Critical Problems for a Slicing Floorplan

Armen Kostanyan
IT Educational and Research Center of Yerevan State University
Yerevan, Armenia
armko@ysu.am

Sona Kurazyan
College of Engineering of American University of Armenia
Yerevan, Armenia
kurazyan.sona@gmail.com

***Abstract*—A slicing floorplan is a geometrical structure obtained by a series of successive dissections of a given rectangle by horizontal or vertical lines. The logical structure of a slicing floorplan can be represented by a binary tree whose leaves denote rectangles of the resulting floorplan and internal nodes specify horizontal and vertical cut lines.**

The slicing structure is widely used in digital circuit physical design to determine rectangles (or, rooms), where blocks (i.e., separated parts of the digital circuit) should be allocated. The quality of a floorplan is estimated based on the area of its enclosing rectangle and the closeness of logically connected blocks to each other.

We suggest a method of transformation of a floorplan to solve several optimization problems such as concentration of predefined blocks around a target point, migration of predefined blocks towards each other, etc. All these transformations are done by keeping the bounding rectangle of the floorplan.

***Keywords*—VLSI physical design, placement, slicing floorplan, optimization.**

I. INTRODUCTION

Floorplanning is the determination of rectangular regions on the plane to optimally accommodate blocks (that is, the integral parts of a digital circuit) into them in order to meet one or another objective. This is the initial phase of the digital circuit physical design – the performance of the entire chip is extremely dependent on it [1]. Due to the huge complexity of integrated circuits, the *divide and conquer* approach is widely used in both their construction and placement.

Specifically, the recursive approach to floorplan construction leads to a recursive structure of the resulting floorplan. There are different forms of representation of the logical structure of a recursively constructed floorplan such as O-tree[2], B*-tree[3], etc. Floorplans can be classified into two categories: slicing and non-slicing. A *slicing floorplan* is an important special case of a general floorplan obtained by a series of successive dissections of a given rectangle by horizontal or vertical lines. The logical structure of a slicing floorplan can be represented by a binary tree whose leaves denote blocks and internal nodes specify horizontal and vertical cut lines [4].

Construction of a feasible floorplan is typically performed in two phases: at the beginning an initial floorplan is constructed in a greedy fashion and then it is optimized under additional conditions. During optimization phase the initial floorplan is successively transformed as follows: at each step of transformation, first, the logical structure of the current floorplan is determined; then it is transformed to another structure by applying some operations; finally, the next floorplan is reconstructed from the modified structure. This approach is used in [5] for greedy optimization of a non-slicing floorplan based on the O-tree representation. The same was done in [6] for genetic optimization of a floorplan based on the B*-tree representation. In [7] and [8] the simulated annealing and the gradient optimizations are used for a slicing floorplan based on binary tree representation.

This paper focuses on the improvement of local characteristics of a globally optimized slicing floorplan that can be considered as the second phase of optimization (such an optimization is generally done due to the deterioration of other characteristics of the original floorplan). More precisely, we consider how to concentrate a high priority set of blocks around a target point (the solution to this problem was presented in [9]), how to approach them to a desired line, or to each other anywhere within the floorplan.

The paper is organized as follows.

Section 2 presents the main concepts. Section 3 presents solutions to the net migration problem. A solution to the net contraction problem by means of reducing it to the problem of migration of a net towards a target point is suggested in sections 4. Finally, the conclusion summarizes the obtained results.

II. MAIN CONCEPTS AND DEFINITIONS

Slicing floorplan. Suppose we are given a set of block identifiers denoted as *Identifiers*. We define a *block domain* (or, simply, a *domain*) to be a triplet $D=<id, width, height>$, where $id \in Identifiers$, *width* and *length* are sizes in horizontal and vertical directions, respectively. In addition, we define an *allocated domain* to be a pair $d=<D, pos>$, where D is a domain, $pos=<x, y>$ is the Cartesian coordinates of the lower left corner of D.

The slicing floorplan *(or, simply,* floorplan*)* with *pos, width* and *height* attributes is defined to be a set F of allocated domains as follows*:*

This work was partially supported by the RA MES State Committee of Science, in the frames of the research project N 15T-18350.

