Modeling Non-Slicing Floorplans with Binary Trees

Florin Balasa

University of Illinois at Chicago, Dept. of EECS, Chicago, IL 60607

Abstract

Several novel topological representations of non-slicing floorplans [2] have been more recently proposed, providing new ideas and techniques for solving block placement problems and other related layout applications. Among these topological representations, ordered trees exhibit a lower redundancy and, therefore, a provable smaller search space, which makes them the best topological candidate for solving general block placement problems. Starting from the early eighties, binary trees have been widely used to represent *slicing* floorplans [7]. This paper shows that binary trees can efficiently model *non-slicing* floorplans as well, as there is a one-to-one mapping between the sets of binary and ordered trees representing the floorplan. Moreover, this paper shows that binary trees exhibiting a certain property can be used to represent block placement configurations with symmetry constraints, which is very useful when dealing with device-level placement problems for analog layout. As the number of these trees is proven to be smaller than the number of symmetric-feasible sequence-pairs [1], using binary trees is better than using either sequence-pairs or O-trees when solving analog placement problems. A comparative evaluation, substantiating these theoretical results, has been carried out by providing alternative optimization engines to a placement tool operating in an industrial environment.

1 Introduction

The placement problem for integrated circuit layout consists, basically, in finding the location of a given set of modules (blocks, cells) on the chip plane, aiming to optimize a certain cost function under certain feasibility constraints. Minimizing the area of the bounding box of all the modules is usually the basic optimization target. The mutual nonoverlap of the modules is usually a minimal feasibility constraint. The block placement under the nonoverlapping constraint is often called *packing*. Even when blocks have a fixed rectangular shape, the packing problem is complex: the decision whether a given set of fixed-oriented rectangles, having widths and heights real numbers, could be packed onto a chip of known width and height was proven to be NP-complete, while the problem of finding a minimum area packing is NP-hard [5].

Rectangle packing problems have been analytically modeled as nonlinear optimizations, where overlapping effect has been described by means of a penalty or barrier function [10]. Onodera *et al.* used a direct branch-and-bound approach, constructing a solution space by assigning one out of the four positioning relations "left of", "right of", "above", "below", to every pair of modules [6]. The branch-and-bound algorithm uses a lower bound of the chosen cost function in order to prune the search; it eventually finds the optimal packing as the solution space is explored exhaustively. However, it is acknowledged that the technique is effective only for problems of small size.

The simulated annealing and genetic algorithms have proven to be a very effective choice for solving block placement problems. These algorithms use stochastically controlled hill-climbing to avoid local minima during the optimization process. In addition, they do not impose severe constraints on the size of the problems or on the mathematical properties of the cost function.

A simulated annealing algorithm can equally operate with two classes of representations of module configurations. The most straightforward and widely used is the *absolute* (or *flat*) representation, where the positions of the modules are directly specified in terms of coordinates relative to an arbitrary system of axes in the chip plane. However, the convergence of the exploration may be slow due to the huge size of the search space (which contains also infeasible placement configurations).

An alternative approach to the absolute representation is to define a set of codes – each code representing a placement configuration – as a solution space. A combinatorial search could find a "best" code (relative to a chosen cost function) in the solution space. Since the packing problem is NP-hard, the size of any such a solution space is expected to be exponential. As an exhaustive exploration of the whole space would be computationally expensive or even infeasible, simulated annealing and genetic algorithms have proven their effectiveness – finding good solutions in moderate time.

An encoding system of feasible placement configurations is usually referred to as a *topological* representation, due to the fact that such an encoding is an encryption of the positioning (topological) relations between any pair of modules. Murata *et al.* formulated the minimum requirements for the solution space of codes such that the exploration be effective [5]: 1) the solution space must be finite; 2) every solution must be feasible in order to ensure the convergence of the exploration towards a feasible solution; 3) the construction of the placement from any code must be performed in polynomial time; 4) there must be a code corresponding to one of the optimal placement configurations.

The first topological representations were derived from the *slicing* floorplan model proposed initially by Otten [7]. The modules are organized in a set of slices which recursively bisect the layout horizontally and vertically. The direction and nesting of the slices is recorded in a (binary) slicing tree or, equivalently, in a normalized Polish expression [9] (see Fig. 1). Any encoding based on the slicing model does not satisfy the 4th requirement of the solution space if area is the criterion of optimality: usually, minimum area solutions do not fit in a slicing floorplan structure. Slicing representations limit the set of reachable layout topologies. This can degrade layout density, especially when cells may be very different in size.

