An Analytical Approach to Floorplan Design and Optimization

Suphachai Sutanthavibul, Eugene Shragowitz, *Member, IEEE,* and J. B. Rosen

Abstract—**An analytical method for general floorplan design and optimization is proposed. This method is based on a mixed integer programming model and application of a standard mathematical software. The method allows arbitrary combinations of rigid and flexible modules. Various objective functions such as chip area, interconnection length, timing delays or any combination of them are permitted. Routing space is estimated by the global router. Experimental data are provided.**

I. Introduction

FLOORPLANNING is one of the first steps of VLSI design. It is assumed, that at this step, rough estimations of the module areas and interconnections between them are known, while topology of the chip and exact dimensions of the modules should be defined. Common goals in the floorplan design are minimization of the chip area, minimization of interconnections length, or a combination of them. Another common problem definition is to minimize the floorplan area for the given topology. A method described in this paper covers all formulations of the floorplanning problems mentioned above without restricting a class of solutions to the slicing structures. In addition, it allows consideration of realistic situations when some modules have rigid shapes and remaining modules are of flexible shapes, or all modules have rigid shapes, i.e., it can be applied to solving placement problems.

Like other methods for this kind of problem, the proposed method provides a suboptimal solution. This suboptimal solution is obtained by the consecutive solution of subproblems of smaller size. Unlike many other methods, our method obtains the optimal solution for each subproblem. Each subproblem is solved as a 0-1 mixed integer programming problem. The final solution is achieved by successive augmentation of new elements to the already constructed partial solution. In the process, the partial solution is reformulated to reduce the number of variables in the mixed integer programming problem. That makes it possible to keep the number of variables close to a constant in each step of the process. The timing complexity of this method is linear with respect to the number of partitions. This allows solution of the floorplanning problems of any practical dimensions. Benchmarks constructed for the Workshop on Physical Design, organized by MCNC in May 1988, are used to illustrate the performance of this methodology.

The text of this paper is structured as follows. Section II reviews the related literature on floorplanning. Section III describes the mixed integer programming formulation of floorplanning. The solution method is described in Section IV. Experimental results are given in Section V.

Manuscript received December 19, 1989; revised June 26, 1990. This paper was recommended by Associate Editor R. H. J. M. Otten.

The authors are with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

IEEE Log Number 9042653.

II. Previous Work

Starting from the early 1970's a substantial number of papers on floorplanning have been published. Several fundamental papers on floorplan design belong to Otten [1], [2]. Otten used the connectivity information to define "distances" between modules and then used a two-step procedure to obtain a floorplan satisfying these distances. He used the linear algebra technique known as the Schoenberg method to produce a two-dimensional configuration, which retains given intermodule distances. At this point he applied a hierarchical rectangular decomposition (named "slicing") to obtain the final floorplan.

Starting from Otten, almost all authors use slicing to produce floorplans. Our methodology does not require the slicing technique. Among the most recent work based on slicing is the paper by Wong and Liu [3], [4]. They presented a floorplan design algorithm using simulated annealing. The floorplan is described by the *normalized Polish expressions* for slicing structures. The algorithm considers simultaneous minimization of the bounding rectangle area and the total wirelength. Experimental results showed near optimal solutions with respect to the area of the bounding rectangle for the cases where all modules were flexible. However, when all modules were rigid, only 79% area utilization was achieved.

Mueller *et al.* [5] described a bottom-up iterative algorithm for the floorplan design. The floorplan representation used in this algorithm is also based on the *normalized Polish expressions* and slicing structures. The greedy algorithm attempts to restructure the subtrees of the slicing floorplan tree from the leaves to the root in order to improve the solution. The experimental results showed significant improvement in execution time over simulated annealing [3], but the size of the chips are 7–31% larger than those obtained by the simulated annealing algorithm [3].

Another direction in floorplanning was introduced by Heller *et al.* [6]. It is rectangular dualization and rectan-

gular dissection. This technique was developed by Kozminski, Kinnen, and others [7], [9]. The problem of rectangular dualization is to find a dissection of a rectangle into rectangles that satisfies the adjacency relation among component rectangles and has the smallest area. The main results in this theory are achieved for planar graphs, describing connectivity of modules. However, real connectivity graphs are not planar, so various techniques are proposed for planarization. These techniques are based on different assumptions about the adjacency of connected modules. In the most recent work based on this approach, Lokanathan and Kinnen [10] started from clustering modules into a set of superblocks. Then a planar graph was produced and used to represent the connectivity between these superblocks. A rectangular dual of this planar graph represents the topology of the general floorplan. Mutual relations of modules are formulated as a system of linear inequalities. This system is solved by a linear programming method. After the floorplan for the superblocks is found, the floorplan of each superblock is obtained by application of a constructive two-dimensional bin-packing heuristic algorithm.

