PARTITIONING-ORIENTED PLACEMENT BASED ON ADVANCED GENETIC ALGORITHM BOA

Josef Schwarz
Jiří Očenášek

Technical University of Brno
Faculty of Engineering and Computer Science
Department of Computer Science and Engineering
CZ - 61266 Brno, Bozetechova 2
e-mail: schwarz@dcse.fee.vutbr.cz
e-mail: ocenasek@dcse.fee.vutbr.cz

Abstract: *This paper deals with an adaptation of the genetic partitioning algorithm BOA, based on the estimation of distribution of promising solution, for the placement of the hypergraph nodes into the regular structure of allocations. This task is a simplification of the placement problem encountered in PCB design or more complex case of physical layout of gate array on VLSI chip level. We present top-down placer based on the recursive bisectioning of the hypergraph/circuits. The hypergraph is repeatedly divided into densely connected subgraphs so the number of nets among them is minimized. We compare performance of our placer with the one based on the hybrid GA algorithm and Breuer's force-directed algorithm on artificial and real hypergraphs.*

Key words: *placement problem, hypergraph bisectioning, simple and advanced GA, estimation of distribution, BOA algorithm, Bayesian network, top-down placer.*

1 Introduction

With modern technology the circuits used nowadays become more and more complex and their design must be based on sophisticated methods. In the printed circuit boards PCB and VLSI design process five steps are offered: specification, logic design, physical design, fabrication and testing. The physical design phase is an important part of this process. It can be formulated as a mapping of circuit description into a physical layout in a target technology. The circuit is often represented in the form of a netlist. The resulted layout describes the geometric representation of all components of the circuit and shapes of the interconnection wires. Multiple, competing criteria have to be optimized with a large number of non-trivial constraints. The main concern is to find a layout with routable nets and fulfil constraints on the timing delay of critical nets/signals. The important phase of physical design is the placement of circuit elements/cells into the layout area.

1.1 Survey of heuristics

Many different techniques are used to solve the placement problem. Min-cut placement methods use the known strategy of divide and conquer, recursively applying min-cut bisection to embed the circuit into layout carrier [1], [2], [3]. Numerous other placement methods have been proposed including force-directed [4] as well as simulated annealing [5]. Recently a new enhancement of min-cut partitioning(useful for placement) was published which validates the multilevel partitioning paradigma for hypergraphs with efficient implementation [6]. A renewed bisection and quadrisection method for standard cell placement is described in [7], [8]. Each of the methods can be adapted for many layout styles-PCB, standard cells, sea of gates, gate arrays and macro-cells.

1.2 Fuzzy logic for VLSI CAD

Fuzzy set theory has been applied in many areas of engineering and science. Application of fuzzy controllers in control system and commercial applications are well known. Computer-Aided VLSI design represents a complex hierarchical structure of design phases with multiple competing objectives and various constraints. The majority of algorithms used are heuristics that are based on the human knowledge about the problems. To express such a knowledge we can use fuzzy logic with its linguistic terms. One of the typical application is the multiobjective decision making [9]. In the process of cell allocation a set of rules is investigated resulting in deciding whether the placement including the cell is acceptable. Fuzzy logic is also able to transform the multiobjective function which is naturally a vector function to a scalar function that serves as a simple knowledge for making decision in iterative placement procedures.

Another contribution to VLSI design is in [10] where a Fuzzy partitioning system FPS is described. The partitioning system employs two basic data structures, cells and net array. To each cell the set of associated net is specified and for each net a set of its cells is declared. To each net four parameters are specified: cell-cell connectivity C_c, cell net connectivity C_n, net-associativity N_a as number of cells associated to the net and N_w representing the weight of the net. These four parameters enter in the inference engine which computes the two outputs - cell and net index. This crisp values enter into the classical non-fuzzy procedure which forms initial clusters and then coalesces them into bigger cluster. The FPS system was tested on the benchmark circuits of the ISCAS89 suite. The FPS is comparable to well known Fiduccia-Mattheyses iterative-improvement techniques but only for smaller problem size up to 500 nodes. Because of very low time complexity of FPS algorithm it can serve as a quick procedure producing a good initial solution for another iterative algorithms e.g. for a placer based on the partitioning techniques.

