Computer-Aided 3D ICs Layout Design

Katarzyna Grzesiak-Kopec[1] and Maciej J. Ogorzalek[2]

[1]Jagiellonian University, katarzyna.grzesiak-kopec@uj.edu.pl
[2]Jagiellonian University, maciej.ogorzalek@uj.edu.pl

ABSTRACT

Computer-aided 3D ICs layout design requires efficient search of large and discontinuous spaces and no deterministic algorithms are able to perform such a task. The paper presents a new framework for visual kind of intelligent layout design. In the proposed approach a shape grammar generates possible design solutions while intelligent algorithms control the direction of the solution space exploration. Although the method is not limited to a particular design assignment, the paper focuses on a 3D ICs layout problem to demonstrate its potential.

Keywords: CAD, shape grammar, optimization, 3D ICs, floor planning.

1. INTRODUCTION

The problem of appropriate spatial distribution of components, also called packing, is one of the most basic engineering design tasks. It requires efficient search of large and discontinuous spaces which consist of components, objectives and constraints. It is important not only for practical and scientific reasons but for entertainment, like virtual reality modeling, as well. The generation of a sophisticated 3D environment sometimes requires several years of work by a team of specialists, which is both very expensive and time consuming. That is why, the automatic generation of three-dimensional (3D) layout design solutions gains more and more attention, especially in the field of floorplanning of integrated circuits design [22] and architectural design [20].

3D integrated circuits (3D IC) enhance performance improvements by overcoming the boundaries in interconnect scaling. A 3D IC consists of multiple device layers which are stacked together with direct vertical interconnects through them (Through Silicon Vias, TSV). Therefore, it drastically reduces the global interconnect length compared with two-dimensional (2D) chip design. There are also other profits from the third dimension such as:

- Higher: packing density, performance, memory bandwidth,
- Lower interconnect power consumption,
- Smaller footprint,
- Mixed-technology chips support [10].

Floor planning is the earliest and the most critical phase in integrated circuits design. The task is to pack all the given circuit elements in a chip without violating design rules, so that the circuit performs well and the production yield is high. All the circuits elements are rectangular modules of fixed orientation, height and width. They cannot overlap. The minimum bounding box of a packing is called the chip [21]. The problem of integrated circuits layout design in two-dimensional spaces has been solved effectively, however the proposed algorithms cannot be easily transformed to introduce the third dimension. What is more, the problem in 3D is much more difficult in terms of computational complexity than in 2D, and no deterministic algorithms are able to perform such a task effectively and stochastic methods are needed to obtain a globally near-optimal solution.

Existing CAD systems are excellent to hold graphical and geometric information. However, in order to automatically generate plausible designs, they have to register semantic layer information as well. The most desirable solution is to entirely separate these two design representations, namely the graphical and the semantic one [9]. In this manner, not only the method of graphical presentation of a design can be freely replaced with another, but also the layout problem definition can be easily changed without affecting the presentation layer. This kind of approach is very generic but may suffer from limited robustness in real-world design tasks. On the other hand, dedicated systems that solve particular design problem are too specific to use for other domains without significant reengineering.

The technologies for two-dimensional (2D) IC chip layout have advanced significantly and many dedicated commercial CAD tools are available. Generally, floor plans are divided into two categories: the slicing and the non-slicing structure. A slicing floor plan is obtained by recursively bisecting a rectangle using a horizontal and/or vertical line [22]. The solution space is much smaller than in a non-slicing case which implies simpler data structures representation and faster runtime. A non-slicing structure is more general and suitable for most of the real design examples and floor planning algorithms are usually based on simulated annealing technique [8]. Different non-slicing floor plan representations has been proposed like sequence pair (SP) [21], bounded sliceline grid (BSG) [23], O-tree [13], B*-tree [6], corner block list (CBL) [16], and transitive closure graph (TCG) [18].

While considering the 3D floor planning representations, two groups can be identified: the true-3D and the quasi-3D ones [8]. The true 3D approach is to extend the existing 2D representation by the third dimension, like sequence triple [28], 3D TCG [2] or 3D slicing tree [7]. However, the today's 3D IC technology has some important limitations such as a restricted number of device layers and a fixed height of the inter-layer. Thus true-3D representations for the z-axis generates too much redundancy, which is not efficient in both time and space dimension. The quasi-3D floorplanning solutions make use of an array of 2D representations for different device layers, e.g. BSG [9] or TCG [8].

