Utilizing 2D and 3D Rectilinear Blocks for Efficient IP Reuse and Floorplanning of 3D-Integrated Systems

Robert Fischbach†∗, Johann Knechtel∗, Jens Lienig
Institute of Electromechanical and Electronic Design
Dresden University of Technology, Dresden, Germany
robert.fischbach@eas.iis.fraunhofer.de, johann.knechtel@ifte.de, jens.lienig@ifte.de

ABSTRACT

The reuse of predesigned intellectual property (IP) blocks is critical for the commercial success of three-dimensional (3D) electronic circuits. In practice, IP blocks can be specified as rectangular as well as rectilinear 2D blocks. The 3D equivalent of 2D rectilinear blocks, orthogonal polyhedra, may be utilized for modeling tightly interconnected (sub-)modules placed onto adjacent dies or for design automation of versatile 3D-integrated systems. Such complex block geometries have not been adequately considered until now. We propose a new 3D layout representation that enables native 3D floorplanning of complex-shaped 3D blocks, i.e., orthogonal polyhedra spread onto multiple dies. Furthermore, it can also be applied during 3D floorplanning of both rectangular and rectilinear 2D blocks. In the former case, experiments reveal superior estimated wirelength and packing density compared to previous work.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Layout*

General Terms

Algorithms, Design

Keywords

3D integration, floorplanning, intellectual property blocks reuse, block geometries: orthogonal polyhedra, 3D design, representation

1. INTRODUCTION

Three-dimensional (3D) integration of electronic circuits has recently gained much attention as a promising option to fulfill ever increasing demands on functionality and performance while limiting cost and power consumption [1–5]. In this context, a coarse design style where available 2D intellectual property (IP) blocks are reused is favored in terms of reliability, cost, testability and design effort [6–8]. In practice, hard IP blocks are generally specified as rectangular as well as rectilinear blocks. Thus, the handling of rectilinear 2D blocks or rectilinear 3D block compounds may be required for 3D design, as further motivated in Section 2.1. Design automation of more complex 3D systems such as System-on-Package (SoP), sensor/chip co-design, or micro-electro-mechanical systems (MEMS) stacks [9] may also benefit from the ability to handle rectilinear 3D blocks. However, related approaches require 3D layout representations which are able to efficiently handle such complex-shaped blocks in the 3D solution space.

Prior work on 3D layout representations and floorplanning has overlooked complex-shaped blocks or related block compounds so far. We note that some publications consider (rectangular) block alignment [10–13], which enables simple block compounds. To do so, the related studies account for additional constraints, possibly decreasing efficiency of solution-space exploration. For example, additional edges for constraint graphs have to be determined in an iterative fashion [10]. In any case, most recent work has neglected (to align) rectilinear blocks. Although such blocks have been successfully considered in some 2D representations such as Corner Block List [14], Sequence Pair [15], Bounded Slicing Grid [16] or B*-Tree [17], rarely have they been considered in their counterparts of 3D representations. The sole exception, the (very recent) work by Quiring et al. [18], extended the T-Tree [19] to enable 3D alignment of rectilinear 2D blocks. However, their approach enforces alignment to a common reference point. This means that complex-shaped 3D blocks including offsets between partial (sub-)blocks on several dies cannot be represented.

In the present work, we propose a 3D layout representation called *3D Moving Block Sequence (3D-MBS)*. Our representation enables handling of complex-shaped 3D blocks in a straightforward and efficient manner. However, it is not restricted to such blocks; 3D floorplanning of both rectangular and rectilinear 2D blocks is also possible. Furthermore, as a "real" 3D representation[1], 3D-MBS allows continuous spreading of blocks in the vertical dimension. We believe this approach is useful for next-generation 3D circuits with versatile IP reuse, massive integration densities or for more complex electronic systems like MEMS stacks.

The remainder of this paper is structured as follows. In Section 2, we provide relevant background and motivate the use of complex-shaped 3D blocks. We then discuss our representation in Section 3 in detail. Experimental results are provided in Section 4; we conclude in Section 5 with recommendations on efficient utilization of complex-shaped blocks in 3D electronic systems.

†Currently with Fraunhofer Institute for Integrated Circuits, Design Automation Division, Dresden, Germany.
∗The work of R. Fischbach and J. Knechtel was supported by the German Research Foundation under project 1401/1.

[1]It is common to distinguish so-called 2.5D layout representations and inherent or real 3D representations. In the former case, several instances of a 2D representation are used to represent multiple dies. Vertical block relations are often modeled by additional constraints, likely restricting the flexibility of optimization techniques. For real 3D representations, blocks and their relations are modeled in a continuous (possibly larger) 3D solution space.

2. BACKGROUND AND MOTIVATION

2.1 Complex-Shaped Blocks

As mentioned in Section 1, accounting for rectilinear blocks has been acknowledged as a relevant feature in traditional (2D) physical design automation. Besides handling rectilinear hard IP blocks, related techniques may be helpful to enforce abutment of particular rectangular blocks or to consider special layout constraints [20]. Figure 1b illustrates rectilinear design blocks, which result from fundamental L-, Z-, and T-shapes (Fig. 1a) by assigning a specific (active layer) thickness. Complex-shaped blocks (Fig. 1c), possibly spread onto multiple dies, can be efficiently described by *orthogonal polyhedra (OPa)* — an orthogonal polyhedron is the three-dimensional equivalent of a rectilinear block. Polyhedra in general are geometric solids bounded by flat faces and constructed by rectilinear edges. OPa are special polyhedra where faces meet exclusively at right angles. This simplification correlates well with typical design constraints and may reduce algorithmic complexity while efficiently handling corresponding blocks.