978-1-5386-2830-0/17/$31.00 ©2017 IEEE

- If d is an allocated domain, then $F=\{ d \}$ is a floorplan whose *pos*, *width* and *height* attributes coincide with the corresponding attributes of d.
- If F_1 and F_2 are two floorplans without common block identifiers such that

$$F_2.pos = F_1.pos + <0, F_1.height>,$$
$$F_1.width = F_2.width,$$

then $F=F_1 \cup F_2$ is also a floorplan (denoted as $F=H(F_1, F_2)$) with the following attributes:

$$F.pos=F_1.pos,\ F.width=F_1.width(=F_2.width),$$
$$F.height=F_1.height+F_2.height.$$

- If F_1 and F_2 are two floorplans without common block identifiers such that

$$F_2.pos = F_1.pos + <F_1.width, 0>,$$
$$F_1.height = F_2.height,$$

then $F=F_1 \cup F_2$ is also a floorplan (denoted as $F=V(F_1, F2)$) with the following attributes:

$$F.pos = F_1.pos,\ F.width = F_1.width+F_2.width,\ F.height = F_1.height(=F_2.height).$$

A floorplan is said to be *trivial* if it consists of a single allocated domain; otherwise it is said to be *non-trivial*. Let us denote by $F=\Sigma(F_1, F_2)$ the non-trivial floorplan constructed from subfloorplans F_1 and F_2 with the use of the dissection operation Σ which is either H or V. If $F=\Sigma(F_1, F_2)$, then we shall use the *left*(F) and *right*(F) notations for F_1 and F_2, respectively.

Given a slicing floorplan F, we define a *slicing tree of F* (denoted as *sTree*(F)) as a labeled 2-tree as follows:

- If F is a trivial floorplan consisting of an allocated domain d, then *sTree*(F) is a single tree vertex labeled by *id*(D).
- Else, if $F=\Sigma(F_1, F_2)$, then *sTree*(F) is a 2-tree whose root is labeled by symbol Σ and whose left- and right-subtrees of the root are *sTree*(F_1) and *sTree*(F_2), respectively.

Fig. 1 presents a floorplan and its slicing tree.

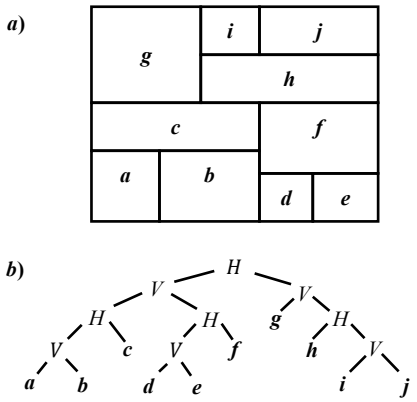

Fig. 1.Example. *a*) Slicing floorplan; *b*) Slicing tree.

Claim 1: Floorplan can uniquely be reconstructed based on the set of domains, slicing tree and position.

Vector of gravity. The notion of the vector of gravity is used to represent the disposition of domains of a specified net considering their areas.

Let F be a floorplan and N be a set of domains. Define the *weight* and the *center of gravity* of F (denoted as *weight*(F, N) and *centerOfGravity*(F, N), respectively) as follows:

- If F is a trivial floorplan consisting of a single allocated domain $d=<D, pos>$, then:
  - If D is not from N, then
    weight(F, N)=0,
    centerOfGravity(F, N) is undefined;
  - Else, if D is from N, then
    weight(F, N) is the area of D,
    centerOfGravity(F, N) is the geometrical center of d;
- Else, if $F=\Sigma(F_1, F_2)$ then:
  - If *weight*(F_1, N)= *weight*(F_2, N)=0, then
    weight(F, N)=0,
    centerOfGravity(F, N) is undefined;
  - Else, if *weight*(F_i, N)≠0, *weight*(F_{3-i}, N)=0, then
    weight(F, N)=*weight*(F_i, N),
    centerOfGravity(F, N)=*centerOfGravity*(F_i, N), (i=1, 2).
  - Else, if *weight*(F_1, N) ≠0 and *weight*(F_2, N) ≠0, then
    weight(F, N)= *weight*(F_1, N) + *weight*(F_2, N),
    centerOfGravity(F, N) is the point on the segment connecting the centers of gravity of F_1 and F_2 that divides it into inverse proportion to *weight*(F_1, N) and *weight*(F_2, N).

