www.cadanda.com
Computer-Aided Design AND APPLICATIONS

3D ICs Layout Hypergraph Representation

Katarzyna Grzesiak-Kopeć[1] and Maciej J. Ogorzałek[2]

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

ABSTRACT

The paper presents a knowledge intensive graphical 3D ICs layout representation in the form of hierarchical layout hypergraphs that enable to demonstrate all possible relations among chip components, especially spatial relations in 3D spaces. The introduction of the hierarchy enables to gather and retrieve information on different levels of details. The proposed graphical knowledge description is also suitable for the grid like neighborhood representation that may be used to divide the circuit into layers.

Keywords: hypergraphs, CAD, 3D ICs, optimization, 3D layout design.

1. INTRODUCTION

The task of 3D ICs floorplanning is a super intellectual challenge. It involves the assembly of millions of components taking into account many different requirements and constraints, just to mention topological, wiring or manufacturability ones. According to Moore's Law, the number of transistors in a typical integrated circuit (IC) design doubles approximately every two years. In 2012 the number of transistors in the CPU went up to an impressive 2.6 billion transistors in Intel's 10-core Xeon Westmere-EX [2]. No wonder that the importance of electronic design automation (EDA) tools rapidly increases.

Computer aided design (CAD) tools are excellent in holding graphical and geometric data. However, it is not enough to provide the complete computer control of the design process. Valid design solutions have to meet various requirements that are very often mutually exclusive, like lightweight and durability or of low cost and of high quality. They also have to fulfill different kinds of constraints, like the ones related to the production process, some aesthetics values or functionality and ergonomics of use. In order to automatically generate feasible design solutions the computational design knowledge representation is needed. Typically, either symbolic or graphical knowledge description may be applied [10]. In the symbolic knowledge description, facts are symbolic terms and the process of inferring involves the manipulation of these terms. While, in the graphical one, the spatial relations between components define the structure of knowledge representation and the way it is exploit. Taking into account the 3D layout problem specification, the latter approach seems to be the natural choice. The paper presents a knowledge intensive graphical 3D ICs layout representation in the form of hypergraphs.

2. LAYOUT SEMANTIC LAYER REPRESENTATION

2.1. 3D ICs Layout Design Knowledge

One of the earliest and the most critical phase in the integrated circuits design process is floorplanning. 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 that cannot overlap. The minimum bounding box of such a packing is called the chip [7].

All the constraints and requirements for this assignment are evaluated on the basis of the proposed spatial arrangement of the chip components. Some of them are straightforward like the no overlapping constraint or the volume constraint defined by the chip dimensions. While the others, namely a wirelength reduction or a hot spot problem minimization need more sophisticated verification. The first requirement is met if and only if the connected chip components are as close to each other as possible, side by side at best. On the contrary, the

Taylor & Francis
Taylor & Francis Group

thermal requirement requests separating the most heating modules and placing them on the chip boundaries to cool them more easily. In order to identify those spatial relations and to enable design process automation, the layout hypergraph representation is introduced.

Hypergraphs are a generalization of graphs, where not only binary but n-ary relations may be represented. The special edges, called hyperedges, can connect not only two, but any number of vertices. Although they are not so popular, they have found many practical applications just to mention architectural design, linear algebra, data mining and information retrieval. Since it is straightforward to map a netlist into a hypergraph, they have also been extensively studied in ICs layout design [6]. Especially in the context of the wirelength connection minimization where the balanced k-way partitioning problem is used. The k-way partitioning assigns vertices of a hypergraph to k disjoint nonempty partitions in such a way to minimize a net cut function and to balance the total vertex weight in each partition. A net cut is the sum of weights of hyperedges that are cut between partitions. The task is known to be NP-hard [4].