The paper presents the essential features of a new framework for visual kind of intelligent computer-aided 3D ICs layout design that on the one hand remains generic, while on the other hand makes use of the domain-specific knowledge. The framework comprises a shape grammar generative engine and an intelligent derivation controller. Since, the fundamental assumption for the framework was to separate the presentation layer from the semantic one, both the optimization technique and the shape grammar geometric representation are problem independent while goals and constraints vary for different problems. The proposed optimization search algorithm takes the problem formulation and identifies promising solutions by evaluating design alternatives and evolving designs states. To verify the suggested approach, a dedicated application PerfectShape has been developed which architecture is based on the established framework. All the examples presented in this paper are generated with a use of the original software.

2. DESIGN KNOWLEDGE AND SHAPE GRAMMARS

Design environment includes a variety of elements such as goals and constraints, and the current state of a design is dynamically changing. A lot of effort is made in computational design to build a framework that will take the advantage of the dynamic design context during solving a design problem. Furthermore, the design visualization and the way of seeing it is being explored as an indispensable part of the design process and the design thinking [19].

The elaborated framework fits perfectly into these tendencies. The reach design context is moderated by a designer, a shape grammar, design knowledge and a derivation controller. A designer defines a startup design context: a shape grammar production system, design knowledge in the form of constraints and goals and derivation controller parameters. A shape grammar plays a role of a generative engine. It is used by a derivation controller which directs the course of generation according to the specified design knowledge. A designer perceiving the ongoing visual execution can moderate it by tuning all the available context elements, including a shape grammar specification. In particular, new grammar rules using emergent shapes may be introduced. Emergence is acknowledged as a significant cognitive phenomenon of visual reasoning [21]. We call emergent those shapes which are not explicitly represented but emerge from a design structure. As we demonstrate later, shape grammar rules using those elements may drastically discriminate a search space for the 3D layout problem.

The shape grammar theory [27] and applications [18] are well documented and represented in the literature. Shape grammars are generative systems dedicated to specific needs of designers and there are applications of the formalism in the field of art, architecture or engineering design. However, since the formalism suffers from the computational complexity and inferring grammar rules generating only valid designs is in general impossible, it is regarded as being impractical. Notwithstanding, there is an example of a successful application of a shape grammar in industry. It is used at Boeing to route system tubing through an airplane [14]. The fruitful adaptation of this method in industry is of particular importance to motivate development of engineering shape grammars like the one proposed in this paper.

One of the fundamental issues that has to be solved while defining engineering shape grammars is how to connect a grammar and design goals. On the one hand, we may create a knowledge intensive grammar which acts as a sophisticated expert system and generates feasible and functional designs. On the other hand, we may prefer a simple grammar that generates topologically valid but not necessarily feasible solutions and use some external directed search mechanisms [4]. There is a recent trend to implement the second approach and integrate a simple grammar with other methodologies to drive the execution. Among them, the most popular are scripting languages [20] that require programming skills which are non common for designers. There are also some machine learning attempts to overcome the main weakness of the formalism by replacing the manual design of grammar rules with the automated one [11].

In our approach we also propose to use a simple grammar accompanied by stochastic optimization algorithms which are able to navigate nonlinear and multi-modal spaces. After [27], we define a *shape grammar* (*SG*) as a 4-tuple (V_T, V_M, R, I) where:

1. $V_T \neq \phi$ is a finite set of shapes (terminals),
2. $V_N \neq \phi$ is a finite set of shapes (nonterminals, markers) such that $V_T \cap V_N = \phi$
3. $R \subset (V_T \cup V_N)^+ \times (V_T \cup V_N)^*$ is a finite set of rules,
4. $I \in V_T{}^+ \cup V_{N^*}$ is an initial shape configuration (axiom).

Strictly speaking, a shape grammar is a production rule-based system. It derives designs by incremental application of shape transformation rules to some evolving shape configuration. Since all the circuits elements are rectangular modules of fixed orientation, height and width, the only transformation we are interested in is translation. The example of such a simple shape grammar is presented in Fig. 1.