Enabling complex-shaped blocks for 3D floorplanning provides several benefits. First, several studies on 3D integration [11, 12, 21] have shown that partitioning design blocks into (few) sub-modules (carefully placed and vertically aligned among adjacent dies) can help to reduce intra-block as well as inter-block wirelength, latency and thermal-related leakage. Given rectilinear 2D (or dedicated 3D) design blocks, such approaches can be facilitated by modeling rectilinear block compounds, i.e., OPa. Second, utilizing OPa may decrease design complexity for large-scale systems containing (rectilinear) blocks and dedicated inter-die interconnect structures [3, 5, 22] — related blocks and interconnect structures can be abstracted as OPa during early design phases. Note that such a modeling approach is not restricted by the integration technology since characteristics of varied interconnects like through-silicon vias, face-to-face bonds or monolithic inter-die vias can be annotated to the modeled 3D blocks for subsequent consideration. Third, custom vertical-alignment constraints on rectilinear blocks can be directly and efficiently enforced by modeling OPa. Fourth, due to utilization of possibly complex-shaped modules, the design of (future) versatile 3D systems may benefit notably from the capability of handling arbitrary rectilinear 3D blocks. Note that the aforementioned modeling of OPa is out of scope for this paper; sophisticated techniques may be required for appropriate modeling.

2.2 Moving Block Sequence

The Moving Block Sequence (MBS) is a 2D layout representation based on a constructive block-insertion process. Unlike other representations where rectilinear blocks are mainly handled by determining rectangular-block subsets and introducing placement constraints (e.g. [20]), the MBS can directly process such blocks. Therefore, it is an interesting candidate for an efficient 3D representation; our related extension is discussed in Section 3. In the following, the basic concept of MBS is explained for readers convenience; further details can be found in [23].

Based on two sequences π and IP, a constructive process transforms a given abstract solution $MBS = (\pi, IP)$ into a physical layout. Sequence $\pi = (\pi_0, \pi_1, \ldots, \pi_{n-1})$ is a permutation of all n blocks and defines the block-insertion order. Sequence $IP = (IP_1, IP_2, \ldots, IP_{n-1})$ defines for each block π_i one out of four possible insertion positions, as illustrated in Fig. 2a. Note that for the first-to-insert block π_0 the position IP_0 is a fixed special case, i.e., the coordinate origin, thus it is not included in IP. For each insertion position, rules for packing the block towards the coordinate origin are applied as follows (Fig. 2). For positions i or iv, packing is only considered downwards or to the left, respectively. For position ii, packing is performed primarily downwards and to

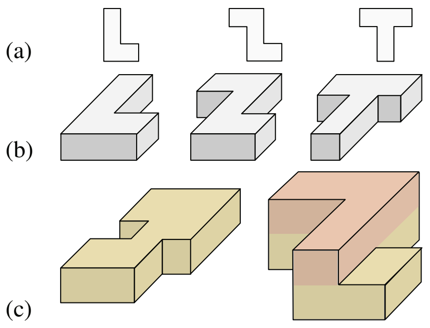

Figure 1: Different types of design blocks. (a) Abstracted blocks in L-, Z-, and T-shaped polygons. (b) Related "real" blocks require a thickness, thus extend the shapes to rectilinear 2D blocks. (c) Complex-shaped blocks, spread onto single or multiple dies, can be described as orthogonal polyhedra (OPa).

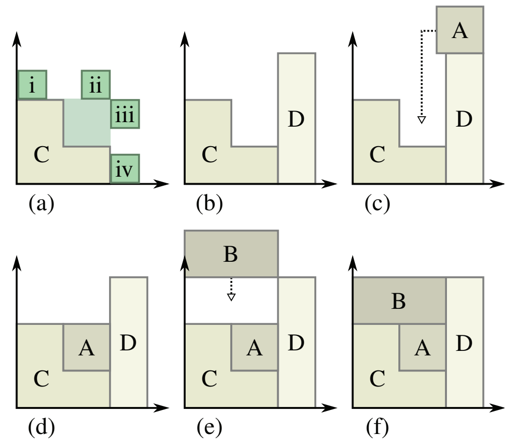

Figure 2: Layout generation of MBS with $\pi = (C, D, A, B)$ and $IP = (iv, ii, i)$. (a) Four different insertion positions are defined. Each position is given specific rules for block packing. (a) to (f) illustrate the stepwise layout generation during constructive block insertion.

the left only if no downward movement is applicable; for position iii, left is the main direction and the downward movement is secondary. Based on the properties of the two sequences, $n! \cdot 4^{(n-1)}$ different abstract solutions exist for n blocks. The layout generation has a runtime complexity of $\mathcal{O}(n^2)$.