1.3 Genetic algorithms

Genetic algorithms are often used to solve hard discrete optimization problems due to their robustness and ability to find a few suboptimal solutions concurrently what allows to choose a proper solution under additional criterion. The VLSI macrocell placement problem which lies in allocation of macrocells of different shapes and size on the chip carrier minimizing its area appears often in chip layout. To solve this hard problem, placement of objects on the free plane a hybrid genetic algorithms were developed [11], [12] using binary and/or binary slicing tree for encoding of the solution. A simpler problem seems to be the placement of objects into discrete positions of the PCB or gate array. In the past we designed and tested simple [13], [14], and parallel [15] genetic algorithm for PCB layout style with path type chromozome for problem size up to 150 circuits elements. For larger problems we found that it is useful to accept the recursive decomposition of circuit to reduce the problem size. Instead of often used heuristic we attempted to investigate a new approach of bisectioning-oriented placer based on the genetic partitioner BOA published recently in [16].

2 The placement problem

The placement problem solved in this paper is focused namely on PCB layout style which allows to compare performance of our placer with the placer based on the hybrid genetic algorithm [13] and older but very interesting force-directed placement algorithm [4]. Let us note, that the proposed algorithm is adequate for gate array layout too. The problem of circuit placement can be formalized as follows: Let us assume a hypergraph $H=(V,E)$, representing a circuit with $n=|V|$ nodes coresponding to a n netlist elements (cells, gates), and $m = |E|$ edges corresponding to signal nets and a position graph $G=(P,D)$ representing) $p= |P|$ locations and $d = |D|$ edges connecting the adjacency positions with unit distance. The positions (slots) are organized upon the layout styles. We consider the layout style of PCB and gate array, see Fig.1. The set of slots forms a regular structure of locations for circuit elements (represented by circles) with c columns and r rows. The external signals are connected to the connector (represented by rectangles). In case of gate array one segment of connector is assigned to each row and column of locations. In case of PCB only one row of connector segments is used.
The placement problem can be defined as a placement of hypergraph nodes V into a regular structure of discrete locations P in two-dimensional plane called carrier. The goal is to find one-to-one mapping $f_p : V \rightarrow P$ such that the objective function is optimized. The placement is legal if elements are not overlapped and are placed to prescribed locations.

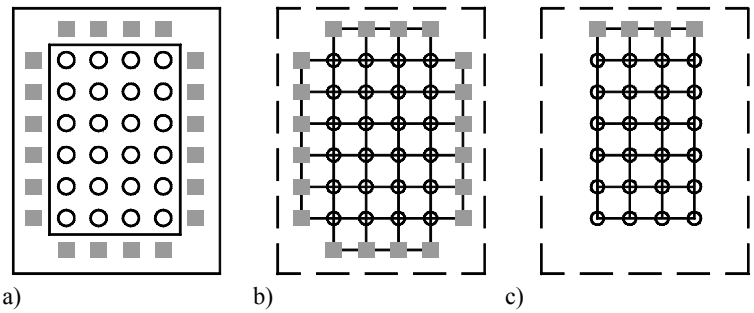

Fig.1 Layout style a) for gate array; the associated position graph b) for gate array, c) for PCB.

In our case the goal is to minimize the total net length. The length of nets is estimated by the half perimeter (HP) of the minimum enclosing bounding box of the placed elements of each net. It is easy to compute and it is often used as standard measure of the net complexity. A better model of the net length is the minimal spanning tree MST.