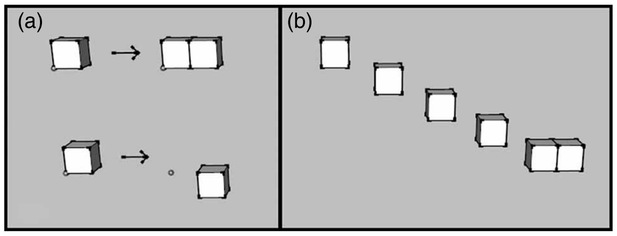

Fig. 1: (a) Sample shape grammar with two rules, (b) 10-step generation example.

Knowledge usually improve the efficiency of the search but in the same time reduce diversity which may be considered as a drawback when novel or unexpected solutions are desirable. Our concept is to encrypt a low level of knowledge in a grammar itself and represent explicit design requirements as goals and constraints [25]. A single *constraint* is a predicate that applied to a shape returns true or false. A generation step can be performed if and only if a derived design meets all the given constraints. That is why, constraints cannot be mutually exclusive. If the constraints are met, all the goals are independently evaluated. A single *goal* is also a predicate but opposite of a constraint it can be satisfied to some extent. The explicit impact factor of a single goal to a design evaluation score is moderated by a derivation controller.

2.1. 3D ICs Layout Design Constraints and Goals

As we have mentioned before, the 3D ICs layout design problem is a kind of the box packing task, where the given set of cuboid modules of fixed orientation have to be placed without overlap in a given area. Of course, this is just the background of the floorplanning task where many other functional and production requirements must be met. In our research we are going to concentrate on one of the biggest advantage of the 3D chips technology which is a wirelength reduction, and on one of the major challenges, which is effective cooling.

For the time being, we are focusing on summarizing those constraints which are geometrical in nature and does not need any additional semantic representation. Fortunately, while investigating the problem it turned out that a lot of them fit into this category (Tab. 1).

Constraint	Description
a) Area constraint	verifies whether a shape is included in a specified area
b) No intersection constraint	verifies whether a shape does not intersect other shapes in the current design
c) Glue shape constraint	assures that a generated solution is consistent
d) Selected neighbor constraint	verifies whether a shape of a selected type is the only one of the type in a specified neighborhood range
e) In layer constraint	verifies whether a shape is in a right layer of the current design, like the boundary layer
f) Selected face constraint	verifies whether neighboring faces of adjoining shapes are of the same type

Tab. 1: Constraints for the 3D ICs layout problem.

In the first place, all the circuit modules must be placed on a given chip (*area constraint*) without overlap (*no intersection constraint*). Taking into account the fact that the bounding box of the chip has to be minimized, a valid layout has to be consistent (*glue shape constraint*). Let us now consider a wirelength reduction requirement. The condition is fulfilled if and only if the chip elements that are connected are as close to each other as possible. It would be best to place them side by side if possible so that relevant faces are adjacent (*selected face constraint*). Thermal management on the contrary, requests separating selected modules to minimize a

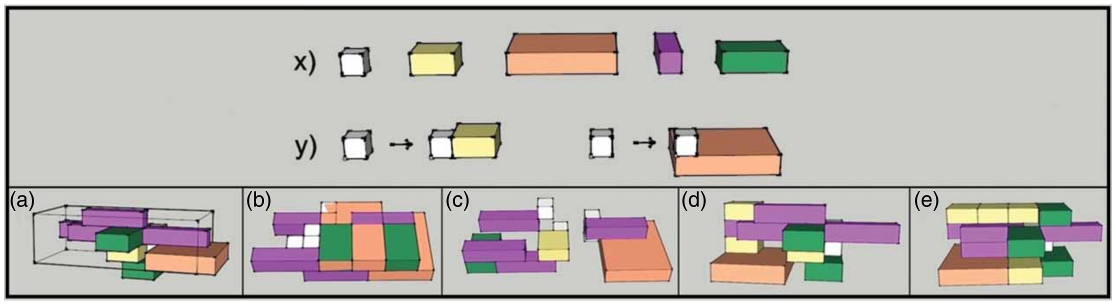

Fig. 2: Designs generated by a simple grammar with constraints: (x) chip modules, (y) rule examples; (a)-(c) designs violating constraints (a)-(c) from Tab. 1, respectively; (d)-(e) designs satisfying constraints (d)-(e) from Tab. 1, respectively.

hot spot problem (*selected neighbor constraint*). Cooling of 3D assemblies also requires settling the most heating components in the outermost layer (*in layer constraint*).

