of different topologies of a binary tree is described by the *Catalan Number* $C(m) = (2m)!/((m+1)!m!)$, the number of variations of slicing operations (i.e., [X, Y, Z]) is 3^m, the number of permutations of module assignments is $n!$, and the number of rotations (six cuboid rotations in 3D space) is 6^n (n being the number of modules or leaf nodes, $m = n - 1$ being the number of inner nodes). It is possible to generate an unique abstract solution for all topologies of the solution space ($3^{n-1}6^n(2n-2)!/(n-1)!$, details omitted due to limited space).

If a layout representation supports such an enumeration, generating both randomly distributed solutions and the complete solution space is trivial.

Randomly distributed solutions. The described enumeration is difficult to achieve for layout representations where certain solutions are invalid (e.g., due to constraints). At the expense of runtime, appropriate recursive and iterative approaches (e.g., backtracking in graph structures) still allow to generate the complete solution space and to achieve an evenly distributed sampling, respectively.

Cost evaluation. The subsequently described experiments (Sec. 4) require a comparable implementation of the cost evaluation. This is achieved by omitting data-structure-specific evaluation methods, thereby reducing the possible efficiency in some cases. Thus, strongly implementation-dependent runtime comparisons are omitted in our investigations. Hence, we believe that the provided worst-case complexities (Table 1) allow an objective runtime comparison between different layout representations.

3.2 Data Sampling Methods

Next, we present our methods of sampling the solution spaces. These methods allow an investigation of different characteristics of the solution spaces as presented in Sec. 4.

Fig. 1 illustrates a general data-sampling approach. Common to all methods is the generation of an abstract solution and its transformation into a real layout. The layout data is then used to evaluate the solution, e.g., with regard to costs.

Complete sampling of the solution space enables an exhaustive investigation of all desired characteristics. Obviously, this approach is limited to small problem sizes and is therefore not applicable for practically relevant benchmarks. However, some properties, such as redundancy or conformity, encourage a complete investigation of the solution space. Depending on the solution space complexity, complete sampling for problems with approximately 5 modules are feasible (e.g., Sequence Quintuple, 10µs per solution result in two months runtime). An exhaustive storage of all solutions is also challenging due to the huge amount of data (e.g., Sequence Quintuple, 5 modules, approx. 50 terabytes depending on stored information).

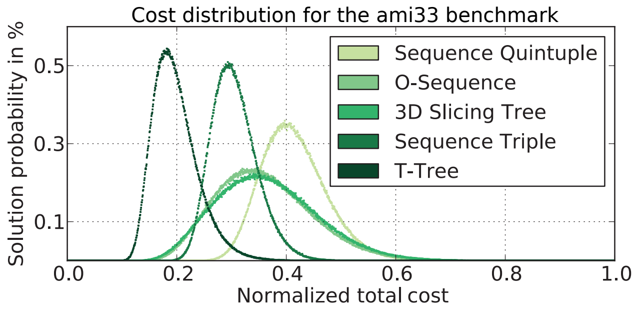

Figure 2: Distribution of total cost for the ami33 benchmark. Despite the strong benchmark dependency, a qualitative comparison of layout representations is possible. Here, minimum-cost solutions of a general stochastic optimization method are best achievable using the T-Tree representation.

Monte Carlo. If complete sampling is impractical, an evenly distributed sampling (i.e., the so-called *Monte Carlo method*) is considered to approximate the solution space. A minimal number of samples is required to allow reasonable predictions. Objective comparisons between layout representations with practically relevant problem sets are feasible, even though not all characteristics can be investigated comprehensively (e.g., redundancy). Among others, the Monte Carlo method reveals the potential of layout representations to be used for stochastic optimization methods, like *Simulated Annealing*, which are relevant for physical-design problems.

4. EXPERIMENTAL RESULTS

Our experiments are limited to specific 3D layout representations that are promising for 3D designs. An extension to further 3D layout representations is straightforward. Representative MCNC benchmarks [3] are expanded to the third dimension by applying a defined z-extension to all modules.

4.1 Cost Distributions

First, the distribution of the solutions with regard to their respective costs is analyzed. This approach reveals characteristics of a specific layout representation such as its applicability to a stochastic optimization method. The Monte Carlo method is applied in order to obtain the distribution for typical sized benchmarks. Two different cost terms are considered throughout the experiments, *estimated wirelength* (HPWL) and *bounding box area.* The latter describes the half of the bounding box surface-area (considering all modules), thus measures the packing density. The *total cost* equals the mean of both cost terms.