Due to the constructive process, no block overlaps occur and only valid solutions are provided. For efficient packing, each block is described by its boundary edges. Given all block edges, considering the edges perpendicular to the packing direction is sufficient for collision detection while inserting a new block. The constructive approach also enables a flexible consideration of additional constraints, such as block symmetries and pre-placed blocks. However, not all possible layouts can be represented. Thus, the MBS is a compacting but not a complete layout representation. Experimental results suggest a similar solution quality compared to common 2D layout representations [23].

3. 3D MOVING BLOCK SEQUENCE

We propose a new real 3D layout representation called *3D Moving Block Sequence (3D-MBS)*, inspired by main features of the MBS (Section 2.2). In particular, we extend the constructive block-insertion process of MBS to the continuous 3D space. Furthermore, the consideration of OPa as well as an efficient collision detection are proposed. Like the original MBS, the 3D-MBS is a non-slicing packing representation.

3.1 Abstract Layout Representation

Similar to the MBS, two sequences defined as follows are used for abstract representation of 3D layouts.

$$3D\text{-}MBS = (\pi, IP') \qquad \begin{aligned} \pi &= (\pi_0, \pi_1, \ldots, \pi_{n-1}) \\ IP' &= (IP'_1, IP'_2, \ldots, IP'_{n-1}) \end{aligned}$$

where n denotes the number of blocks, π defines the insertion order and IP' the insertion positions of blocks. Differing from the MBS, we propose *3D insertion positions* for the 3D-MBS. The positions are denoted as an ordered sequence $IP'_i = C^1C^2C^3 \mid C^i \in \{\lambda, X, Y, Z\}$ of coordinates, where λ represents the omission of the particular coordinate. The twelve applied positions are illustrated in Fig. 3a. By proposing specific block-insertion rules (Section 3.2) for various types of positions, we are able to fully and efficiently exploit the 3D space.

The number of possible solutions is defined by $n!$ permutations of π and $12^{(n-1)}$ variations of IP'; the overall number of abstract solutions is $n! \cdot 12^{(n-1)}$. Besides random generation of both sequences, an initial pair can be altered to obtain new solutions (Section 4). Fig. 4 illustrates an exemplary exchange of blocks.

3.2 Layout Generation

Given an abstract solution *3D-MBS*, the constructive process for layout generation stepwise considers pairs (π_i, IP'_i) where $1 \leq i \leq n-1$. Note that block π_0 is always placed in the corner of the coordinate origin, thus no pair (π_0, IP'_0) is required.

During insertion of block π_i, its position IP'_i defines the initial location and allowed shifting directions. To obtain IP'_i, we keep track of the bounding box covering previously inserted blocks. Each outer corner of this box describes a particular insertion position, i.e., initial location (Fig. 3a). The block-shifting directions (towards the coordinate origin, i.e., packing directions) are restricted by the insertion-position types illustrated in Fig. 3a as follows. The red (dark gray) positions provide only one direction (i.e., either along x, y, or z axis), whereas the green (medium gray) positions offer two degrees of freedom. In that case, the shifting process is partitioned into primary and secondary movement, where the first coordinate defines the primary movement direction. A movement along the secondary direction is conducted only if encountering an obstacle during packing along the primary direction. The three yellow (light gray) positions allow movements comprising all three dimensions. That means a ternary shifting direction is considered when both the primary and secondary direction are blocked. Besides previously placed blocks, the coordinate-axis planes (of the first quadrant) are considered as fixed obstacles in any case. Due to the constructive process, additional constraints such as pre-placed blocks can be easily accounted for.

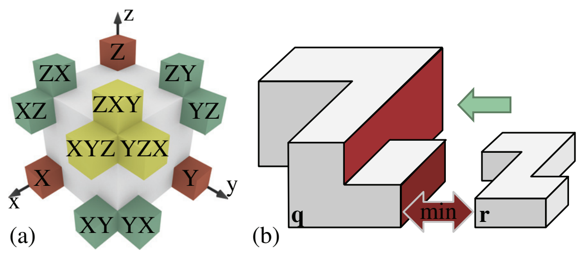

Figure 3: 3D-MBS characteristics. (a) 3D insertion positions. (b) Block shifting and marked faces for collision detection.

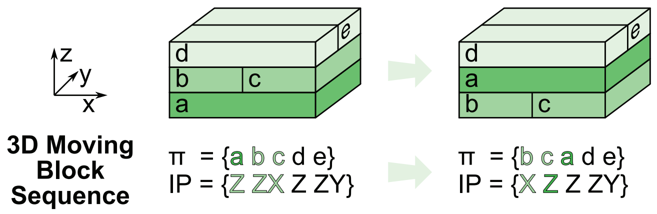

Figure 4: 3D-MBS solutions and corresponding layouts. The exchange of block a with blocks b and c can be accomplished by applying the illustrated modifications to the solution.

The layout-generation process is outlined in Algorithm 1. The obstacle detection, i.e., face-collision detection, is explained in Section 3.3. Note that the runtime complexity of a naive implementation is $\mathcal{O}(c \cdot n^2)$, where c is a complexity factor and collision detection has to consider all previously placed blocks (worst case). The term $c \cdot n$ denotes how many cuboids are required to replace n OPa.