More recently, several novel topological representations, not restricted to slicing floorplan topologies, have been proposed. Murata *et al.* suggested to encode the "left-right" and "above-below" positioning relations between modules using two sequences of module permutations, named *sequence-pair* [5]. Guo and Cheng proposed the *ordered tree (O-tree)* data structure which needs a smaller amount of encoding storage and linear time computation effort to generate each
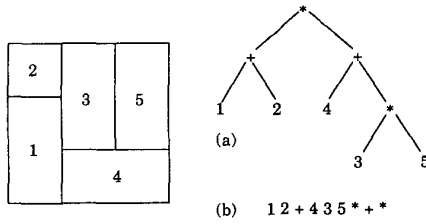
13

Figure 1: Slicing floorplan: (a) slicing tree, and (b) normalized Polish expression

placement configuration. In addition, the number of possible encodings suggests a reduced redundancy and, therefore, a smaller solution space.

This paper shows that binary trees can be efficiently used to perform block placement when the floorplan topology is non-slicing. This results from the fact that there is a one-to-one mapping between the sets of binary trees and O-trees representing the floorplan. Moreover, this paper shows that binary trees exhibiting a certain property can be employed more efficiently than the O-trees to solve block placement problems with symmetry constraints, which is extremely useful in analog placement. In high-performance analog circuits it is often required that groups of devices are placed symmetrically with respect to one or several axes. The main reason of symmetric placement and routing is to match the induced parasitics in the two halves of a group of devices.

Symmetry constraints are easy to model when absolute placement configurations are explored. However, in topological representations, handling symmetry is significantly more difficult. This problem was successfully addressed in the context of sequence-pairs [1], where only the *symmetric-feasible* sequence-pairs are explored. The problem was also addressed in the context of the O-trees [8]: although the techniques to detect whether an O-tree representation is feasible or not are very effective, the exploration space is the entire set of O-trees, most of them being unfeasible in symmetry point of view. Introducing the new concept of *symmetric-feasible* binary trees, this paper shows how to restrict the exploration to only feasible representations. As the number of symmetric-feasible binary trees is proven to be smaller than both the numbers of O-trees and of symmetric-feasible sequence-pairs, using binary trees proves to be theoretically advantageous. The implementation of a placement tool having alternative optimization engines employing different topological representations confirms in practice the theoretical results.

The paper is organized as follows. Section 2 introduces the binary tree representation and proves its equivalence to the O-tree representation. Section 3 briefly reviews the symmetric-feasible sequence-pairs. Then, Section 4 introduces the symmetric-feasible binary trees and explains how to use them to solve block placement problems with symmetry constraints. Finally, Section 5 gives an overview of the experimental results, and Section 6 presents the basic conclusions of this research.

2   The binary tree representation

According to [3], a placement configuration of n rectangular blocks can be represented by an O-tree, that is a tree with $n + 1$ nodes, encoded by $(\mathcal{T}, \pi)$, where $\mathcal{T}$ is a $2n$-bit string identifying the branching structure of the tree relative to a traversal order (a '0' corresponds to descending an edge, while an '1' – to subsequently ascending that edge), and π is a permutation of the block names. Fig. 2(a) shows an O-tree having the encoding $(\mathcal{T}, \pi)=(00110100011011 , adbcegf)$.
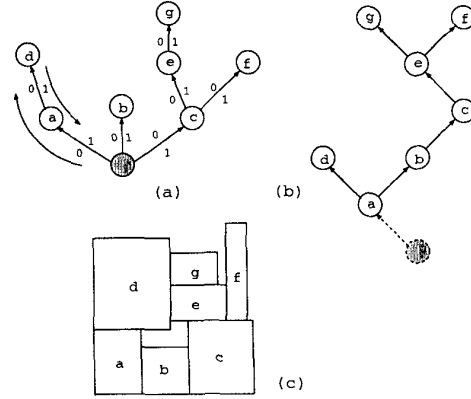

Figure 2: (a) O-tree representation, (b) binary tree representation, and (c) the block placement

Let $T = (T_1, \ldots, T_k)$ be an O-tree, where $T_1, \ldots, T_k$ are its subtrees relative to the root. The O-tree can be transformed recursively into a binary tree $B(T)$ as follows:

a) if $k = 0$, $B(T)$ is empty;

b) if $k > 0$, the root of $B(T)$ is the root of T_1; the left subtree of $B(T)$ is $B(T_{11}, T_{12}, \ldots)$, where T_{1j} are the subtrees of T_1; the right subtree of $B(T)$ is $B(T_2, \ldots, T_k)$.