Unfortunately, the formerly considered hypergraph representations and partitioning algorithms [1] completely neglect the third dimension in the 3D space. Performing partitioning of the netlist into tiers using only one dimension just like in 2D often fail to give near optimal solutions. In order to overcome the limitations, we propose to adopt the hierarchical layout hypergraphs defined in [5]. In the contrast to commonly used simple hypergraph definition with the only one type of a hyperedge (a set of vertices), layout hypergraphs have two distinguishable kinds of hyperedges. The hyperedges of the first type are non-directed and correspond to chip components which, if necessary, may be examined at the different level of details (transistors or block-level) using a hierarchy relation. The hyperedges of the second type are used to represent relations among chip elements like adjacency or wiring. They can be either directed or non-directed depending on the relation symmetry property. The vertices serve as pin points and, as such, allow to represent wire connections without the need for any additional attributes.

2.2. Hierarchical Layout Hypergraphs

We propose to adapt to our problem domain the hierarchical layout hypergraphs defined in [5]. They were originally defined to represent floor-layouts in architectural design. After [5], we introduce a two step *hierarchical layout hypergraph* (*HG*) definition. We start with the *layout hypergraph* definition and then we define a *child nesting function* that realizes the hierarchy relation.

Let [i] denote the interval {1,2, … i}, where $i \geq 0$ and $[0] = \emptyset$. Let Σ_V, Σ_E be non empty alphabets of node and hyperedge labels respectively, where $\Sigma_E = \Sigma_C \cup \Sigma_R$ and $\Sigma_C \cap \Sigma_R = \emptyset$. A *layout hypergraph* over $\Sigma = \Sigma_V \cup \Sigma_E$ is a system $HG = (E,V,s,t,lb,ext)$ where:

1. $E = E_C \cup E_R$ and $E_C \cap E_R = \emptyset$, is a finite set of hyperedges, where E_C represents object components, E_R represents relations,
2. V is a finite set of nodes,
3. $s:E \to V^*$ and $t:E \to V^*$ assign to hyperedges sequences of source and target nodes respectively, in such a way that $\forall e \in E_C$, $s(e) = t(e)$,
4. $lb = (lb_V, lb_{EC}, lb_{ER})$, where $lb_V:V \to \Sigma_V$, $lb_{EC}: E_C \to \Sigma_C$, $lb_{ER}:E_R \to \Sigma_R$ are labeling functions,
5. $ext:[n] \to V$ is a mapping specifying a sequence of hypergraph external nodes.

The example simple logic circuit without hierarchy and the corresponding layout hypergraph are presented in Fig. 1. One group of hyperedges of the layout hypergraph is labeled by names of the corresponding circuit components (i1, i2, i3, i4, U1, … U10, o1, o2). They represent object components and are depicted in the form of rectangles. The hyperedges

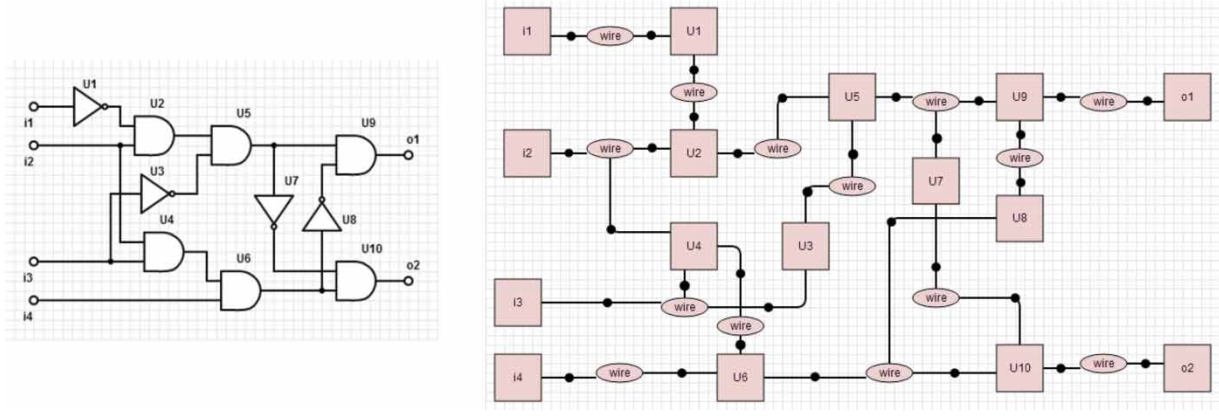

Fig. 1: The example simple logic circuit and the corresponding layout hypergraph.