3.3 Collision Detection and Face Segmentation

Block packing is performed in only one direction at any particular moment, even for insertion positions with multiple degrees of freedom (Algorithm 1). Checking for overlaps among a limited subset of block faces is thus sufficient for collision detection, as elaborated in the following. Handling an orthogonal polyhedron is based on partitioning its rectilinear faces into six groups. Figure 5 illustrates such dissection into left (red), right (green), upper (blue), lower (yellow), front (orange), and back (brown) faces. The relevant faces for collision detection are the *boundary faces*, i.e., the inner opposing faces of blocks perpendicular to the packing direc-

Algorithm 1: 3D-MBS Layout Generation

```
Data: number of blocks $n$; block data $B$
Input: block-permutation sequence $\pi$, insertion-position
       sequence $IP'$
Output: layout $L$
1  Insert $\pi_0$ at position $(0,0,0)$ into $L$;
2  for $i \leftarrow 1$ to $n-1$ do
3      Determine bounding box $bb$ of L;
4      Determine initial position $P$ of $\pi_i$ based on $IP'_i$ and $bb$;
5      while shifting towards coordinate origin succeeds do
6          Determine boundary faces $bf_p$ in primary direction;
7          Determine shortest distance $d_p$ of opposing $bf_p$;
8          Perform primary shift of $P$ until $d_p$ is reached
           (some $bf_p$ collide);
9          if degrees of freedom for $IP'_i > 1$ then
10             Determine boundary faces $bf_s$ in secondary
               direction;
11             Determine shortest distance $d_s$ of opposing $bf_s$;
12             Perform secondary shift of $P$ until $d_s$ is reached
               or no more collision of $bf_p$ exists (primary shift
               possible);
13         if degrees of freedom for $IP'_i > 2$ then
14             Determine boundary faces $bf_t$ in ternary direction;
15             Determine shortest distance $d_t$ of opposing $bf_t$;
16             Perform ternary shift of $P$ until $d_t$ is reached or no
               more collision of $bf_s$ or $bf_p$ exists;
17     Insert $\pi_i$ at position $P$ into $L$;
```

tion. Figure 3b illustrates the particular boundary faces of block **q**; assume **q** is already placed and **r** is to be inserted from the right, i.e. moved along the x-axis. Then, the left faces of **r** and the right faces of **q** are checked for collision. To do so, these boundary faces are analyzed for any overlap of their rectangular segments while considering the segments' y- and z-coordinates. In case of some overlapping segments, the shifting amount is defined as the minimal distance min between any pair of related segments. In general, if no segments are overlapping (this does not apply to Fig. 3b), the block is to be shifted towards the coordinate-axis plane. Similarly, block packing towards y- or z-direction considers front and back or upper and lower faces as boundary faces, respectively.

As indicated, the collision detection can be further enhanced by *face segmentation*, i.e., rectilinear faces are segmented into (separate) sets of rectangles. Hence, it is sufficient to analyze pairs of rectangles from different sets in order to detect collisions of related faces. Note that performing segmentation is required only once for each block. We predetermined face segments for our experimental validation (Section 4.1). For practical applications, it may be advisable to leverage efficient computational-geometry algorithms for face unfolding and segmentation, e.g., as proposed by Biedl et al. [24] or Keil [25].

4. EXPERIMENTAL INVESTIGATION

We implement the 3D-MBS for experiments using *Python*; implementations including varied optimization heuristics can be retrieved from [26] (see "Evaluation tool").

In Section 4.1 we validate the capability of 3D-MBS to handle OPa properly during 3D floorplanning. We compare with previous work on 3D representations in Section 4.2. This comparison shows that 3D-MBS outperforms other work when applying various optimization heuristics.

4.1 Floorplanning of Complex-Shaped Blocks

In order to enable 3D floorplanning of OPa, we modify the benchmarks *ami33* (MCNC suite) and *n100* (GSRC suite) [27]. In these benchmarks we replace 8 and 10 (randomly selected) blocks with custom blocks, respectively — six rectilinear blocks along with two OPa in *ami33* (Fig. 6) and 6 rectilinear blocks along with four OPa in *n100*. We also predetermine rectangular face segments for simplified collision detection (Section 3.3).

We implement our 3D-MBS representation along with a *simulated annealing (SA)* engine [28]. In this setup, we focus on the packing capability of the 3D-MBS. We therefore define the SA cost function $f_1 = \alpha_1 * A + \beta_1 * WL_w$ where $A = h_d * w_d$ denotes the (common) die footprint and $WL_w = \sum_n w_n * \big(\sum_d \text{HPWL}(bb_n, d)\big)$ a *weighted wirelength estimate* for all nets n. WL_w is determined using the half-perimeter wirelength (HPWL) metric for net bounding boxes bb_n, where net pins are assumed in the block's geometric center. Boxes bb_n are separately determined for each related die d in order to estimate intra-die routing more

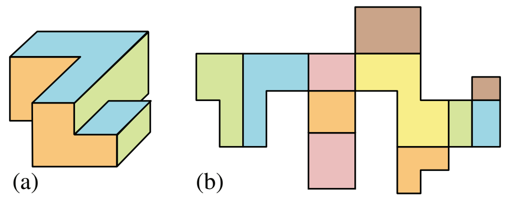

Figure 5: An orthogonal polyhedron (a) can be unfolded into joined rectilinear faces (b). Note that groups of opposite faces cover the respective same area.