The Fig. 2 presents the constraints from (a) to (e) from Tab. 1. Let us assume that we have to design a 3D layout configuration from the given set of components (x). The example shape grammar rules may look like (y). The design indicated by (a) violates the *area constraint* and is beyond the scope of the given region. The second example design (b) brakes the *no intersection constraint* and some of the chip modules intersect one another. The third layout is no consistent and as such does not conform to the *glue shape constraint*.

The last two designs are the examples of layout designs that meet given requirements. For the layout marked with (d), the *selected neighbor constraint* was defined and the yellow components where selected as the one that are forbidden to be immediately adjacent. In the latter layout (e), the yellow components were expected in the boundary layer.

In order to better visualize, the *selected face constraint* is presented one at a time in Fig. 3. In this example components generated by a simple grammar have colorful faces (a), namely white, blue and yellow. The shape grammar rules are defined in such a way that when imposing a restriction on the grammar that only faces in the same color may be adjacent, there is only one possible design solution (b). When abandoning this limitation, different designs may be generated (c).

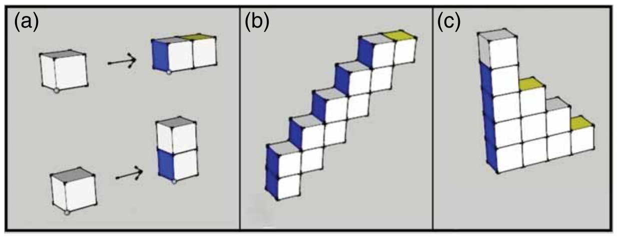

Fig. 3: (a) Simple grammar with two rules, (b) design satisfying a selected face constraint, (c) design violating it.

When all the constraints are met by a generated layout design, goals evaluation is performed. Like in the case of constraints, at this stage of our research, we focus mainly on the goals which are geometrical in nature (Tab. 2).

Goal	Description
a) Minimal space goal	calculates the area occupied by a current design in relation to the expected minimal area
b) Layer goal	evaluates whether components are generated in the expected layers (e.g. boundary layer)
c) Spatial relation goal	evaluates whether components are arranged in an expected way (e.g. aligned vertically)

Tab. 2: Goals for the 3D ICs layout problem.

As stated before, we are looking for the minimum bounding box of a packing (*minimal space goal*). Unlike constraints, which are either true or false, goals are less strict and do not reject imperfect solutions but rather direct our search towards better ones. For example, instead of demanding placing a component in the boundary layer we may prefer to put it as close to the boundary layer as possible, the closer the better (*layer goal*). Defining this goal for the yellow module, the layout design in Fig. 4(a) will have a higher evaluation value than the one in Fig. 4(b). Similarly, rather than require aligning some elements vertically, we may align as many of them as possible, the more are aligned the better solution we obtain (*spatial relation goal*). And likewise previously, having this goal for the yellow module, the layout design in Fig. 4(c) will be better rated than the one in Fig. 4(d).

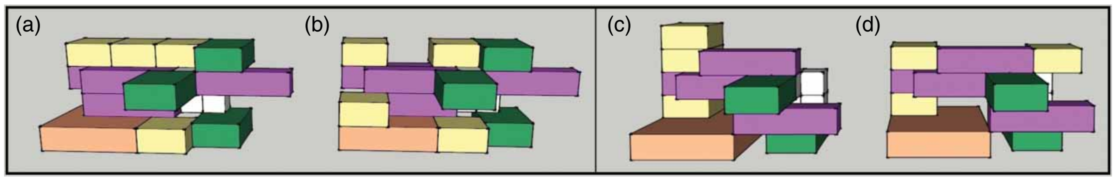

Fig. 4: The goal is to place the yellow modules in the boundary layer: the layout (a) is better than (b). The goal is to align the yellow modules vertically: the layout (c) is better than (d).

3. OPTIMIZATION AND DERIVATION CONTROLLER

Optimization plays a vital role in the design and usability of many engineering products. However, shape grammars were originally presented for architectural design where optimization is usually neglected. The prerequisite for building engineering shape grammars is a directed search techniques selection. There are several approaches for the 3D layout problem, like genetic algorithms, simulated annealing and a hybrid approach using a combination of simulated annealing and expert systems [9].