from the second group reflect the wire relation and are outlined as ellipses. There are fourteen wire hyperedges where four of them connect three components, while the others are binary connections. Nodes specify pin points for potential connections between hyperedges and are drawn as black dots. Each hyperedge has a sequence of source and target nodes assigned. If two sequences are identical, a hyperedge is called non-directed.

Now let us introduce a *child nesting function*. A *hierarchical layout hypergraph* over $\Sigma = \Sigma_V \cup \Sigma_E$ is a system HHG = (HG,ch) where HG is a layout hypergraph and $ch: E_C \to P(A)$ is a child nesting function, where $A = V \cup E$ is called a set of hypergraph atoms and the following conditions are satisfied:

1. one atom cannot be nested in two different hyperedges: $\forall a \in A \forall e_1, e_2 \in E_C \quad a \in ch(e_1) \wedge a \in ch(e_2) \Rightarrow e_1 = e_2$,
2. hyperedge cannot be its own child: $\forall e \in E_C \; e \notin ch^+(e)$, where $ch^+(e)$ denotes all descendants of a given hyperedge e,
3. source and target nodes are nested together with their hyperedge: $\forall e_1 \in E \; \exists e_2 \in E_C \; e_1 \in ch(e_2)$: $\forall v_1$ of s(e) $\wedge \forall v_2$ of t(e) $v1 \in ch(e_2) \wedge v_2 \in ch(e_2)$.

Let us come back to the simple circuit example depicted in Fig. 1 and try to divide it in two layers so that the wire length be minimized and the number of components in each layer be balanced. One of the possible optimal solutions together with its hierarchical layout hypergraph representation is illustrated in Fig. 2. The wire length treated as the distance between cells in a grid cube equals one for each pair of connected components and the number of components in both layers is perfectly balanced. It should be kept in mind that any not immediately adjacent vertical connections may be too expensive in terms of routing resources and delay [9]. Nevertheless, the hyperedge cut between layers in such a way that all components are enclosed in the neighborhood of radius one is more preferred than minimizing a net cut function.

The hierarchy relation enables to analyze the proposed layout at different levels of details. Depending