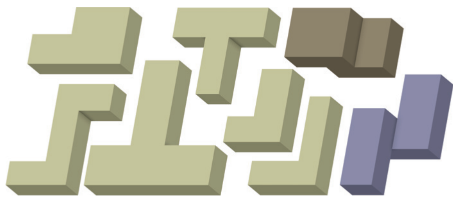

Figure 6: Complex-shaped blocks, for adaption of the benchmark *ami33*. Two new blocks (OPa) spread over multiple dies (🟦 and 🟫), the others are single-die rectilinear blocks (🟩).

accurately. Weight (priority) factors w_n for nets are given in the benchmarks. As SA operations we consider the permutation of π, the variation of randomly selected IP'_i, and the rotation of randomly selected blocks i. The SA engine is configured to consider 3 dies with a common outline. Results are presented in Table 1. Generated floorplans are illustrated in Fig. 7. The SA process, i.e., cost reduction, for an particular optimization run is illustrated in Fig. 8.

Considering these results, we make the following observations. First, 3D-MBS can be successfully applied to 3D-floorplanning problems that includes complex-shaped blocks, in particular arbitrarily rectilinear 2D and 3D blocks. Second, the packing capability scales well with the problem size; results for modified *n100* (100 modules) are comparable to results for modified *ami33* (33 modules) in terms of die utilization (reported as *blocks f.p.* in Table 1). Third, applying SA is a simple yet effective approach. Within few (hundred) iteration steps, cost for both wirelength and die footprint can be reduced down to 50% of the initial value.

Since our 3D-MBS is the first real 3D representation to handle OPa directly, we cannot compare this setup to previous work. How-

Table 1: Final results of five subsequent SA runs. *Blocks f.p.* denotes the sum of block footprints in relation to die footprints. Footprints are reported in μm^2, (weighted) wirelength in μm.

	Criteria	1st	2nd	3rd	4th	5th
ami33	*Die footprint*	560011	592735	596573	569504	588464
	Blocks f.p.	43.1 %	45.6 %	45.9 %	43.9 %	45.3 %
	Wirelength	2483694	2394242	2444321	2541327	2395956
n100	*Die footprint*	93009	95718	93796	98356	97940
	Blocks f.p.	43.9 %	45.2 %	44.3 %	46.4 %	46.3 %
	Wirelength	278552	280264	284582	288954	293002

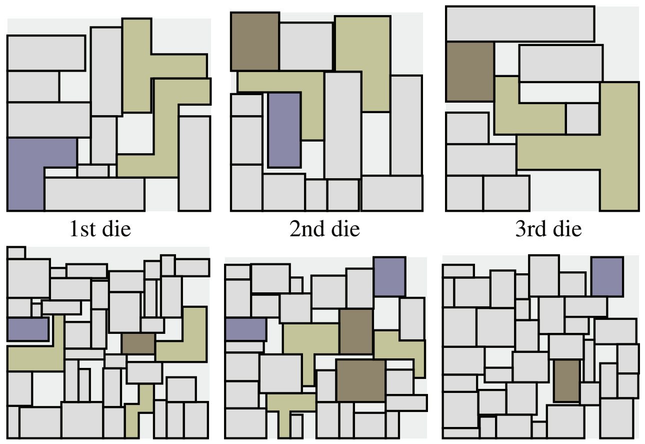

Figure 7: Floorplans for adapted *ami33* (top) and *n100* (bottom) obtained by applying SA for 3D-MBS.

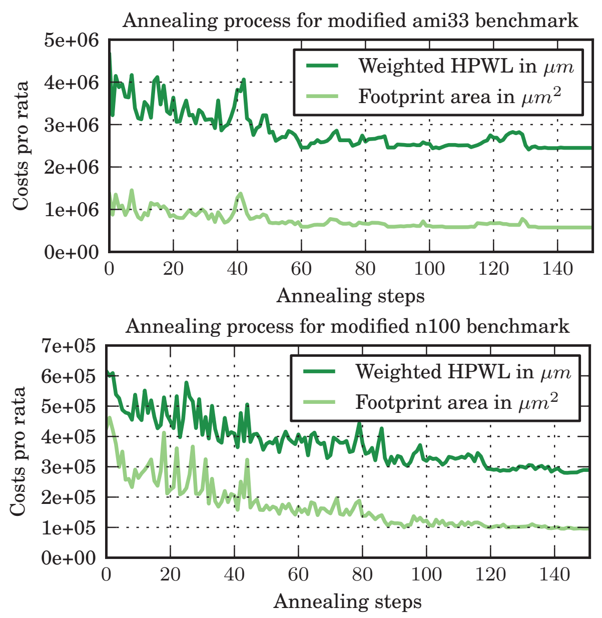

Figure 8: SA process while floorplanning benchmarks including complex-shaped blocks. SA is guided by cost function f_1.

ever, we can apply 3D-MBS also to 3D floorplanning of common rectangular blocks, as discussed in the next subsection.

