Migration of a Net in a Slicing Floorplan

Armen Kostanyan
Yerevan State University
Yerevan, Armenia
e-mail: armko@ysu.am

Sona Kurazyan
American University of Armenia
Yerevan, Armenia
e-mail: selart66@gmail.com

ABSTRACT
Creation of a feasible floorplan is an important part of the digital circuit physical design. The floorplan is defined as a decomposition of a given rectangle into rectangular domains determining the regions where blocks (i.e., separated parts of the digital circuit) should be allocated. The quality of floorplan is estimated based on the area of the enclosing rectangle and the closeness of logically connected blocks to each other.

We suggest an algorithm for migration of a net (that is, an interconnected set of blocks) towards a target point by keeping the floorplan enveloping rectangle. It is assumed that the net involved in the migration is a high priority net the desired disposition of which is determined a priori. The suggested improvement is generally done by means of degrading of less priority nets.

Keywords
Floorplanning, slicing tree, optimization

1. INTRODUCTION
The digital circuit physical design deals with realization of a circuit in physical space to minimize both the occupied area and the total wirelength. The physical design is divided into placement, routing and compaction tasks [1].

Floorplaning is a generalization of the placement task aiming to determine rectangular areas to optimally allocate blocks (that is, the separated parts of a digital circuit) into them. Construction of a feasible floorplan is typically performed in two phases: at the beginning an initial floorplan is constructed in greedy fashion and then it is optimized considering additional circumstances.

The floorplan optimization is based on some representation of its logical structure such as O-tree[2], B*-tree[3], etc. The *slicing floorplan* is an important specific case of a general type 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 the leaves of which denote blocks, and internal nodes specify horizontal and vertical cut lines [4].

During optimization phase the initial floorplan is successively transformed so that at each step of transformation the logical structure of current floorplan is determined, the latter is transformed to another structure by applying some operation, and 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 the 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. More precisely, we consider how to concentrate a high priority set of blocks around a target point. Such optimization generally worsens the cumulative assessment of other nets of the original floorplan.

2. 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* □ *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 angle of D.

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

- 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 $F=F_1 \square F_2$ 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 $F=F_1 \square F_2$ 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 $F=\square(F_1, F_2)$ the non-trivial floorplan constructed from subfloorplans F_1 and F_2 with the use of the dissection operation $\square$ which is either H or V.

Given a slicing floorplan F, we define a *slicing tree of F* (denoted as *sTree*(F)) as a labeled 2-tree as it 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 $F=\square(F_1, F_2)$, then $sTree(F)$ is a 2-tree whose root is labeled by symbol $\square$ and whose left- and right- subtrees of the root are $sTree(F_1)$ and $sTree(F_2)$, respectively.

Fig. 1 below presents a floorplan and its slicing tree.

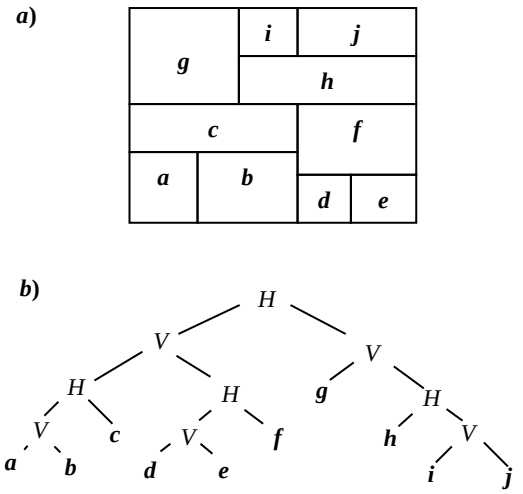

Fig. 1. Example. a) Slicing floorplan; b) Slicing tree.

The following claim follows from the definitions above.