3 The BOA partitioner

The BOA partitioner [16] for hypergraph bisectioning is based on the BOA implementation published in [17]. The particular bisectioning problem is defined as follows: Let us assume a hypergraph $H=(V,E)$, with $n=|V|$ nodes and $m=|E|$ edges. We are supposed to find such a bisection $(V1,V2)$ of V into equal sized parts ($/V1/ = /V2/$, $V= V1 \cup V2$) that minimizes the number of hyperedges that have nodes in different set $V1, V2$. The set of external hyperedges can be labelled as $E_{cut}\ (V1,V2)$. The objective function is the number of external hyperedges, shortly called cut size:

$$c\ (V_1,V_2)= /E_{cut}\ (V_1,V_2)/ = /\{e \in E / e \cap V_1 \neq 0,\ e \cap V_2 \neq 0\}/ \qquad (1)$$

It can be proven that the sum of half perimeters equals the sum of cut size (canonical set of horizontal and vertical cuts placed between neighbouring columns and rows) which is often used as a criterion in decomposition algorithms [1].

The BOA pertains into the field of Estimation of Distribution Algorithms (EDAs)[18] often called probabilistic model-building genetic algorithm, where crossover and mutation operators are replaced by probability estimation and sampling techniques. It uses statistical information contained in the set of selected parents to detect gene dependencies. The estimated probability model is then used to generate new promising solutions according to this distribution. The process can be described as follows:

Generate initial population of size N (randomly);
While *termination criteria is false* **do**
begin
Select parent population of M individuals according to fitness function (M≤N);
Estimate the distribution of the selected parents and construct the Bayesian network BN;
Generate new offspring according to the estimated model and BN network;
Replace some individuals in current population by generated offspring;
end

In BOA the Bayesian graph/network encodes the structure of a problem. The bisection of the hypergraph is encoded as a binary string $X=(X_0, X_1, .., X_{n-1})$. Variable X_i be equal to zero or one according to assigning of *i-th* node of hypergrah to one or second partition. The variables are not independent due the phenomenon of epistasis. For each variable X_i it is defined a set of variables Π_{X_i} called parents it depends on, so the probability distribution of individuals is encoded as

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i \mid \Pi_{X_i}\right) \qquad (2)$$

As an illustration the six-nodes elementary hypergraph is considered for the bisectioning and one associated BN network is shown in Fig. 2. Using a contingency table (including bivariate marginal probability of variables in current population) each variable using conditional probability can be generated sequentially beginning from X_1 to X_6.

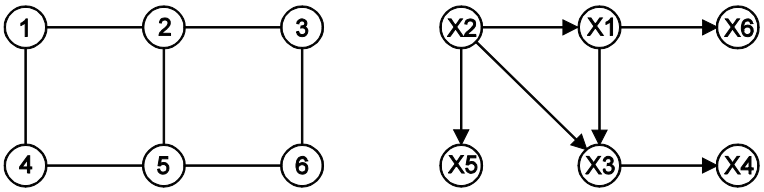

Fig.2. A simple bisectioning problem (left) and one of its Bayesian network (rights).

Notice, the existence of oriented edge from X_j to X_i in the network implies that the variable X_j belongs to the set Π_{X_i}. To reduce the space of possible networks, the number of incoming edges into each node is limited to k=3.

4 The top-down recursive placer

The idea of top-down recursive bisectioning-oriented placement is based on repeated division of a given circuit/hypergraph into subhypergraphs to optimize a given bisectioning objective e.g. cut size $c\ (V_1,V_2)$. With each bisectioning of the circuit, the given layout area/carrier is bisected in either the horizontal or the vertical direction. Each subhypergraph is assigned to a partition. This process is repeated recursively until the subgraph has only one node/elements - the element can be mapped to an unique position on the carrier. When bisectioning a subhypergraph it is necessary to account internal nets in the current subhypergraph as well as the external

connector segments and other elements in another higher-level bisection. This concept called terminal propagation was involved by Dunlop and Kernighan [19] that adds to the current netlist dummy elements that are fixed in the currently processed bisection. In our approach these elements remains in the centre of their higher-level block and are included in the calculation of half perimeter of all nets associated to current element. The top-down recursive bisectioning algorithm is described as follows:

Push the basic block representing original placement problem onto the queue;
While *queue of blocks is not empty* **do**
begin
Pull the block from queue;
if *Block is trivial (includes only one element)* **then**
Map the element to unique position ***else***
begin
Specify internal netlist and dummy elements;
Apply BOA bisectioner for block bisectioning into two subblocks;
Push each block onto the queue;
end
end

In Fig.3 the process of recursive bisectioning of layout area is shown. The original block is divided by vertical cut into two blocks B1, B2. In the blocks two free elements V and W are shown. The block B1 is divided into block B11 and B12 by horizontal cut. During the cutting the element W is taken as dummy element and placed in the centre of B2. When block B2 is divided by horizontal cut into blocks B21 and B22, the element V in the centre of B11 serves as dummy element that influences the shift of the element W into block B21. Finally the division of block B11 into B111 and B112 is shown resulting in the shift of the element V into block B112 closer to element W. The effect of dummy modules in our implementation is replaced by minimization of HP associated to elements V and W.

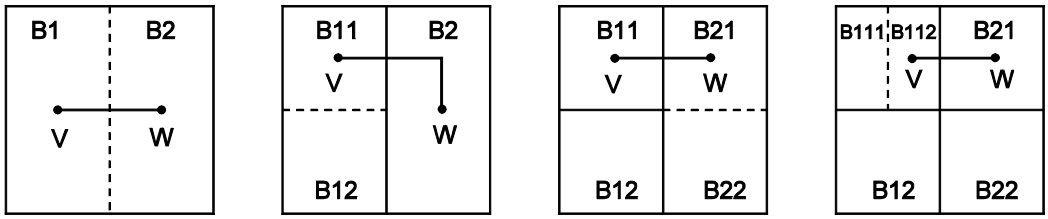

Fig.3 Example of cut sequence and dummy elements phenomenon.

5. Experimental results

5.1 Test graphs

The two types of graph structures are used:

1. Grid graphs *Rn, Rn.B* with grid structure [2], [16] where the notion *n* specifies the number of nodes and *B* the existence of bottleneck in graph structure. As an example a grid graph *RL100* is represented in Fig.4a and *RL100B* in Fig.4b having a 2-edge bottle-neck in the edge structure.
2. Hypergraphs representing real circuits labelled by *ICn* [4], [16], [20]. The global optima is not known. The hypergraphs can be also specified by pair nodes/edges*: IC67/138, IC151/419.*

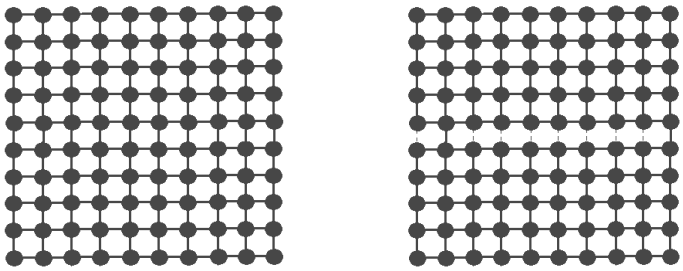

Fig.4 Grid graphs a) RL100 with full grid structure, b) RL100B with the bottleneck

The parameters of the placement problem solved are in the Table 1. The fixed elements are placed into the

connector segments. The number of the rows includes one row of the connector segments.

Circuit	Parameters					
	Elements	Free elements	Nets	Connector segments	Rows	Columns
RL64	64	56	112	8	8	8
RL64B	64	56	106	8	8	8
RL100	100	90	180	10	10	10
RL100B	100	90	172	10	10	10
IC67	67	52	138	15	5	15
IC151	151	136	419	15	11	15

Table.1. Parameters of placement problems.