Given a floorplan F and a set of domains N, define the *vector of gravity* of F to be the pair <*centerOfGravity*(F, N), *weight*(F, N)>.

Swap condition. Let $F=\Sigma(F_1, F_2)$ be a non-trivial floorplan, N be a set of domains, p be a point on the plane. Define *swapCondition*(F, N, p) to be *true*, iff swapping F_1 with F_2 makes the center of gravity of the resulting floorplan closer to p.

III. NET MIGRATION

The first optimization problem that we shall consider, is the problem of moving a critically important net (i.e., a set of interconnected blocks) as near as possible to a target point by keeping the bounding rectangle of the floorplan unchanged. In fact, this problem is about the reallocation of the domains of a floorplan so that the specified domains would be located near the target point.

A. The Net Migration Problem

Given a floorplan F, a net N identified by a subset of allocated domains from F and a target point p, define the *net migration problem* to be the problem of reallocating the elements of N near p by swapping subfloorplans of F.

Note, that this problem can optimally be solved by *brute force* by considering all the possibilities of making swaps at internal nodes of *sTree*(F). But the number of possibilities to be explored in that case will exponentially depend on the number of domains, which is not acceptable for a large

number of them. Our *heuristic approach* to this problem is based on the upward and downward traversals of *sTree*(*F*) by making swaps at inner nodes when it is necessary. To decide whether a swap is needed, the notion of the *vector of gravity* is used.

B. Solution to the Net Migration Problem

We suggest an approximate solution to the net migration problem consisting of two phases. During the *first phase*, we traverse the slicing tree of floorplan in postorder and make swaps at inner nodes, if the swap condition holds. During the *second phase*, we do the same by traversing the slicing tree in preorder.

As an illustration, consider the floorplan in Fig. 2 consisting of domains <***a***, 1, 1>, <***b***, 1, 1>, <***c***, 3, 2>, <***d***, 2, 2>, <***e***, 2, 1>, <***f***, 2, 1>. Suppose we need to approach the domains ***a***, ***d*** and ***f*** to the specified target point. The first (*upward*) phase of the algorithm transforms the slicing tree of the floorplan in Fig. 2 as it is described in Fig. 3. Fig. 4 presents the floorplan reconstructed from the slicing tree transformed.

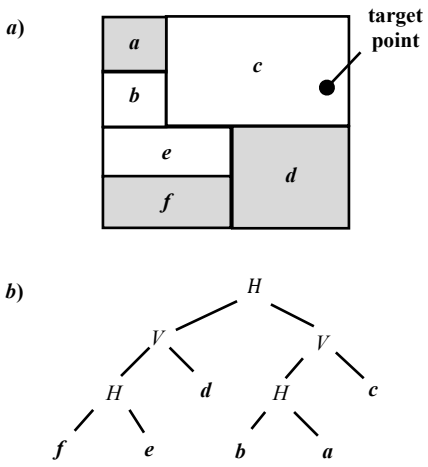

Fig. 2. Example: *a*) Slicing floorplan, *b*) Slicing tree.

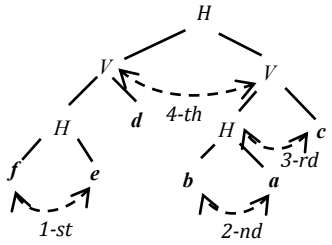

Fig. 3. Transformation of the slicing tree during upward optimization.

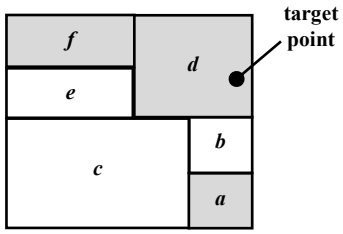

Fig. 4. The upward optimization result.

The second (*downward*) phase of the algorithm transforms the slicing tree of the upward optimized floorplan in Fig. 4 as it is described in Fig. 5. Finally, Fig. 6 presents the fully optimized floorplan reconstructed from the slicing tree transformed during downward optimization.