Claim. 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 $F=\square(F_1, F_2)$ then:
  - If $weight(F_1, N)= weight(F_2, N)=0$, then
    $weight(F, N)=0$,
    $centerOfGravity(F, N)$ is undefined;
  - Else, if $weight(F_i, N)\neq 0$, $weight(F_{3-i}, N)=0$, then
    $weight(F, N)=weight(F_i, N)$,
    $centerOfGravity(F, N)=centerOfGravity(F_i, N)$,
    $(i=1, 2)$.
  - Else, if $weight(F_1, N)\neq 0$ and $weight(F_2, N)\neq 0$, 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 $F=\square(F_1, F_2)$ 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.

Net migration problem. Given a floorplan F, a set $N\subseteq domains(F)$ of domains and a target point p, we define the *net migration problem* to be the problem of moving the elements of N towards p as close as possible by keeping the enveloping rectangle of F. We shall consider a solution to this problem based on making optimizing swaps at inner nodes of $sTree(F)$.

3. MOVING A NET TO A TARGET POINT

Our approach to solve the net migration problem consists 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 make 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 concentrate the domains ***a***, ***d*** and ***f*** around the specified target point:

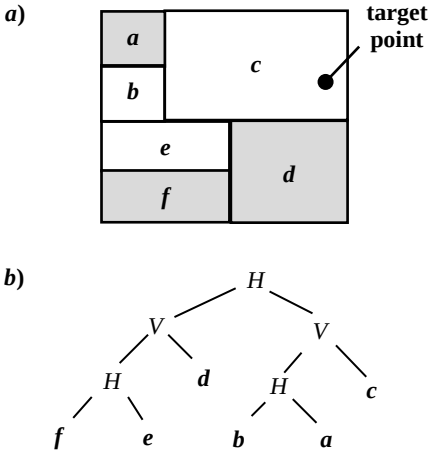

Fig. 2. Example: *a*) Slicing floorplan, *b*) Slicing tree.

The first (*upward*) phase of the algorithm transforms the slicing tree of the floorplan in **Fig. 2** as follows:

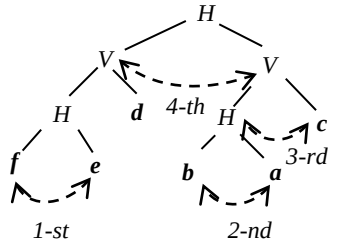

Fig. 3. Transformation of the slicing tree during upward optimization.

The floorplan transformation described in **Fig. 3** results in upward optimized floorplan below:

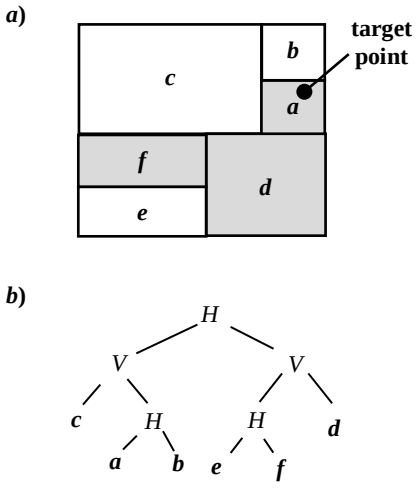

Fig. 4. The upward optimization result: *a*) Slicing floorplan; *b*) Slicing tree.

The second (*downward*) phase of the algorithm transforms the slicing tree of the upward optimized floorplan in **Fig. 4** as follows:

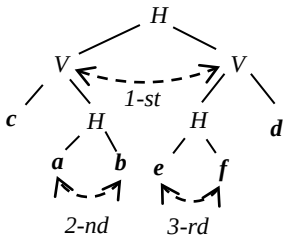

Fig. 5. Transformation of the slicing tree during upward optimization.