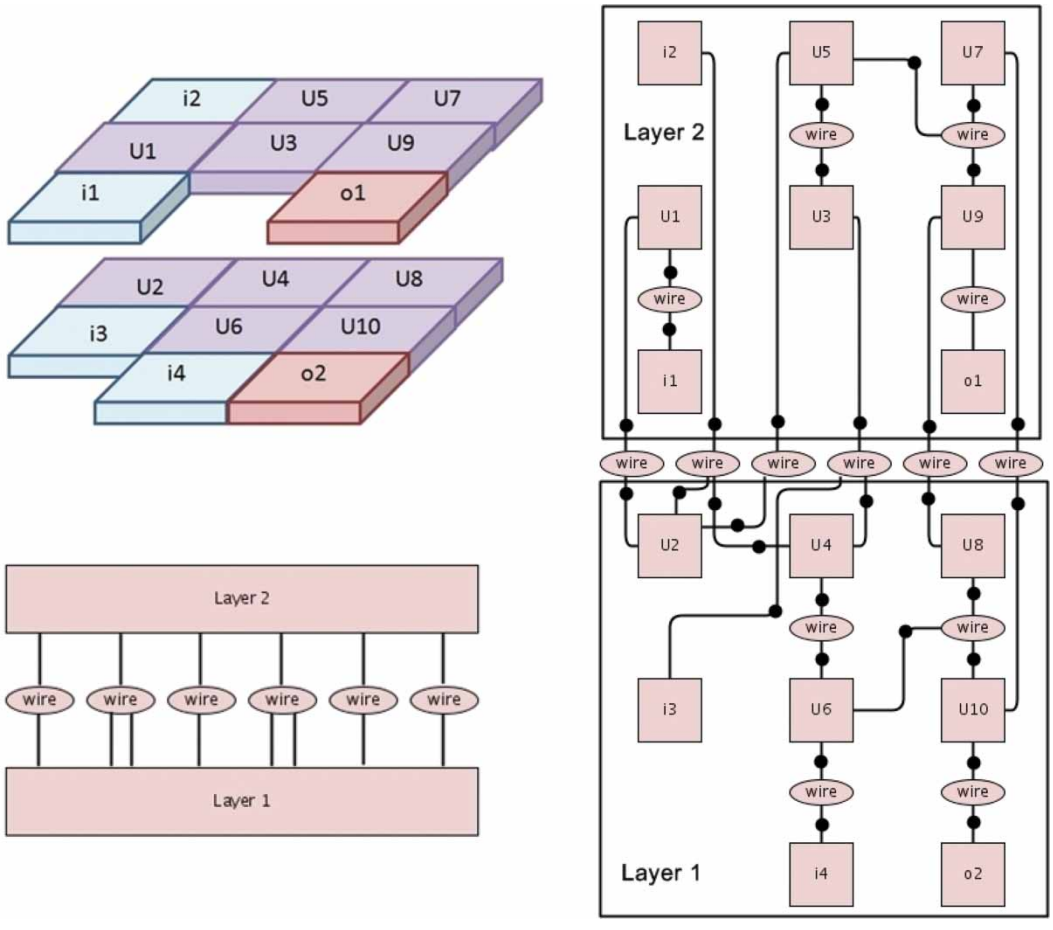

Fig. 2: The example optimal 2-layer partitioning of the circuit in Fig. 1 and the corresponding hierarchical layout hypergraph in the collapsed and the expanded form.

on the currently considering design aspects the hypergraph may be either collapsed or expanded. For example, if the net cut function undergoes examination, the hypergraph can be collapsed only to two hyperedges representing the layers and six wire hyperedges that are cut between them. However, in order to verify whether the partition is balanced, the expanded variant of the hierarchical layout hypergraph is needed.

Additional semantic information may be easily introduced as attributes assigned to different hypergraph elements, namely hyperedges and vertices.

2.3. 3D Partitioning Heuristics

Having realized the huge size of the hypergraphs representing the real-world circuits, the partition heuristics should be extremely efficient in order to be practical. The most popular approaches apply Fiduccia-Mattheyses algorithm [3] which is not only computationally effective (a linear time heuristics) but easily adjustable to different objective functions as well. Nevertheless, as stated before, the commonly applied hypergraph partitioning algorithms in ICs floorplanning to minimize the total wirelength connections do not take into consideration the third dimension in the 3D space. They perform partitioning of the netlists into layers in the same way they work in 2D and use the single dimension. That is why, they often fail to find near optimal solutions. The partitioning heuristics is applied in a top-down recursive manner. The min-cut bisection is used to divide a chip volume geometrically into sectors, while the logic inside the sectors is partitioned topologically. The procedure is recursively repeated until achieved subdivisions are small enough for an optimization algorithm to be solved in a reasonable amount of time [8].