As already mentioned, a shape grammar is a production rule-based system successively applying shape transformation rules to some evolving shape configuration. Every generation step requires four actions to be made: select a rule, select an embedding, apply a rule and finally decide whether to approve or to reject this step. Usually, the rule is selected randomly, but it may also involve some functional decomposition [1] or machine learning techniques [12,25]. The embedding selection is hardly ever mentioned in the literature. The generation step is committed if the evaluation value of a design increases. Otherwise, the generation step is rollback or decision is left to some computational intelligence method like simulated annealing [3].

In our approach, we propose to select a rule according to some assigned probability which is adaptively changing during the course of derivations. The dynamic rules probabilities values modification is the responsibility of a *derivation controller* and is highly dependent on design knowledge. For the ICs layout problem we have adopted a functional decomposition and identified two main stages of the task: generating all the required components and rearranging them to obtain the best possible solution. Following this plan of actions, we divide all grammar rules into two main functional groups. The first group contains additive rules which introduce new chip elements (e.g. a core) and the second one contains the rules which translate already generated components. In accordance with this strategy, the probabilities of rules generating new circuit elements decrease in direct proportion to the number of elements already generated. The implemented rule selection mechanism is based on a well known, in the field of evolutionary computations (EC), roulette wheel selection. We assign some startup rules weights (corresponding to probabilities), equivalent to a fitness function value in EC, to both groups of rules and decrease them appropriately when new modules are generated. In this way, we not only have some rule selection controlling mechanism, but we reach a goal of generating all the required components as well.

Having a rule selected, we have to decide where to embed it. In other words, we have to indicate a part of the current design we are going to apply the rule to. We have established four different methods of a rule embedding (Tab. 3).

Embedding	Description
a) Execute first admissible	a rule is applied to the first admissible shape configuration, if one is recognized
b) Execute last	a rule is tried to being applied to the part of the design which came into being during the last execution step
c) Execute random	a rule is applied to randomly selected one from the all admissible shape configurations
d) Execute all	a rule is applied to all admissible shape configurations

Tab. 3: Rule embedding methods.

As we can see in Fig. 5, the rule embedding method has an essential influence on the final design. Even though we use a single simple grammar (a) and always a 10 step generation, the achieved results are quite different (b)-(e). The result of the *first admissible embedding* method is highly dependent on the order in which design elements are stored. In the presented example, the generated shapes are inserted at the end of the shape collection. That is why, we can observe such a difference between the *execute first admissible* (b) and the *execute last* (c) routine. In the first case, generated design are much more compact and in the second one more expanded. It is worth noting, that the *execute all* (e) method may be especially useful while generating a fractal like designs like Menger sponge. Depending on a design task, a single method of embedding for all rules may be preferred (like in Fig. 5) or it can vary for different rules, or even for different stages of a single design solution generation.

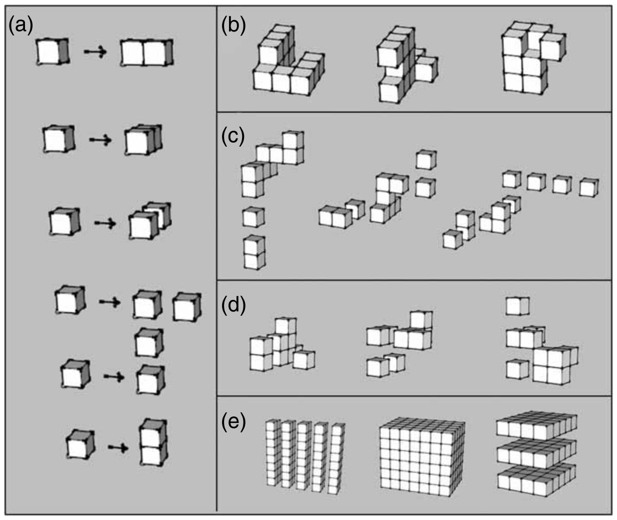

Fig. 5: Ten step generation design solutions for a simple grammar (a) with different embedding methods: (b) execute first admissible, (c) execute last, (d) execute random, (e) execute all.