5.2 Performance of the placer

To test the BOA placer some experiments were done. Six circuits was used to compare the performance, see Table 2, of our BOA placer with a classical but sofisticated genetic algorithm GPLACE [13] and force-oriented placement algorithm [4]. The results of FORCE algorithm are relevant to parameter epsilon=0.001 which specifies very small final level of the force system equilibrium leading to very good results. The time complexity is expressed only by number of cycles; each cycle represents one step of the numerical solution of a system of n nonlinear equations that models the shift of n nodes along the free plane. For each benchmark five independent trials were done and the average values are presented.
The GPLACE genetic algorithm was performed with best known parameter values: in case of graphs IC67 and IC151 the number of population $N_p=30$, offspring rate $R_o=0.3$, mutation rate $R_m=0.02$ are used and inversion operator was switched off. Each tenth generation ten pairwise exchanges of genes leading to cost improvement are done. The stopping criterion is activated when during the epoch of 500 generation the total length of nets represented by HP is not changed. In case of regular graphs the inversion operator an dynamical setting of mutation must be activated. The BOA partitioner as the genetic core of BOA placer is set in the following way: the population size N is taken from the interval <650, 1500> as a minimal successful value with respect to the conclusion done in [16], the 50% selection truncation is used ($M/N= 0.5$), the maximal degree of BN nodes $k= 3$. The experiments were done on the 100MHz PC computer.

	GPLACE					FORCE		BOA			
Circuit	n	HP	MST	Ng	TIME	MST	Cycles	HP	MST	Time	N
RL64	64	126	126	4818	5	-	-	112	112	3.0	800
RL64B	64	127	127	10400	10	-	-	106	106	2.6	650
RL100	100	180	180	33840	26	-	-	180	180	18	1600
RL100B	100	188	188	41443	32	-	-	172	172	10	1000
IC67	67	639	709	1254	2.5	713	184	652	734	3.7	650
IC151	151	2227	2304	2087	17	2025	388	2060	2105	64	1000

Table 2 Comparison of the placers : n - number of nodes (size of problem), N_g - number of generations, N-population size for BOA algorithm, HP - half perimeter (cost), MST- minimal spanning tree, TIME - computational time in minutes, cycles - number of steps of the numerical optimization process.

6 Conclusions

We have developed a top-down placer based on the BOA recursive partitioner. From Table 2 it is evident that the BOA placer produces global optimum for grid graphs that serve as hard benchmarks. For the hybrid genetic algorithm GPLACE it is difficult to find the global optimum for the grid graphs and all the sofisticated tools of algorithm have to be activated, namely the dynamical setting of the mutation and the inversion operator. In case of random logic - hypergraphs IC67 and IC151 the results are comparable in MST value for all the methods. The time consumption of BOA placer depends namely on the problem size and is mainly influenced by the first bisectioning of the original hypergraph. The time complexity of the BOA placer can be estimated as $O(n^{5/2})$ for n<100. Let us notice that the only minimization of MST value can cause wire congestion in some local area of the layout - it holds namely for the GPLACE and FORCE algorithm. From the nature of BOA placer follows that the probability of the congestion is minimized due to minimization of cut size between each pair of currently divided blocks. Our contribution can be seen namely in the extension of the BOA application and acknowledgement of BOA robustness in optimization of discrete combinatorial problems. The main problem still lies in proper choice of cut lines - we used a slightly modified standard approach based on the periodical change of vertical and horizontal cuts. The future work will be focused namely on an exploration of the BOA or BMDA placer to a quadrisection version which can solve cut ordering problem. Other activities will be directed towards the usage of BOA placer for the other layout styles.

Acknowledgement

This research has been carried out under the financial support of the Research intention No. CEZ: J22/98: 262200012 – ”Research in information and control systems” and it was also supported by the Grant Agency of Czech Republic grant No. 102/98/0552 ”Research and Application of Heterogeneous Models”.