4.2 Floorplanning of Rectangular Blocks

We apply and evaluate further optimization heuristics besides SA to provide a thorough evaluation of obtainable solution quality. In particular, we leverage concepts for an *evolutionary algorithm (EA)* [29] and the *Great Deluge algorithm (GDA)* [30]. EAs simulate generic mutations and the survival of the fittest in biology. Individuals (i.e., solutions) with favorable features (i.e, low cost) pass their characteristics to the next generation (*selection*). New individuals emerge due to *mutation* or *recombination*. In contrast, the GDA extends the well-known *hill-climbing approach*; this extension is based on a “mountainous” solution space where only “climbers” on mountains above sea level survive. While increasing this threshold value, climbers can also “jump” onto higher mountains, i.e., the search process is not restricted by local maxima. Table 2 provides the main parameters applied in our experiments.

We compare our 3D-MBS to 4 other real 3D layout representations, namely T-Tree [19], 3D Slicing Tree [31], Sequence Triple [32], and Sequence Quintuple [32]. Our implementation of those representations follows the respective publication's description. For the optimization heuristics, only meaningful operations like swapping tree nodes are considered. The applied cost function is defined as $f_2 = \alpha_2 * HESA + \beta_2 * WL_w$ where WL_w denotes the weighted wirelength estimate (Section 4.1) and $HESA = w_s * h_s + w_s * d_s + h_s * d_s$ the *half of enveloping surface area* of the 3D stack where w_s, h_s and d_s denote the die width, die height and stack height, respectively. Thus, $HESA$ measures the 3D packing density. Compared to die footprint or packing volume, we observe that this metric is more efficient in guiding optimization towards a balanced aspect ratio and limited whitespace. We perform 100 full optimization runs for each combination of benchmark and representation, providing detailed cost distributions. (Such distributions are helpful for investigations of the solution quality of 3D layout representations; for further reading refer to [33].) Figure 9 illustrates such distributions for the (original) benchmark *ami33* as box plots; results for further MCNC benchmarks are provided in Table 3. The reported cost f'_2 are based on normalized values, i.e.,

Table 2: Applied optimization heuristics parameters

Heuristic	**Parameter**	**Short description**	**Value**
	α_1	Cost factor for die footprint	0.5
	β_1	Cost factor for wirelength	0.5
	α_2	Cost factor for $HESA$	0.5
	β_2	Cost factor for wirelength	0.5
SA	TStart	Initial temperature	1300
	TEnd	Final temperature	100
	TSteps	Temperature-reduction steps	25
	TSamples	Iterations per temperature step	50
EA	NParents	Parents per generation	8
	NChildren	Children per generation	5–15
	NGen	Generations count	30
GDA	LInit	Initial threshold (normalized)	1.0
	NSamples	Iterations per threshold level	2000
	SVapor	Threshold-reduction value	0.005

$f'_2 * i = \sum_i \alpha_2 / \mathrm{AM}(HESA) * HESA_i + \beta_2 / \mathrm{AM}(WL_w) * WL_{w_i}$ for $i = 100$ optimization runs; AM denotes the respective average means, obtained by analyzing a sufficiently large set of previously determined solutions.

We observe that 3D-MBS allows us to *reduce average cost and cost variability for all benchmarks when applying various optimization heuristics*. This emphasizes the efficiency and flexibility of 3D-MBS for 3D floorplanning of rectangular 2D blocks, despite been tailored to handle complex-shaped rectilinear 3D blocks.

5. CONCLUSIONS

The reuse of versatile IP blocks is critical for industry adoption of 3D-integrated circuits. However, such reuse still faces serious challenges. In particular, hard IP blocks may exhibit rectilinear shapes; handling such blocks is not effectively supported by recent 3D layout representations. Furthermore, vertical alignment of (rectilinear) blocks may be required for several 3D integration scenarios, e.g. placement of partitioned design blocks (Section 2.1). However, recent representations support such alignment only by (iteratively) generating and evaluating additional constraints, which likely impedes efficient optimization approaches.

Our proposed layout representation 3D-MBS facilitates the native handling of orthogonal polyhedra (OPa). It allows us to directly and thus efficiently handle designs containing blocks in any rectilinear 2D and/or 3D shape. This also implies that floorplanning of complex-shaped 3D blocks is supported for the first time. The 3D-MBS is based on the classical (2D) MBS; our approach has the advantage of preserving the ease of understanding and flexibility of traditional MBS while supporting OPa and extended collision de-

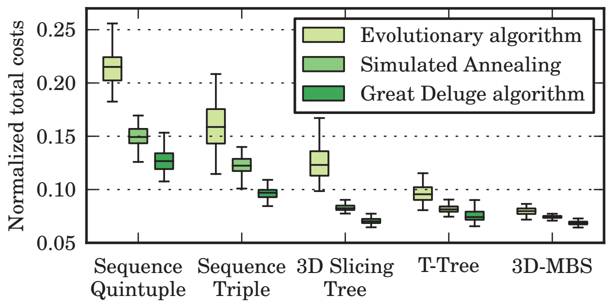

Figure 9: Cost box plots for the benchmark *ami33*. Each box illustrates the interquartile range and the median of normalized cost f'_2, obtained by performing optimization 100 times. The whiskers represent the minimal or maximal cost, respectively.

Table 3: Cost comparison for 3D layout representations (REP) applied to floorplanning of MCNC benchmarks. Cost f_2' are normalized and averaged over 100 optimization runs. SEQ=Sequence Quintuple, SLT=Slicing Tree, SET=Sequence Triple, TTR=T-Tree, Our=3D Moving Block Sequence.