Let us try to apply such a partitioning to the example simple logic circuit in Fig. 1. Originally, the aim is to minimize a net cut function and to balance the total vertex weight in each partition. However, since in the proposed layout hypergraph representation the interpretation of vertices and some hyperedges is novel, the notion of balance has to be specified. For the time

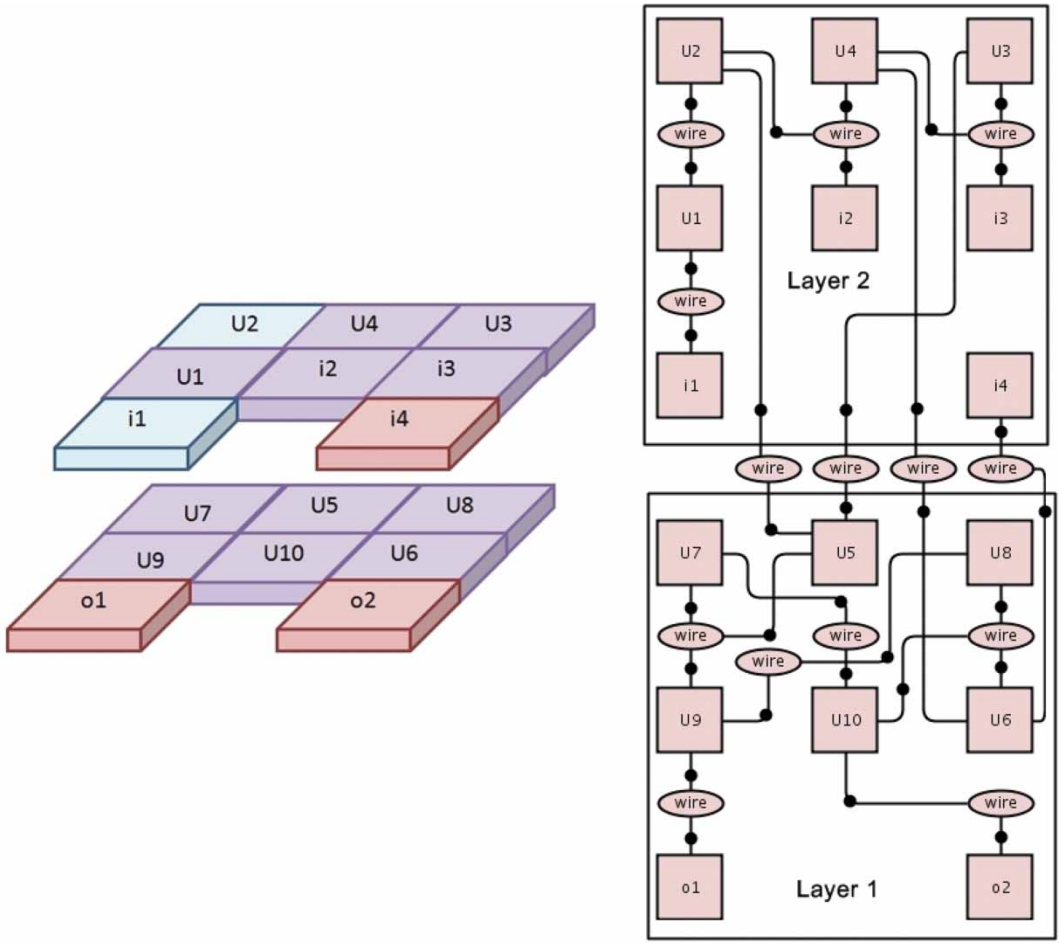

Fig. 3: The example 2-layer partitioning of the circuit in Fig. 1 and the corresponding hierarchical layout hypergraph.

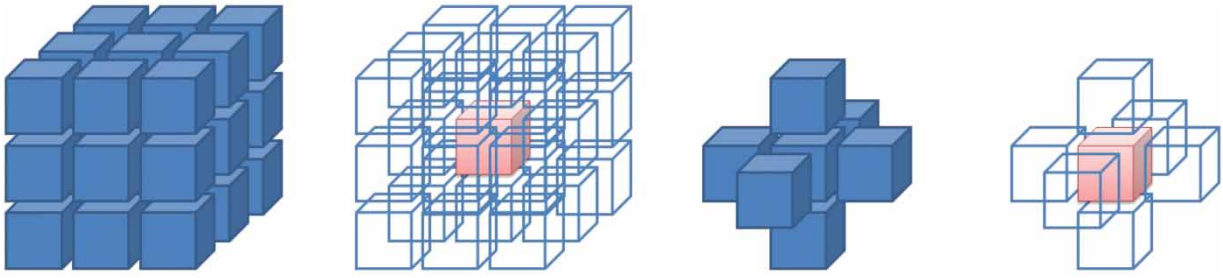

Fig. 4: The grid-like types of neighborhoods: the Moore neighborhood and the von Neumann neighborhood.