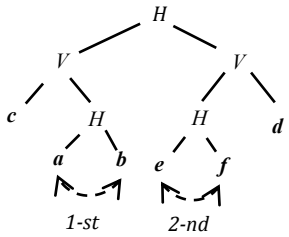

Fig. 5. Transformation of the slicing tree during downward optimization.

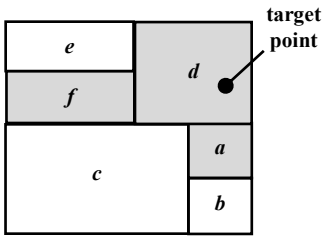

Fig. 6. The downward optimization result: *a*) Slicing floorplan, *b*) Slicing tree.

Fig. 7 demonstrates a particular case of migration of a net to a target point when the average distance between domains belonging to the net and the target point is reduced by 53%. It is important to note that subsequent upward and downward passes through the resulting floorplan will no longer optimize it. Indeed, no swaps will be done during the upward pass next to the downward one, as otherwise these swaps would have to be made during the previous pass. Hence, the subsequent downward pass will also do nothing.

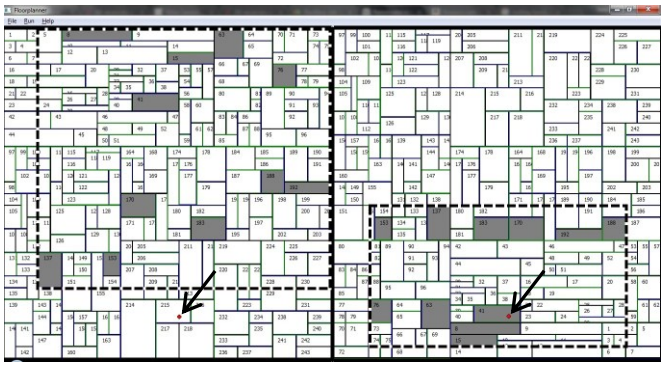

Fig. 7. Particular case of a net migration to a target point (the target point is shown by arrow). The average distance between domains belonging to the net and the target point is reduced by 53%.

C. Analysis

Let us estimate the time complexity of the net migration procedure, assuming that there are n domains in F and k domains in N. During each optimization phase, we successively visit nodes of *sTree*(F) spending O(1) time at each, except time needed for deciding whether the domain at a leaf node belongs to N. Considering that *sTree*(F) consists of $2n$-1 nodes and assuming that N is represented by means of a structure allowing to carry out O(log k)-time search, we obtain O(n·log k) time complexity for the net migration procedure.

D. Net Migration Varieties

The solution to the problem of migration of a net towards a target point can easily be adopted to solve the problem of migration of a net towards another goal. To do this, one can redefine the swap condition taking into account the closeness of a given net to the specified goal. Fig. 8 demonstrates migration of the net in Fig. 7 to a target line.

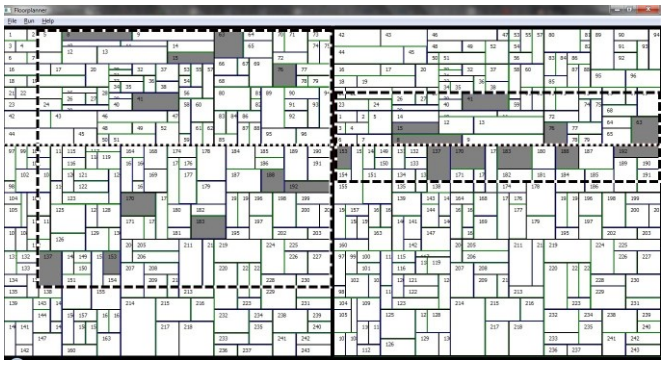

Fig. 8. Particular case of a net migration to a target line. The average distance between domains belonging to the net and the target line is reduced by 80%.

IV. Net Contraction

Another critical problem we shall consider is the problem of the contraction of a net within a given floorplan. This problem can be generalized as the problem of moving predefined domains towards each other anywhere within the floorplan. The net contraction problem often arises after the construction of a floorplan, if it turns out that there are a number of high priority (or, *critical*) nets that should be additionally optimized in spite of worsening of the lower priority nets.