For the 3D ICs layout problem application we apply a selected rule using the *execute random* method for specified number of tries. After all tries, the application with the highest evaluation value is chosen. If the resulting new evaluation value is higher than the one before the execution, the generation step is committed. If not, we have to decide whether to accept or to reject it. In the meanwhile, we use the simulated annealing to support the decision process. However, in the future work we would like to incorporate some multiagent environment like in [12]. We look for a design solution until stop criteria are met.

Carrying out experiments, we have acknowledged a necessity for implementing a *chain of translation* moves instead of a single one. Let us consider the example in Fig. 6. Suppose that our goal is to create a minimal space design from the given components configuration (a). We can achieve the goal by properly moving one of the cube. One of such a possible three moves sequence is presented by the configurations (b)-(d).

In a chain of translation moves, a randomly selected embedding undergoes a sequence of transformations and the subsequence with the highest design evaluation value is taken into further consideration. This routine does not guarantee finding the optimal design solution, but increases the likelihood of success.

Even more indispensable seems the need for introducing emergent elements into the course of visual computations. Let us again consider the example in Fig. 6. Encountering such a shape configuration, a human-being applying a single swap transformation obtains the solution with the minimal bounding box. She/he easily perceives an emergent cube hole in the design and fills it with the appropriate element. In further research we are going to use this phenomenon extensively to drastically reduce the total search time.

Finally, let us present some preliminary results for a very simple floor planning design task. In Fig. 7 four example design solutions are presented. The task was to spatially arrange given 10 elements to minimize the total volume of the chip. In the first two examples (a)-(b) the additional constraint (*in layer constraint*) was to place all the purple modules in the boundary layer, and in the second example (c)-(d) all the yellow ones.

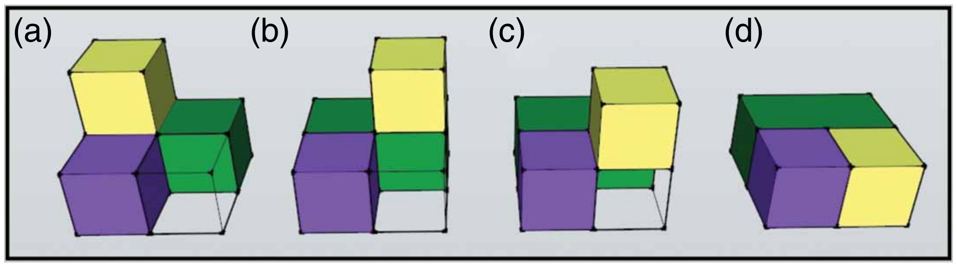

Fig. 6: Startup layout with emergent void (a) and a chain of three translation moves (b)-(d).

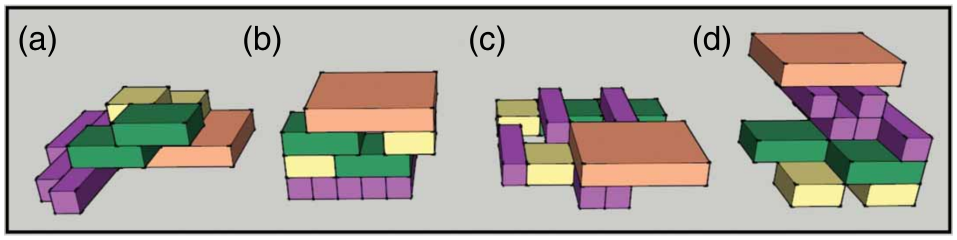

Fig. 7: Example ICs layout design solutions.

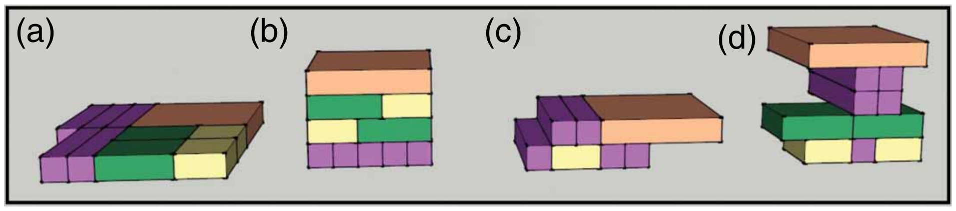

Fig. 8: Example ICs layout design solutions without voids.