Finally, the floorplan transformation described in **Fig. 5** results in downward optimized floorplan in **Fig. 6**:

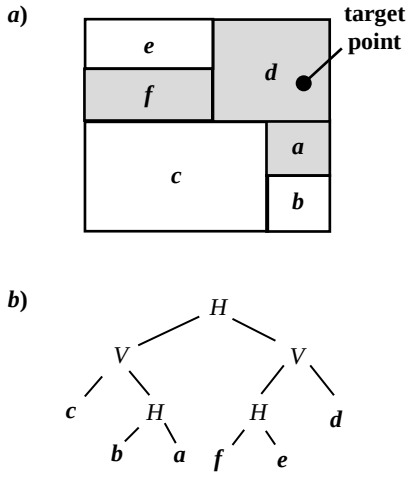

Fig. 6. The downward optimization result: *a*) Slicing floorplan, *b*) Slicing tree.

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 were to be made at the previous pass. It follows from this that the subsequent downward pass will also do nothing.

Fig. 7 presents a "good" and a "bad" cases of running the algorithm:

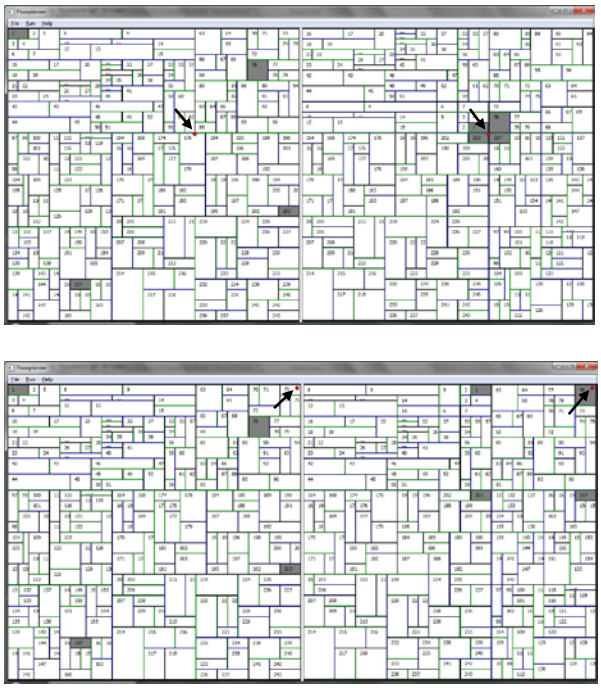

Fig. 7. Different cases of running the algorithm.

4. ANALYSIS

Let us estimate the time complexity of 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 $\Theta(1)$ 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 $\Theta(\log k)$-time search, we obtain $\Theta(n \log k)$ time complexity for the whole procedure.

5. CONCLUSION

In this paper, an approximate algorithm for migration of a net towards a target point inside the slicing floorplan is suggested. It is assumed that the net involved in the migration is a high priority net the desired disposition of which is determined by an expert after automatic construction of an entire floorplan. The suggested improvement of a high priority net is generally done due to degrading of less priority nets.

REFERENCES

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

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

[3] Y. Chung, Y Chang, G. Wu, S. Wu, “B*-tree : A new representation for non-slicing floorplans”, *Proc. of Design Automation Conference*, pp. 458-463, 2000.
[4] R. H. J. M. Otten, “Layout structures”, *Proc. of IEEE Large Scale Systems Symposium*, pp. 96-99, 1982.
[5] M. Tang, “A New Greedy Algorithm for VLSI Floorplan Optimization”, *Proc. of the International Conf. on Computer Science and Software Engineering*, pp. 1126-1129, 2008.
[6] T. Singha, H.S. Dutta, M. De, “Optimization of Floor-planning using Genetic Algorithm”, *Published by Elsevier Ltd.*, 2012.
[7] D. F. Wong, C. L. Liu, “A new algorithm for floorplan design”, *Proc. of ACM/IEEE Design Automation Conference*, pp. 101-107, June 1986.
[8] A. Kostanyan, Kh.Hayrapetyan, “Method of Slicing Floorplan Optimization “, *Proc. of Computer Science & Information Technologies (CSIT) Conference, Yerevan, Armenia*, pp. 506-510, 2005.