	REP	**apte**	**xerox**	**hp**	**M198**	**ami33**	**ami49**	**playout**	**ø**
Simulated Annealing	*SEQ*	0.077	0.120	0.077	0.148	0.150	0.136	0.087	*0.113*
	SLT	0.042	0.087	0.054	0.082	0.083	0.068	0.047	*0.066*
	SET	0.068	0.112	0.065	0.122	0.123	0.109	0.072	*0.096*
	TTR	0.044	0.099	0.057	0.081	0.082	0.067	0.047	*0.068*
	Our	0.042	0.088	0.055	0.074	0.074	0.060	0.045	*0.062*
Evolutionary Algorithm	*SEQ*	0.142	0.180	0.138	0.214	0.215	0.204	0.185	*0.183*
	SLT	0.069	0.128	0.089	0.122	0.125	0.112	0.086	*0.105*
	SET	0.117	0.160	0.121	0.153	0.160	0.137	0.126	*0.139*
	TTR	0.081	0.122	0.073	0.095	0.096	0.082	0.062	*0.087*
	Our	0.056	0.109	0.062	0.079	0.080	0.069	0.056	*0.073*
Great Deluge Algorithm	*SEQ*	0.099	0.138	0.092	0.126	0.127	0.114	0.075	*0.110*
	SLT	0.043	0.101	0.068	0.070	0.071	0.050	0.033	*0.062*
	SET	0.088	0.133	0.085	0.096	0.097	0.080	0.053	*0.090*
	TTR	0.053	0.115	0.065	0.076	0.076	0.054	0.035	*0.068*
	Our	0.041	0.093	0.059	0.069	0.069	0.049	0.038	*0.060*

tection. The proposed techniques may be extended to furthermore facilitate custom design constraints, like preplacement.

The experimental investigation reveals the effectiveness of 3D-MBS for 3D floorplanning of complex-shaped 3D blocks, even along with rectangular and rectilinear 2D blocks. Furthermore, we show that applying 3D-MBS for 3D floorplanning of solely rectangular 2D blocks allows us to reduce cost compared to previous work. Due to the efficiency of 3D-MBS, this holds true for various optimization heuristics and benchmarks. Both observations suggest 3D-MBS as an attractive representation, enabling the reuse of arbitrary rectilinear IP blocks and related 3D-floorplanning tasks.

6. REFERENCES

[1] K. Banerjee *et al.*, "3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration," *Proc. IEEE*, vol. 89, no. 5, pp. 602–633, 2001. DOI: http://dx.doi.org/10.1109/5.929647

[2] G. H. Loh, Y. Xie, and B. Black, "Processor design in 3D die-stacking technologies," *Micro*, vol. 27, pp. 31–48, 2007. DOI: http://dx.doi.org/10.1109/MM.2007.59

[3] S. Borkar, "3D integration for energy efficient system design," in *Proc. Des. Autom. Conf.*, pp. 214–219, 2011. DOI: http://dx.doi.org/10.1145/2024724.2024774

[4] T. Thorolfsson *et al.*, "Logic-on-logic 3D integration and placement," in *Proc. 3D Sys. Integr. Conf.*, pp. 1–4, 2010. DOI: http://dx.doi.org/10.1109/3DIC.2010.5751451

[5] D. H. Kim *et al.*, "3D-MAPS: 3D massively parallel processor with stacked memory," in *Proc. Int. Solid-State Circ. Conf.*, pp. 188–190, 2012. DOI: http://dx.doi.org/10.1109/ISSCC.2012.6176969

[6] X. Dong, J. Zhao, and Y. Xie, "Fabrication cost analysis and cost-aware design space exploration for 3-D ICs," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 29, no. 12, pp. 1959–1972, 2010. DOI: http://dx.doi.org/10.1109/TCAD.2010.2062811

[7] D. H. Kim, R. O. Topaloglu, and S. K. Lim, "Block-level 3D IC design with through-silicon-via planning," in *Proc. Asia South Pacific Des. Autom. Conf.*, pp. 335–340, 2012. DOI: http://dx.doi.org/10.1109/ASPDAC.2012.6164969

[8] J. Knechtel, I. L. Markov, and J. Lienig, "Assembling 2-D blocks into 3-D chips," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 31, no. 2, pp. 228–241, 2012. DOI: http://dx.doi.org/10.1109/TCAD.2011.2174640

[9] M. Schuenemann *et al.*, "MEMS modular packaging and interfaces," in *Proc. Elec. Compon. Technol. Conf.*, pp. 681–688, 2000. DOI: http://dx.doi.org/10.1109/ECTC.2000.853232

[10] J. H. Y. Law, E. F. Y. Young, and R. L. S. Ching, "Block alignment in 3D floorplan using layered TCG," in *Proc. Great Lakes Symp. VLSI*, pp. 376–380, 2006. DOI: http://dx.doi.org/10.1145/1127908.1127994

[11] R. K. Nain and M. Chrzanowska-Jeske, "Fast placement-aware 3-D floorplanning using vertical constraints on sequence pairs," *Trans. VLSI Syst.*, vol. 19, no. 9, pp. 1667–1680, 2011. DOI: http://dx.doi.org/10.1109/TVLSI.2010.2055247