In both cases, the required constraint is met, however achieved solutions are too far from being optimal. Such a huge space of possible designs forces the use of more directed search. We strongly believe, that the recognition of emergent holes is the answer. Eliminating all the identified voids from the presented layout design will greatly improve their quality. Their probable versions after applying such a procedure may look as in Fig. 8.

4. SUMMARY AND FUTURE WORK

The aim of the presented research is to build a flexible software architecture framework which will enable solving the 3D layout problem for different engineering design assignments. The proposed solution has to be both methodologically advanced and easily reconfigurable. There are many reasons why the approach is applied to the 3D ICs layout design. First of all, the 3D ICs floor planning effective computer-aided design is not only up-to-date but very challenging one as well. Secondly, the available on the market electronic design automation (EDA) tools are dedicated solutions adjusted to present technology limitations. Most of them are not fully 3D aware but rather adapt 2D algorithms and change only some stages in a design flow (2.5D IC design flow) [26]. And finally, the selected problem design space contains so various goals and constraints that successful application in this domain will confirm the general usefulness of the elaborated framework.

Taking into consideration the floor planning problem statement it appeared that many constraints and goals are geometrical in nature and some of them may be verified visually. This was the reason why we have decided to adopt shape grammars and visual computations to solve the problem. The approach enables us to take the advantage of the diagrams not only in communication but in the cognition process as well. The proposed approach is being verified with a use of a dedicated application PerfectShape. Even though we are at the conceptual stage of our studies, the achieved results are highly promising and we strongly belief that it is worth pursuing. Having the core generating engine developed, now we will concentrate on optimally directed derivations. We intend to involve emergent structures and multiagent environment to support the decision making process. Furthermore, we plan to introduce a semantic layer representation to facilitate wirelength reduction and thermal management in 3D chips. Since the framework deliberately and entirely separates the graphical and the semantic design representations, it may take the advantage of any from the 3D floor planning representations reported in the literature.

ACKNOWLEDGEMENTS

Research supported by FNP under the Program “Mistrz”: “New computational approaches for solving next generation microelectronic design problems”.

REFERENCES

[1] Agarwal, M.; Cagan, J.: A blend of different tastes: the language of coffee makers, Environment and Planning B: Planning and Design, 25(2), 1998:205–226. http://dx.doi.org/10.1068/b250205

[2] Bazargan, K.; Kastner, R.; Sarrafzadeh, M.: 3-D floorplanning: simulated annealing and greedy placement methods for reconfigurable computing systems, Rapid System Prototyping, 1999:38–43. http://dx.doi.org/10.1109/IWRSP.1999.779029

[3] Cagan, J.: Shape annealing solution to the constrained geometric knapsack problem, Computer-Aided Design, 28(10), 1994, 763–769. http://dx.doi.org/10.1016/0010-4485(94)90014-0

[4] Cagan, J.: Engineering shape grammars. In: Antonsson EK, Cagan J (ed) Formal engineering design synthesis, Cambridge University Press, New York, 2001, 65–92. http://dx.doi.org/10.1017/CBO9780511529627.006

[5] Cagan, J.; Shimada, K.; Yin, S.: A survey of computational approaches to three-dimensional layout problems, Computer Aided Design, 34(8), 2001, 597–611. http://dx.doi.org/10.1016/S0010-4485(01)00109-9

[6] Chang, Y.; Chang, Y.; Wu, G. et al: B*-trees: a new representation for non-slicing floorplans, Design Automation Conference, 2000, 458–463. http://dx.doi.org/10.1109/DAC.2000.855354

[7] Cheng, L.; Deng, L.; Wong, M.: Floorplanning for 3-D VLSI design, Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005:

405–411. http://dx.doi.org/10.1109/ASPDAC.2005.1466197

[8] Cong, J.; Wei, J.; Zhang, Y.: A thermal-driven floorplanning algorithm for 3D ICs, in Proc. ICCAD, Nov. 2004, 306–313. http://dx.doi.org/10.1109/ICCAD.2004.1382591

[9] Deng, Y.; Maly, W.: Interconnect characteristics of 2.5-D system integration scheme. In Proceedings of the 2001 international symposium on Physical design (ISPD '01). ACM, New York, NY, USA, 2001, 171–175. http://doi.acm.org/10.1145/369691.369763

[10] Dong, X.; Xie, Y.: System-level cost analysis and design exploration for three-dimensional integrated circuits (3D ICs), In Proceedings of the 2009 Asia and South Pacific Design Automation Conference (ASP-DAC '09), IEEE Press, Piscataway, NJ, USA, 2009, 234–241.