The binary tree derived from the O-tree in Fig. 2(a) is shown in Fig. 2(b) (the root of the O-tree becomes obsolete). As the inverse transformation is straightforward, it follows that there is a one-to-one correspondence between the sets of O-trees and of binary trees having one node less.

A binary tree which nodes represent rectangular blocks imposes the following vertical and horizontal positioning constraints:

(a) each block in the left binary subtree is above its root block;

(b) if two blocks are overlapping along their y-coordinate projection, the block visited first in a preorder traversal of the binary tree[1] is to the left of the block visited the second.

Let $B_1, \ldots, B_n$ be rectangular blocks to be placed on a chip area, each block b_i having width w_i and height h_i, and having (x_i, y_i) as coordinates of its left-bottom corner. To construct a block placement from a binary tree representation, the positioning constraints imply the following coordinate updates:

(a) if B_j is in the left subtree of B_i, then $y_j = y_i + w_i$;

(b) for each block B_i, let $\Psi(i)$ be the set of blocks B_k visited before B_i in a preorder traversal of the binary tree, which y-spanning intervals $(y_k, y_k + h_k)$ overlap $(y_i, y_i + h_i)$. If $\Psi(i)$ is non-empty, then

$x_i = \max_{k \in \Psi(i)} \{x_k + w_k\}$.

Due to the one-to-one correspondence between O-trees and binary trees, the block placement of a binary tree representation can be performed using the linear-time algorithm described in [3], which provably yields a minimum area packing. Consequently, the binary tree representation satisfies Murata's 2nd and 3rd requirements (see Section 1). Fig. 2(c) shows the result of the block placement algorithm.

A binary tree may be encoded by $(\mathcal{T}, \pi)$, similarly to an O-tree, but the bit string $\mathcal{T}$ needs 2 bits less (as there is one edge less in the binary tree). However, there are more important advantages in using in using the binary tree representation[2]. One advantage refers to the computation

---
[1]Visit the root; traverse the left subtree; traverse the right subtree.
[2]It should be noticed that, due to the one-to-one mapping, any property of the binary tree representation may be translated to the O-trees. However, the referred advantages are naturally embedded in the binary tree framework.

14

of the number of representations, i.e. the size of the solution space in a block placement problem.

Lemma The number of binary trees with n nodes is [4]

$$b_n = \frac{1}{n+1} \begin{pmatrix} 2n \\ n \end{pmatrix}, \text{ where } \begin{pmatrix} \alpha \\ n \end{pmatrix} = \frac{\alpha(\alpha-1)\cdots(\alpha-n+1)}{n!}.$$

As the rectangular blocks $B_1, \ldots, B_n$ can be attached in $n!$ different ways to the n nodes of the binary tree, it follows that the size of the solution space is $b_n n!$, thus finite. Therefore, Murata's first requirement is satisfied.

Although the existence of a binary tree representation corresponding to a minimum area placement configuration can be directly proven (Murata's 4th requirement), we may use for brevity the similar result obtained for O-trees in [3] and the one-to-one mapping between the two representations. As all four requirements are satisfied, the binary tree representations can be effectively explored employing, e.g., simulated annealing as an optimization engine.

A significant advantage offered by the binary trees consists in the efficient handling of symmetry constraints, which is very useful in analog placement, as explained in Section 1. For the sake of consistency, some previous results need to be mentioned first.

3 Symmetric-feasible sequence-pairs

The basic idea of the sequence-pair representation is to encode any rectangle packing as a pair of block sequences (α, β). Denoting by α_i the block having position i in the sequence α, and by α_A^{-1} the position of block A in the sequence (as α is a one-to-one mapping, α^{-1} is well-defined), the topological relations "left-right" and "above-below" between blocks are defined by the following conditions:

if $\alpha_A^{-1} < \alpha_B^{-1}$ and $\beta_A^{-1} < \beta_B^{-1}$ then A is to the left of B;
if $\alpha_A^{-1} < \alpha_B^{-1}$ and $\beta_B^{-1} < \beta_A^{-1}$ then A is above B.