being let us assume that the partition is balanced if the total weight of the hyperedges that correspond to chip components is balanced in each layer.

After the balanced net cut function minimization, the total wirelength connections are minimized in each layer separately. A net cut is the sum of weights of hyperedges that are cut between partitions. Let us assume that all the weights are equal to one. The achieved solution may look like the one in Fig. 3. A net cut equals four and the partition is perfectly balanced since there are eight component-hyperedges in each layer. Comparing this partition to the one in Fig. 2:

- the net cut value has been significantly improved from six to four,
- the total vertex weight in each partition has not been changed,
- the total wirelength has increased by one (the length of the connection between U9 and U8 is two).

Hence, two additional objectives are met, but the primary goal is not accomplished. Of course, in this very situation the difference in the total wirelength is not essential but the example is also pretty simple. Scaling the number of circuit elements to millions the difference in the wire length may also significantly increase so that it cannot be disregarded. Proposing 3D partitioning heuristics, the following facts should be considered [1]:

- size – the number of hyperedges representing components may range to as many as several millions, furthermore, the number of hyperedges representing nets is typically between 0.8x and 1.5x of the number of components,
- sparsity – each component has in the average from 3 to 5 connections and the higher numbers occur in a block-level design, also average net sizes varies between 3 and 5,
- granulation – very large nets (hundreds or thousands of components) are very rare.

Taking into account all this factors we propose to perform a 3D topological partitioning of the IC hierarchical layout hypergraph. Instead of searching for the minimal and balanced cut of the hypergraph, the optimal grouping of interconnected 3D neighborhoods is needed. The proposed hierarchical hypergraph representation is ideally suited for such a task.

Depending on the problem representation, the neighborhood may have many different interpretations. For example, taking into account a graph representation, the graph neighborhood of a vertex may be considered. However, we suggest to take a closer look at the grid like structure of a chip and take the advantage of the component's neighbors in a grid. There are two types of neighborhoods, the Moore neighborhood and the von Neumann one (Fig. 4), that are commonly used in the cellular automata evolution on a square grid. The first one is a square-shaped neighborhood and there are 26 immediately adjacent neighbors for a single cell. While the second neighborhood is diamond-shaped and each cell has only 6 immediately adjacent neighbors. Since the average IC net size varies between 3 and 5 even the von Neumann neighborhood is good enough to realize a single net in such a way to minimize the total wirelength. However, the Moor neighborhood allows a more flexible neighbors arrangement and a single hyperedge representing the net connecting 5 components may be realized in the number of distinct 4-element subsets of 26 neighbors.

Hence, let us show how the grid like neighborhood may be represented in the form of layout hypergraph proposed in this paper. In order to represent the Moore neighborhood, it is enough to attach 26

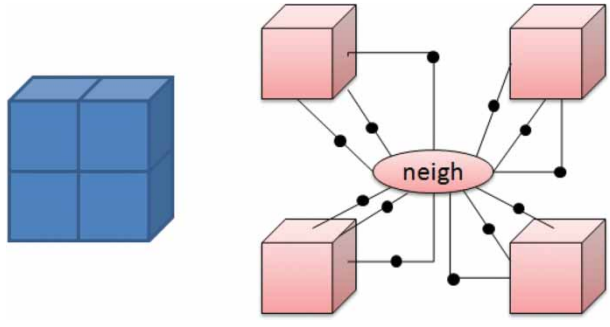

Fig. 5: The example grid like neighborhood and the corresponding layout hypergraph.

attributed vertices to each hyperedge that correspond to a chip component. Each of those vertices may be connected at most to 7 others. Let the hyperedges that represent neighborhood relation be labeled *neigh*. The example grid like neighborhood and the corresponding layout hypergraph are presented in Fig. 5. The group of hyperedges that represent object components are depicted in the form of cubes.