References

[1] Breuer M. A.: Min-cut Placement, Journal of Design Automation and Fault Tolerant Computing, Vol. 1, No.4., Oct.1977, pp. 343-362.

[2] Schwarz J.: Design Automatization of Composition and Placement of Integrated Circuit. PhD. Theses, Technical University of Brno, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, Brno, 1993, 136 pp., in Czech.

[3] Fiala P.: Decomposition Placement Methods, Diploma Project, UIVT, FEI, VUT Brno, 1992, in Czech.

[4] Breuer M. A.: Forced-oriented Placement Algorithm, Experimental Study with IC Benchmarks, Chapter 5 and 6, pp. 89 - 162, delivered under personal request.

[5] Schwarz, J.: Simulation oriented placement methods for VLSI chips, In: Proceedings of the MOSIS'93 conference, MARQ Ostrava, Olomouc, 1993, pp. 295-300.

[6] Karypis, G., Kumar, V.: Hmetis - A Hypergraph Partitioning Package, version 1.5.3, University of Minnesota, Department of Computer Science&ENGINEERING, Army HPC Research Centre Minneapolis, http://www-users.cs.umn.edu/~karypis/metis/hmetis/download.shtml.

[7] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Can Recursive Bisection Produce Routable Placements?", to appear in Proc. Design Automation Conf., Los Angeles, June 2000.

[8] D. J. Huang and A. B. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective", Proc. ACM/IEEE Intl. Symp. on Physical Design, Napa, April 1997, pp. 18-25.

[9] Sharagowitz E., et al.: Application of Fuzzy Logic i Computer-Aided VLSI Design, IEEE Transactions on Fuzzy Systems, Vol. 6, No.1, February 1998, pp.163-172.

[10] Manzoul M. A.., Valavala K.R.: A Fuzzy Partitioning System, IEEE Micro, December 1995, pp. 1-8.

[11] Esbensen H.: A genetic Algorithm for Macro Cell Placement, Proceedings of the European Design Automation Conference, pp. 52-57, September 1992.

[12] Schnecke V.: Hybrid Genetic Algorithm for Solving Constrained Packing and Placement Problems, PhD. Theses, University of Osnabrueck, pp. 1- 186.

[13] Bartoš J.: Genetic Algorithm for the Placement Optimization, Diploma Project, Technical University of Brno, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, Brno,1994, 66 pp., in Czech.

[14] Schwarz J.:GPLACE-Genetic Algorithm for Placement Optimization. I. International Conference of Genetic Algorithm on the Occasion of 130th Anniversary of Mendel’s Law, Brno, September 26-28,1995, pp.139-144.

[15] Schwarz, J.: Experimental Study on Parallel Genetic Algorithm for Placement Optimalization, Mendel '97, Technical University of Brno, Faculty of Mechanical Enginnering, Brno, Czech Republic, 1997, pp. 148-153, ISBN 80-214-0884-7.

[16] Schwarz, J., Očenášek, J.: Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Genetic Algorithm BMDA and BOA, Proceedings of the Mendel'99 Conference, Brno University of Technology, Faculty of Mechanical Engineering, Brno, 1999, pp. 124-130, ISBN 80-214-1131-7.

[17] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++(Version1.0). Illigal Report 99011, February 1999, pp. 1-16.

[18] Pelikan, M., Goldberg, D. E., & Cantú-Paz, E.: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, November 1998, pp. 1-25.

[19] Dunlop A. E., Kernighan, A Procedure for Placement of Standard Cell VLSI circuits, IEEE Transactions on Computer-Aided Design 4(1) (1985), pp. 92-98.

[20] A Benchmark Set, University of California, Los Angeles, VLSI CAD Laboratory, http://vlsicad.cs. ucla.edu /~ cheese/benchmarks.html.