In order to apply this topological representation to analog placement, it has been shown that symmetry constraints can be efficiently handled using the concept of *symmetric-feasible* sequence-pair [1]. Let (α, β) be the sequence-pair of a placement configuration containing a symmetry group G composed of several pairs of symmetric and self-symmetric blocks [2] relative to a common vertical axis. Denoting by $sym(x)$ the block symmetric to x, the sequence-pair (α, β) is called *symmetric-feasible* if for any distinct blocks x, y in G:

$$(S) \quad \alpha_x^{-1} < \alpha_y^{-1} \iff \beta_{sym(y)}^{-1} < \beta_{sym(x)}^{-1}$$

Example The sequence-pair encoding of the placement configuration in Fig. 3(a) is $(\alpha, \beta) = (ebfcdag, ebacdfg)$. (α, β) is symmetric-feasible relative to the symmetry group $G = ((c,d),(b,g))$ with $p = 2$ pairs of symmetric blocks. Indeed, condition (S) is true for any distinct blocks x, y in G: for instance, chosing $x = c$ and $y = g$, we have $\alpha_c^{-1} < \alpha_g^{-1}$ and $\beta_b^{-1} < \beta_d^{-1}$ (as $4 < 7$ and $2 < 5$).

Exploring the solution space of symmetric-feasible (S-F) sequence-pairs is extremely efficient as the size of this space is usually significantly smaller than the total number of sequence-pair encodings, which is $(n!)^2$ for n blocks. For instance, the number of sequence-pairs when $n = 7$, as in the example above, is $(7!)^2 = 25,401,600$; however, the number of *symmetric-feasible* sequence-pairs relative to the symmetry group G is $(n!)^2/(2p)!$, according to [1], that is only $(7!)^2/4! = 1,058,400$.

According to the *Lemma* in Section 2, the number of binary trees encoding the admissible placement configurations for the example in Fig. 3(a) is $b_7 \cdot 7! = 2,162,160$, significantly smaller than the total number of sequence-pairs, but
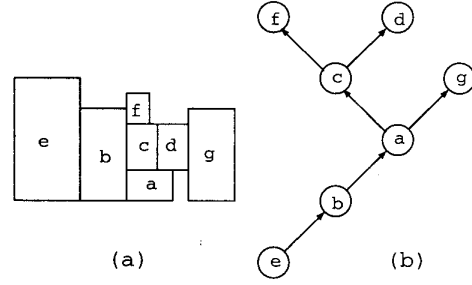

Figure 3: (a) Block placement with a symmetry group, and (b) a symmetric-feasible binary tree

still larger than the number of *symmetric-feasible* sequence-pairs. If, for instance, $n = 2p$ (all the blocks belong to a symmetry group), then the number of symmetric-feasible sequence-pairs is $(2p)!$ [1]. This number is smaller than $b_{2p} \cdot (2p)!$ – the number of binary tree (and, also, O-tree) representations, as the number of unlabeled binary trees with $2p$ nodes ($p \geq 1$) is greater than 1. However, when the asymmetric part of the circuit is larger, the number of binary tree representations becomes smaller than the number of symmetric-feasible sequence-pairs (e.g., $n \geq 8$, $p = 2$).

This explains why the placement algorithm in [8], exploring the space of O-tree representations and checking whether they are $x-$ and $y-$ symmetric-feasible, behaves sometimes worse in terms of CPU time than the algorithm in [1] which restrict the exploration to the S-F sequence-pairs. The solution space of binary trees can be reduced though, by using the concept of *symmetric-feasible* binary tree.