[11] Dehbi, Y.; Plümer, L.: Learning grammar rules of building parts from precise models and noisy observations, International Journal of Photogrammetry and Remote Sensing, 66(2), 2011, 166–176. http://dx.doi.org/10.1016/j.isprsjprs.2010.10.001

[12] Grabska, E.; Grzesiak-Kopec, K.; Slusarczyk, G.: Designing floor-layouts with the assistance of curious agents, Computational Science – ICCS 2006, LNCS 3993, 2006, 883–886.

[13] Guo, P.; Cheng, Ch.; Yoshimura, T.: An O-tree representation of non-slicing floor plan and its applications, Design Automation Conference, 1999, 268–273. http://dx.doi.org/10.1109/DAC.1999.781324

[14] Heisserman, J.; Callahan, S.: Interactive grammatical design, Workshop Notes on Grammatical Design, Stanford, CA, June 24–27, 1996.

[15] Hohmann, B.; Havemann, S.; Krispel, U. et al: A GML shape grammar for semantically enriched 3D building models, Computers and Graphics 34, 2010, 322–334. http://dx.doi.org/10.1016/j.cag.2010.05.007

[16] Hong, X.; Huang, G.; Cai, Y. et al: Corner block list: an effective and efficient topological representation of non-slicing floorplan, Computer Aided Design, ICCAD-2000, 2000, 8–12. http://dx.doi.org/10.1109/ICCAD.2000.896442

[17] Knight, T.-W.: Applications in architectural design, and education and practice, Tech. rep., NSF/MIT Workshop on Shape Computation, 1999.

[18] Lin, J.; Chang, Y.: TCG: a transitive closure graph-based representation for non-slicing floorplans, Design Automation Conference, 2001, 764–769. http://dx.doi.org/10.1109/DAC.2001.156239

[19] Liu, Y.: Some phenomena of seeing shapes in design, Design Studies, 16, 1995, 367–385. http://dx.doi.org/10.1016/0142-694X(94)00001-T

[20] Müller, P.; Wonka, P.; Haegler, S. et al: Procedural modeling of buildings, ACM Trans. Graph. 25(3), 2006, 614–623. http://dx.doi.org/10.1145/1141911.1141931

[21] Murata, H.; Fujiyoshi, K.; Nakatake, S. et al: VLSI module placement based on rectangle-packing by the sequence-pair, IEEE TCAD, 15(12), 1996, 1518–1524.

[22] Nain, R.-K.; Chrzanowska-Jeske, M.: Fast placement-aware 3-D floor planning using vertical constraints on sequence pairs, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 19(9), 2011, 1667–1680.

[23] Nakatake, S.; Fujiyoshi, K.; Murata, H et al: Module packing based on the BSG-structure and IC layout applications, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, 17(6), 1998, 519–530. http://dx.doi.org/10.1109/43.703832

[24] Oxman, R.: The thinking eye: visual re-cognition in design emergence, Design Studies 23, 2002, 135–164. http://dx.doi.org/10.1016/S0142-694X(01)00026-6

[25] Ruiz-Montiel, M.; Mandow, L.; Perez-De-La-Cruz, J.-L. et al: Shapes, grammars, constraints and policies, Proc of the First Interdisciplinary Workshop on SHAPES (SHAPES 1.0), Karlsruhe, Germany, 2011.

[26] Rhines, W.: 3D IC Design Challenges, GSA Memory Conference, San Jose, CA, 2011. Stiny, G.: Introduction to shape and shape grammars, Environment and Planning B: Planning and Design, 7, 1980, 343–351. http://dx.doi.org/10.1068/b070343

[27] Stiny, G.: Introduction to shape and shape grammars, Environment and Planning B: Planning and Design, 7, 1980, 343–351. http://dx.doi.org/10.1068/b070343

[28] Yamazaki, H., Sakanushi, K., Shigetoshi. N. et al: The 3D-Packing by Meta Data Structure and Packing Heuristics(Special Section on Discrete Mathematics and Its Applications), IEICE Trans. Fundamentals, 83(4), 2000, 639–645.