One-dimensional histograms allow observing the distribution of a single criterion, in our case total cost. In Fig. 2, the layout representations Sequence Quintuple, O-Sequence, 3D Slicing Tree, Sequence Triple, and T-Tree are compared with regard to the total-cost distribution for the ami33 benchmark. The parameters of these Gaussian-shaped curves are benchmark-dependent but nevertheless allow an application-specific rating of different layout representations. The T-Tree is characterized by the most narrow cost-distribution curve with lowest (mean) total cost. Hence, it provides a good applicability for 3D stochastic optimization problems

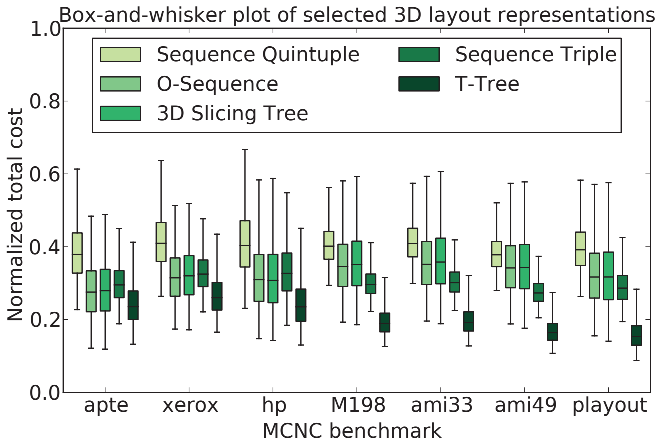

Figure 3: Cost distributions of selected 3D layout representations, applied to representative MCNC benchmarks. Boxes range from lower to upper quartile, with the corresponding median outlined. The whiskers point out the 1st and the 99th percentile respectively. The T-Tree appears to provide best applicability for stochastic optimization methods.

– stochastically chosen solutions are characterized by low (mean) costs, i.e., high solution quality.

In order to generalize this observation, we applied this investigation to 7 MCNC benchmarks, ranging in size from 9 (apte) to 56 modules (playout). Fig. 3 illustrates different layout representations and benchmarks with regard to normalized total cost. Again, the T-Tree layout representation appears to be best tailored for 3D stochastic optimization methods (due to low total cost).

Two-dimensional histograms. Fig. 4 illustrates a solution space in detail. Contrary to Fig. 2, two cost terms are visualized simultaneously. Additionally, we also depict the influence of permutation operations (e.g., exchange, rotation, shift). Here, the dark diamond represents a specific solution, the white dots represent all reachable solutions using the rotation operation only. This way, the impact of specific operations on costs can be analyzed.

Based on this and similar investigations, operations can be grouped into global and local operations. According to our observations, an efficient stochastic optimization method requires a balanced availability of global and local operations, assuring good reachability of different solutions. Using a certain operation to modify a solution results in a specific cost difference. Statistically, this difference spreads over a certain range, where a larger range indicates a global operation. The 3D Slicing Tree offers the most balanced availability of local and global operations when applied to representative benchmarks. We observe that the T-Tree (providing a promising cost distribution) should be combined with more global operations in order to better utilize its cost-distribution advantage when applied for 3D designs.

4.2 Complete Sampling Experiments

Investigations without stochastic variances are feasible when sampling the complete solution space. For example, the reachability of particular solutions, such as the global optimum, can be verified. Two possibilities for complete-solution-space investigations are described next.

Layout-representation components. An important characteristic of a layout representation is the influence of its intrinsic operations on the quality (cost) of the represented

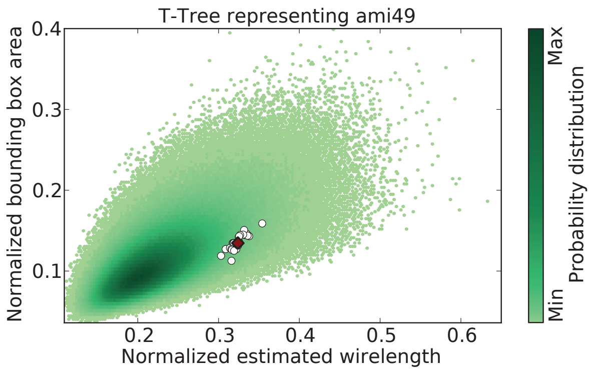

Figure 4: The T-Tree solution space of the ami49 benchmark. Here, two cost terms (estimated wirelength and bounding box area) are considered. Each data point represents a solution. The color indicates the quantity of solutions with identical costs. The dark diamond represents a specific solution to illustrate the influence of its permutation on the solution quality. Further solutions obtained using module rotations are illustrated as white dots.