4 Symmetric-feasible binary trees

A binary tree can be represented recursively as $((leftSubtree)root \ rightSubtree)$ based on its in-order traversal, or $(root(leftSubtree) \ rightSubtree)$ based on its preorder traversal. The nested parentheses are used to express the tree structure.
Definition Let (γ, δ) be the pair of sequences derived from the inorder and preorder traversals of a binary tree. Using the same notations as for sequence-pairs, the binary tree encoding is called *symmetric-feasible* (S-F) relative to a symmetry group G if for any distinct blocks x, y in G:

$$(S') \quad \gamma_x^{-1} < \gamma_y^{-1} \iff \delta_{sym(y)}^{-1} < \delta_{sym(x)}^{-1}$$

Example The inorder representation of the binary tree in Fig. 3(b) is $(e \ b \ ((f)c \ d)a \ g)$; the preorder representation is $(e \ b \ a(c(f) \ d) \ g)$. As $(\gamma, \delta) = (ebfcdag, ebacfdg)$, the binary tree is symmetric-feasible relative to the symmetry group $G = ((c,d),(b,g))$. Indeed, condition (S') is true for any distinct blocks x, y in G: for instance, $\gamma_c^{-1} < \gamma_g^{-1}$ and $\delta_b^{-1} < \delta_d^{-1}$ (as $4 < 7$ and $2 < 6$).

Remark Usually (γ, δ) is not equal to the sequence-pair representation (α, β). There are sequences α and β which do not correspond to the inorder and preorder traversals of any binary tree: such a sequence-pair is, for instance, (CAB, ABC).

Theorem 1 Any compact placement configuration containing a symmetry group can be encoded with a symmetric-feasible binary tree.
This theoretical result shows that, in particular, there is at least an S-F binary tree encoding a minimum area placement; Murata's 4th requirement is consequently valid. The 2nd and 3rd conditions are equally satisfied as a result of:
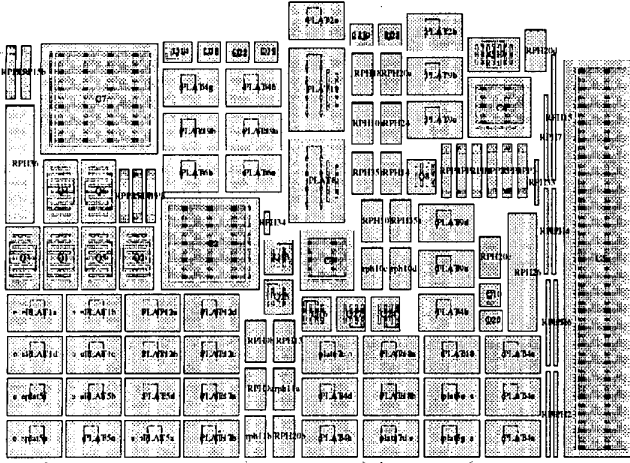
15

Figure 4: Placement of *lnamixbias* (S-F bin. trees)

Design	Nr.	Time [min]		
(# sym.groups)	cells	Sq.pair	O-tree	Bin.tree
bias.crt.gen.	85	32	25	25
biasynth(3)	65	21	17	14
modbias(5)	87	41	43	33
lnamixbias(6)	110	76	90	62
freq.divider(5)	116	68	84	57

Table 1: Placement results

Theorem 2 Given a symmetric-feasible binary tree encoding, one can build in polynomial time a minimum area placement satisfying the positioning and symmetry constraints.

Proof: A key remark is is that (γ, δ) imposes the same "above-below" topological relations as the sequence-pairs. Indeed, if $\gamma_A^{-1} < \gamma_B^{-1}$ and $\delta_B^{-1} < \delta_A^{-1}$, it follows that node A is in the left subtree of node B; according to the positioning constraints (b) of the binary tree encoding, it results that A is above B. Therefore, the y coordinates of the blocks can be computed as in [1], replacing (α, β) by (γ, δ).

However, the "left-right" relations are not the same as for the sequence-pairs. Denoting Y_k the intervals $(y_k, y_k + h_k)$, the x coordinates of the blocks are computed as follows:

initialize $x_i = 0$ and $x_{symAxis} = 0$;
for each block B_i // preorder traversal
     for each visited block B_k;
         if Y_k overlaps Y_i then $x_i = \max\{x_i, x_k + w_k\}$;
     if B_i has a symmetric not visited yet
     then $x_{symAxis} = \max\{x_{symAxis}, x_i + w_i\}$;
     if B_i has a symmetric $sym(B_i)$ already visited
     then $x_i = \max\{x_i, 2x_{symAxis} - (x_{sym(B_i)} + w_i)\}$;
for each block B_i // inverse preorder traversal
     for each visited block B_k;
         if Y_k overlaps Y_i then $x_i = \min\{x_i, x_k - w_i\}$;
     if B_i has a symmetric $sym(B_i)$ already visited
     then $x_i = \min\{x_i, 2x_{symAxis} - (x_{sym(B_i)} + w_i)\}$;