A. The Net Contraction Problem

Given a floorplan F and a net $N \subseteq domains(F)$, define the *net contraction* problem to be the problem of moving elements of N to each other as close as possible by swapping subfloorplans of F and thus keeping the bounding rectangle of F.

B. Solution to the Net Contraction Problem

We suggest an approximate solution to the net contraction problem by reducing it to the problem of migration of a net towards a target point as follows.

If $F = \Sigma(F_1, F_2)$, then

- Compute $c_2 = centerOfGravity(F_2, N \cap F_2)$;
- Transform the floorplan F_1 into F_1' by solving the problem of migration the net $N \cap F_1$ towards c_2;
- Compute $c_1' = centerOfGravity(F_1', N \cap F_1)$;
- Transform the floorplan F_2 into F_2' by solving the problem of migration the net $N \cap F_2$ towards c_1';
- Return $F' = \Sigma(F_1', F_2')$.

Fig. 9 demonstrates contraction of the net in Fig. 7 resulting in reduction the average distance between domains of the specified net by 80%.

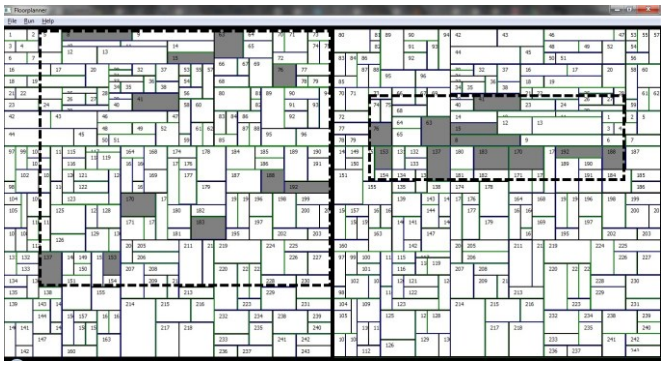

Fig. 9. Particular case of a net contraction. The average distance between domains belonging to the net is reduced by 42%.

C. Analysis

Because of solving the net migration problem a constant number of times, the net contraction algorithm runs in $O(n \cdot \log k)$ time, where $|F|=n$, $|N|=k$.

V. Conclusion

Transformations of a slicing floorplan to optimize a critical net have been considered in this paper. The considered transformations are based on the rearrangement of constituent elements of the floorplan in a way that the bounding rectangle is kept unchanged. The optimization of a critical net is performed in the following directions:

- Migration of a critical net towards a given goal (i.e., point and line),
- Convergence of elements of a critical net to each other.

It is assumed that the critical net to be optimized is a high priority net the desired disposition of which is determined by an expert. The transformations targeted to the improvement of a high priority net are generally done at the cost of deterioration of other nets.

References

[1] S. M. Sait and H. Youseff, *VLSI Physical Design Automation: Theory and Practice*, World Scientific publishing, 2004.

[2] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," in *Proc. of ACM/IEEE Design Automation Conference*, 1999, pp.268-273.

[3] Y. Chung, Y Chang, G. Wu, and S. Wu, "B*-tree : A new representation for non-slicing floorplans," in *Proc. of Design Automation Conference*, 2000, pp. 458-463.

[4] R. H. J. M. Otten, "Layout structures," in *Proc. of IEEE Large Scale Systems Symposium*, 1982, pp. 96-99.

[5] M. Tang, "A new greedy algorithm for vlsi floorplan optimization," in *Proc. of the International Conf. on Computer Science and Software Engineering*, pp. 1126-1129, June, 2008.

[6] T. Singha, H.S. Dutta, and M. De, "Optimization of floor-planning using genetic algorithm," *Published by Elsevier Ltd.*, vol. 4, pp. 825—829, 2012.

[7] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proc. of ACM/IEEE Design Automation Conference*, 1986, pp. 101-107.

[8] A. Kostanyan and Kh.Hayrapetyan, "Method of slicing floorplan optimization," in *Proc. of Computer Science & Information Technologies (CSIT) Conference*, Yerevan, Armenia, 2005, pp. 506-510.

[9] A. Kostanyan and S. Kurazyan, "Migration of a net in a slicing floorplan," in *Proc. of Computer Science & Information Technologies (CSIT) Conference*, Yerevan, Armenia, 2017, pp. 156-159.