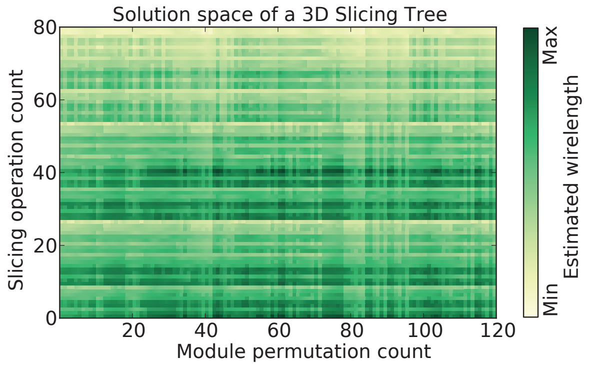

Figure 5: Solution space of a 3D Slicing Tree representing five (arbitrary) modules. The horizontal "line structures" indicate that any variation of slicing operations (y-axis) strongly influences estimated wirelength while module permutations (x-axis) only have limited impact.

3D layout structure. Fig. 5 uses data-structure-dependent axes to illustrate the solution space of a 3D Slicing Tree. Here, the impact on the estimated wirelength of the variation of slicing planes and of the permutation of modules can be reviewed. This plotting method is strongly data-structure-dependent; for some layout representations it is difficult to generate useful permutations. However, if applicable, this method can be used to investigate layout representations in detail with regard to the influence of layout-representation components and their modifications on the solution quality. An important application is to find efficient cost correlations between a layout and a layout representation – one of the most time-consuming operations during layout optimization.

Redundancy. To enable efficient optimization strategies, a solution space with minimal redundancy is required. Solutions are rated redundant if identical geometrical relationships exist between modules. If solution quality only is considered, former identical solutions can become different ones due to (changed) optimization objectives. Redundancy investigations are limited since they require sampling the complete solution space to provide reasonable conclusions.

Table 3: Redundancy and covering of the solution space (for four modules). Due to the completeness of Sequence Quintuple, the number of unique solutions and thus the solution space size is known. All layout representations include the global optimum but reveal different levels of redundancy and exploitation of the solution space.

Representation	Number of Solutions	Redundancy*	Unique Solutions *(C.+Adj.)*	Solution Space *(C.+Adj.)*	Unique Solutions *(Adj. only)*
Sequence Quintuple	127,401,984	4889 X	26,056	100 %	3,745
Sequence Triple	221,184	26 X	8,345	32 %	2,390
3D Slicing Tree	51,840	16 X	3,211	12 %	1,396
T-Tree	21,120	7 X	2,837	11 %	1,347

*compared to unique solutions w. r. t. costs and adjacency *(C.+Adj.)*

A small problem set with only four modules is illustrated in Table 3. Here, *all* solutions of the layout representations Sequence Triple, 3D Slicing Tree, and T-Tree are generated and compared with the complete layout representation Sequence Quintuple. Redundancy is determined by transforming each packing into the corresponding adjacency matrix and comparing this matrix with already obtained solutions (considering rotation and mirroring). As expected, Sequence Quintuple generates a large number of solutions that cover all possible unique solutions (in addition to many redundant ones). Only 0.02% of all Sequence Quintuple solutions are unique with regard to costs and adjacency relations, indicating a high rate of redundant solutions. None of the other layout representations provides nearly as many different packings as Sequence Quintuple. For example, only a fraction of all possible packings are considered by the 3D Slicing Tree and the T-Tree. However, due to low redundancy and inclusion of best solutions, the 3D Slicing Tree and the T-Tree facilitate efficient realizations of stochastic optimization methods best.

5. GUIDELINES & TOOL

Summarizing our investigations, we provide the following conclusions that can be used as guidelines for development and application of future 3D representations.

- Efficient 3D layout representations should enable inherent constraints and operations, while remaining as abstract as possible.
- Layout representations should have a small solution space with minimal redundancy that includes best solutions.
- A balance between global and local operations, the possibility to transform any layout configuration into its abstract representation, and an "easy" reachability between different solutions support efficient optimization strategies (i.e., layout tools).
- Cost evaluation (to determine the solution quality) during layout optimization is typically a very runtime-intensive operation. Therefore, direct correlations between costs and layout-representation characteristics are needed to reduce costs and enable incremental evaluation.

Our presented methodology is available for interested researchers as open-source tool [1]. Hence, researchers will be able to select the best layout representation for each application and to replace individual layout representations with more powerful implementations for improved 3D designs.

Please note that the presented methods are also applicable to 2D layout representations (except for 3D specific characteristics, such as vertical constraints and operations). Thus, improving these representations is possible using our methodology as well.