This algorithm has a quadratic complexity. In can be proven that, due to condition (S'), the resulting placement has minimum area, satisfying both symmetry and positioning constraints.□

There is a one-to-one mapping between the set of unlabeled binary trees with n nodes and the set of unlabeled S-F binary trees with $2n$ nodes. Due to this remark and the *Lemma*, if all the blocks in the layout are organized in a symmetry group ($n = 2p$), the number of (labeled) symmetric-feasible binary trees is $b_p \cdot p! \cdot 2^p$. As $2^p \le 2 \cdot 3 \cdots (p+1) = (p+1)!$, it follows that $b_p \cdot p! \cdot 2^p \le (2p)!$ which is the number of symmetric-feasible sequence-pairs [1]. The annealing algorithm can be adapted to explore only the subspace of S-F binary trees rather than the whole space of binary trees.

5 Experimental results

The placement algorithm based on binary tree representations has been embedded in a placement tool having alternative optimization engines which employ different non-slicing topological representations – sequence-pairs, O-trees. In order to ensure a correct comparative evaluation, the simulated annealing schedule is identical for all the placement algorithms. The benchmark tests are component blocks of a digital spread spectrum transceiver used in cordless telephone applications and wireless modems. The example in Fig. 4 contains 110 cells, 52 of them being organized in 6 symmetry groups.

Table 1 shows that the CPU times (obtained on an HP 9000/777 workstation) when running the algorithm based on (S-F) binary trees are better than the ones employing the other topological representations. (However, in the absence of symmetry constraints the times for binary- and O-trees are similar.) In addition, the algorithm using sequence-pairs is more effective that the one based on O-trees in the examples exhibiting more symmetry.

6 Conclusions

This paper has addressed the problem of using binary trees as a topological representation for placement configurations in non-slicing floorplans. The paper has proven that binary trees are as good as ordered trees when solving general block placement problems. In addition, the binary trees exhibiting the property of symmetric-feasibility can be efficiently used to solve placement problems with symmetry constraints – very common in analog layout design, performing better than both ordered trees and the sequence-pair representation due to a provably smaller search space.

References

[1] F. Balasa, K. Lampaert, "Symmetry within the sequence-pair representation in the context of placement for analog design," *IEEE Trans. on CAD of IC's and Systems*, Vol. 17, No. 7, pp. 721-731, July 2000.
[2] J. Cohn, D. Garrod, R. Rutenbar, L. Carley, *Analog Device-Level Automation*, Kluwer Acad., 1994.
[3] P.-N. Guo, C.-K. Cheng, T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," *Proc. 36th ACM/IEEE Des. Aut. Conf.*, pp. 268-273, June 1999.
[4] D.E. Knuth, *The Art of Computer Programming* (3rd edition), Addison Wesley Longman, 1997.
[5] H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. on Comp.-Aided Design of IC's and Systems*, Vol. 15, No. 12, pp. 1518-1524, Dec. 1996.
[6] H. Onodera, Y. Taniguchi, K. Tamaru, "Branch-and-bound placement for building block layout," *Proc. 28th ACM/IEEE Design Automation Conf.*, pp. 433-439, 1991.
[7] R. Otten, "Complexity and diversity in IC layout design," *Proc. IEEE Intn'l Symp. Circuits and Computers*, 1980.
[8] Y.-X. Pang, F. Balasa, K. Lampaert, C.-K. Cheng, "Block placement with symmetry constraints based on the O-tree non-slicing representation," *Proc. 37th ACM/IEEE Design Automation Conf.*, June 2000.
[9] D.F. Wong, C.L. Liu, "A new algorithm for floorplan design," *Proc. 23rd ACM/IEEE Des. Aut. Conf.*, pp. 101-107, 1986.
[10] C.-S. Ying, S.-L. Wong, "An analytical approach to floorplanning for hierarchical building blocks layout," *IEEE Trans. on CAD of IC's and Syst.*, Vol. 8, pp. 403-412, April 1989.

16