[12] Y. Liu *et al.*, "Fine grain 3D integration for microarchitecture design through cube packing exploration," in *Proc. Int. Conf. Comput. Des.*, pp. 259–266, 2007. DOI: http://dx.doi.org/10.1109/ICCD.2007.4601911

[13] X. Li, Y. Ma, and X. Hong, "A novel thermal optimization flow using incremental floorplanning for 3D ICs," in *Proc. Asia South Pacific Des. Autom. Conf.*, pp. 347–352, 2009. DOI: http://dx.doi.org/10.1109/ASPDAC.2009.4796505

[14] Y.-C. Ma *et al.*, "General floorplans with L/T-shaped blocks using corner block list," *J. Comput. Sci. Technol.*, vol. 21, pp. 922–926, 2006. DOI: http://dx.doi.org/10.1007/s11390-006-0922-y

[15] K. Fujiyoshi and H. Murata, "Arbitrary convex and concave rectilinear block packing using sequence-pair," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 19, no. 2, pp. 224–233, 2000. DOI: http://dx.doi.org/10.1109/43.828551

[16] M. Kang and W. W.-M. Dai, "General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure," in *Proc. Asia South Pacific Des. Autom. Conf.*, pp. 265–270, 1997. DOI: http://dx.doi.org/10.1109/ASPDAC.1997.600145

[17] G.-M. Wu, Y.-C. Chang, and Y.-W. Chang, "Rectilinear block placement using B*-trees," in *Proc. Int. Conf. Comput. Des.*, pp. 351–356, 2000. DOI: http://dx.doi.org/10.1109/ICCD.2000.878307

[18] A. Quiring *et al.*, "3D floorplanning considering vertically aligned rectilinear modules using T*-tree," in *Proc. 3D Sys. Integr. Conf.*, pp. 1–5, 2012. DOI: http://dx.doi.org/10.1109/3DIC.2012.6263030

[19] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "Temporal floorplanning using the T-tree formulation," in *Proc. Int. Conf. Comput.-Aided Des.*, pp. 300–305, 2004. DOI: http://dx.doi.org/10.1109/ICCAD.2004.1382590

[20] Z. Liu *et al.*, "VLSI fast initial placement with abutment constraints and L-shaped/T-shaped blocks based on less flexibility first principles," in *Proc. Int. Conf. Comm. Circ. Sys.*, vol. 2, pp. 1228–1232, 2004. DOI: http://dx.doi.org/10.1109/ICCCAS.2004.1346396

[21] M. Healy *et al.*, "Multiobjective microarchitectural floorplanning for 2-D and 3-D ICs," *Trans. Comput.-Aided Des. Integr. Circuits Sys.*, vol. 26, no. 1, pp. 38–52, 2007. DOI: http://dx.doi.org/10.1109/TCAD.2006.883925

[22] F. Li *et al.*, "Design and management of 3D chip multiprocessors using network-in-memory," in *Proc. Int. Symp. Comput. Archit.*, pp. 130–141, 2006. DOI: http://dx.doi.org/10.1109/ISCA.2006.18

[23] J. Liu *et al.*, "Moving block sequence and organizational evolutionary algorithm for general floorplanning with arbitrarily shaped rectilinear blocks," *Trans. Evol. Computation*, vol. 12, no. 5, pp. 630–646, 2008. DOI: http://dx.doi.org/10.1109/TEVC.2008.920679.

[24] T. Biedl *et al.*, "Unfolding some classes of orthogonal polyhedra," in *Proc. Canadian Conf. Comput. Geom.*, pp. 70–71, 1998.

[25] J. M. Keil, "Polygon decomposition," *Handbook of Computational Geometry*, vol. 2, pp. 491–518, 2000. DOI: http://dx.doi.org/10.1016/B978-044482537-7/50012-7

[26] http://www.ifte.de/english/research/3d-design/index.html

[27] http://vlsicad.cs.binghamton.edu/benchmarks.html

[28] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. DOI: http://dx.doi.org/10.1126/science.220.4598.671

[29] T. F. Gonzalez, *Handbook of Approximation Algorithms and Metaheuristics*. CRC Taylor & Francis, 2007. DOI: http://dx.doi.org/10.1201/9781420010749

[30] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *J. Comput. Phys.*, vol. 104, no. 1, pp. 86–92, 1993. DOI: http://dx.doi.org/10.1006/jcph.1993.1010

[31] L. Cheng, L. Deng, and M. D. F. Wong, "Floorplanning for 3-D VLSI design," in *Proc. Asia South Pacific Des. Autom. Conf.*, pp. 405–411, 2005. DOI: http://dx.doi.org/10.1145/1120725.1120899

[32] H. Yamazaki *et al.*, "The 3D-packing by meta data structure and packing heuristics," *IEICE Trans. Fundamentals Elec. Comm. Comput. Scie.*, vol. 83, no. 4, pp. 639–645, 2000. [Online]. Available: http://ci.nii.ac.jp/naid/110003208571/en/

[33] R. Fischbach, J. Lienig, and J. Knechtel, "Investigating modern layout representations for improved 3D design automation," in *Proc. Great Lakes Symp. VLSI*, pp. 337–342, 2011. DOI: http://dx.doi.org/10.1145/1973009.1973076