6. CONCLUSION

3D integration is helping to maintain the validity of Moore's law in today's nano era. Numerous 3D layout representations have been developed recently to make 3D integration accessible to design tools.

Our investigation reveals the untapped potential of several frequently used 3D layout representations. On the other hand, it also reveals open challenges for efficient 3D layout representations and thus relevant obstacles to enable 3D integration in general.

We believe that our investigation methodology, our observations, the resulting guidelines, and the open-source tool (to enable easy analysis of layout representations) are major contributions for the development of efficient 3D layout representations. This work provides a solid base for developing (new) representations that will allow future tools to take full advantage of the extra dimension in 3D integrated circuits.

Acknowledgment. This work was supported by the German Science Foundation (DFG, project 1401/1).

7. REFERENCES

[1] http://www.ifte.de/english/research/3d-design/.
[2] J. Berntsson and M. Tang. A slicing structure representation for the multi-layer floorplan layout problem. *EvoWorkshops 2004, LNCS 3005*, pp. 188–197, 2004.
[3] F. Brglez. A D&T special report on ACM/SIGDA design automation benchmarks: Catalyst or anathema? *IEEE Design & Test of Computers*, 10(3):87–91, 1993.
[4] H. H. Chan, S. N. Adya, and I. L. Markov. Are floorplan representations important in digital design? *ISPD '05*, pp. 129–136, 2005.
[5] L. Cheng, L. Deng, and M. D. F. Wong. Floorplanning for 3-D VLSI design. *ASP-DAC '05*, pp. 405–411, 2005.
[6] J. Cong, J. Wei, and Y. Zhang. A thermal-driven floorplanning algorithm for 3D ICs. *ICCAD '04*, pp. 306–313, 2004.
[7] Y. Deng and W. P. Maly. Interconnect characteristics of 2.5-D system integration scheme. *ISPD '01*, pp. 171–175, 2001.
[8] R. Fischbach, J. Lienig, and T. Meister. From 3D circuit technologies and data structures to interconnect prediction. *SLIP '09*, pp. 77–84, 2009.
[9] K. Fujiyoshi, H. Kawai, and K. Ishihara. DTS: A tree based representation for 3D-block packing. *ISCAS '07*, pp. 1045–1048, 2007.
[10] Y. Kohira, C. Kodama, K. Fujiyoshi, and A. Takahashi. Evaluation of 3D-packing representations for scheduling of dynamically reconfigurable systems. *ISCAS '06*, pp. 4487–4490, 2006.
[11] J. H. Law, E. F. Young, and R. L. Ching. Block alignment in 3D floorplan using layered TCG. *GLSVLSI '06*, pp. 376–380, 2006.
[12] Y. Ma, X. Hong, S. Dong, and C. Cheng. 3D CBL: An efficient algorithm for general 3D packing problems. *MWSCAS '05*, volume 2, pp. 1079–1082, 2005.
[13] H. Ohta, T. Yamada, C. Kodama, and K. Fujiyosi. The O-Sequence: Representation of 3D-floorplan dissected by rectangular walls. *PRIME '06*, pp. 317–320, 2006.
[14] P. H. Shiu, R. Ravichandran, S. Easwar, and S. K. Lim. Multi-layer floorplanning for reliable System-on-Package. *ISCAS*, volume 5, pp. 69–72, 2004.
[15] R. Wang, others, R. Wang, E. F. Young, Y. Zhu, F. C. Graham, R. Graham, and C.-K. Cheng. 3-D floorplanning using Labeled Tree and Dual Sequences. *ISPD '08*, pp. 54–59, 2008.
[16] R. Wang, E. F. Y. Young, and C.-K. Cheng. Representing topological structures for 3-d floorplanning. *ICCCAS '09*, pp. 1098–1102, 2009.
[17] H. Yamagishi, H. Ninomiya, and H. Asai. Three dimensional module packing by simulated annealing. *CEC '05*, volume 2, pp. 1069–1074, 2005.
[18] H. Yamazaki, K. Sakanushi, S. Nakatake, and Y. Kajitani. The 3D-packing by meta data structure and packing heuristics. *IEICE '00*, E83-A(4):639–645, 2000.
[19] B. Yao, H. Chen, C.-K. Cheng, and R. Graham. Revisiting floorplan representations. *ISPD '01*, pp. 138–143, 2001.
[20] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang. Temporal floorplanning using the T-Tree formulation. *ICCAD '04*, pp. 300–305, 2004.
[21] P.-H. Yuh, C.-L. Yang, Y.-W. Chang, and H.-L. Chen. Temporal floorplanning using 3D-subTCG. *ASP-DAC '04*, pp. 725–730, 2004.