3. SUMMARY

The paper presents a 3D ICs hierarchical layout hypergraph representation that fully reflects possible relations among chip components. Most of all, in contrast to previously proposed 3D ICs hypergraph representations, it naturally resembles spatial relations in 3D spaces. Using labeling mappings allow describing different relations among chip elements and introducing various cost functions, depending on the actual optimization task. Engaging hierarchy enables to gather and retrieve information on different levels of details. In the future, applying hypergraph transformation rules may be used to derive important facts about the generated design solution and the knowledge stored in the layout hypergraph may also be translated into first-order logic sentences [10] describing the design and design requirements that should be met by the hypergraph. In this way, comparing the actual design solution with formulas expressing design goals and constraints the near-optimal design may be obtained.

In the next stage of the research, the layout hypergraph representation will be applied to the task of the total wirelength minimization. The stacking grid like neighborhoods algorithm playing a Tetris like game will be evaluated. Since many ICs floorplanning papers refer to *MCNC benchmark netlists* that are represented in the YAL (Yet Another Language) file format, the parser reading the YAL file netlist into a layout hypergraph has been implemented.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] Ababei, C.; Feng, Y.; Goplen, B.; Mogal, H.; Zhang, T.; Bazargan, K.; Sapatnekar, S.: Placement and Routing in 3D Integrated Circuits, IEEE Des. Test, 22(6), 2005, 520–531, http://dx.doi.org/10.1109/MDT.2005.150

[2] De Gelas, J. Westmere-EX: Intel's Flagship Benchmarked, May 2011, http://www.anandtech.com/show/4285/westmereex-intels-flagship-benchmarked, Retrieved on January 19, 2014

[3] Fiduccia, C. M.; Mattheyses, R. M.: A Linear-time Heuristic for Improving Network Partitions, DAC, 1982, 175–181. http://dx.doi.org/10.1109/DAC.1982.1585498

[4] Garey, M. R.; Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, San Francisco, CA: Freeman, 1979

[5] Grabska, E.; Grzesiak-Kopeć, K.; Lembas, J.; Łachwa, A.; Ślusarczyk, G.: Hypergraphs in Diagrammatic Design, in K. Wojciechowski et al. (eds.), Computational Imaging and Vision 32, Springer, 2006, 111–117, http://dx.doi.org/10.1007/1-4020-4179-9_17

[6] Karypis, G.; Aggarwal, R.; Kumar, V.; Shekhar, S.: Hypergraph Partitioning: Application in VLSI domain, In Proceedings of 34th Annual Conference on. Design Automation, DAC 1997, 1997, 526–529.

[7] Murata, H.; Fujiyoshi, K.; Nakatake, S.; Kajitani, Y.: VLSI module placement based on rectangle-packing by the sequence-pair, IEEE TCAD, 1996, 15(12), 1518–1524, http://dx.doi.org/10.1109/43.552084

[8] Papa, D. A.; Markov, I. L.: Hypergraph Partitioning and Clustering, In Approximation Algorithms and Metaheuristics, 2007

[9] Sawicki, S.; Wilke, G.; Johann, M.; Reis, R.: 3D-Via Driven Partitioning for 3D VLSI Integrated Circuits, CLEI ELECTRONIC JOURNAL, 13(3), 2010

[10] Ślusarczyk, G.: Visual language and graph-based structures in conceptual design, Advanced Engineering Informatics, 26(2), 2012, 267–279. http://dx.doi.org/10.1016/j.aei.2011.10.005