The most elaborate comparison of experimental results appeared in [11] for the benchmark test case known as AMI33 or PrimBBL2. The same benchmark was also used in many other works on floorplanning and placement [12]–[14]. A discussion on this subject will be presented in the experimental part of this paper.

A mixed integer programming model of placement different from ours was proposed in [15]. Several transformations of the original placement problem were made there. For example, any rectangular block with the aspect ratio that is less than or equal to a user-specified number is replaced by a group of square blocks with an edge size equal to the minimal edge of the original block. Then these squares are made contiguous. Conditions for nonoverlapping are formulated as quadratic inequalities. Integer variables are introduced to describe rotation. The resulting model in this work can be characterized as a nonlinear mixed integer programming problem. It is difficult to judge the effectiveness of this technique because only one simple example with three blocks was provided. Unlike that model, our model does not require the simplifications mentioned. In addition, we are using a linear mixed integer programming model instead of nonlinear. Our solution technique for the mixed integer programming problem allows us to solve placement problems with a substantial number of blocks.

There are several advantages of our algorithm over the existing algorithms mentioned above. Our algorithm considers simultaneously all factors involved in the development of the workable floorplans. Shape and area information for modules, areas for routing, assignment of connections to channels, and constraints on timing—all these factors can be taken into consideration. Unlike many other algorithms, our method provides the proven optimal solution for each subproblem solved in the process of construction of a global solution.

III. Mixed Integer Programming Model of Floorplanning

3.1. Problem Definition

Given a set of K_1 flexible modules and K_2 rigid modules, $K_1 \cup K_2 = K$. For each flexible module i ($i \in K_1$) area S_i is given, i.e., $w_i h_i = S_i$, where w_i and h_i are the unknown width and height of the module i. Constraints are imposed on the acceptable aspect ratios for each flexible module. For each $i \in K_1$ the inequality $a_i \le w_i/h_i \le b_i$ must be satisfied. For rigid modules ($i \in K_2$) w_i and h_i are given and it is assumed that the 90° rotation of modules is allowed. This is equivalent to conditions $a_i \le w_i/h_i \le b_i$ or $1/a_i \ge w_i/h_i \ge 1/b_i$. The connectivity information is given in the form of the *netlist*, where for each module a set of nets incident to the module is presented. This allows us to define a number of common nets c_{ij} for each pair of modules i and j. We define a floorplan by positions of the lower left corner of each module (x_i, y_i) in the system of coordinates with the center in the lower left corner of the chip. Nets are assigned to routing channels and estimations of channel widths are performed. Additional constraints on the length of critical nets can also be included. Among input information are the widths and spacings of wires for routing in both horizontal and vertical directions. This information is technology dependent. The optimal floorplan corresponds to a minimal covering rectangle R of a set of K nonoverlapping rectangles presenting modules and spaces between them reserved for interconnections.

3.2. First Mixed Integer Programming Formulation

Let us consider the constraints preventing overlapping of any pair of rectangular modules i and j. Let (w_i, h_i), (w_j, h_j) (assumed to be known at this point) denote the dimensions (width and height) of modules i and j, respectively. Let (x_i, y_i), and (x_j, y_j) (unknown variables) denote the position of the lower left corners of the modules i and j with respect to the center of coordinates.

To prevent overlapping of modules i and j, it is required that at least one of the following linear inequalities holds:

$$\begin{aligned} x_i + w_i &\le x_j, && i \text{ is to the left of } j \\ x_i - w_j &\ge x_j, && i \text{ is to the right of } j \\ y_i + h_i &\le y_j, && i \text{ is below } j \\ y_i - h_j &\ge y_j, && i \text{ is above } j. \end{aligned} \tag{1}$$

In order to ensure that at least one of these inequalities always holds we introduce two additional 0-1 integer variables x_{ij} and y_{ij}, which take only 0 to 1 values. Let us define bounding functions W and H, such that we always have $|x_i - x_j| \le W$ and $|y_i - y_j| \le H$.

Possible choices for W and H are: $W = W_{\max}$ (maximal width of the chip allowed) and $H = H_{\max}$ (maximal allowed height of the chip). If $W_{\max}$ and $H_{\max}$ are not known, then $W = \Sigma_{i=1}^{K} w_i$ and $H = \Sigma_{i=1}^{K} = h_i$ can be used. Now

consider the following system of linear inequalities for any pair of modules i and j:

$$\begin{aligned} x_i + w_i &\le x_j + W(x_{ij} + y_{ij}) \\ x_i - w_j &\ge x_j - W(1 - x_{ij} + y_{ij}) \\ y_i + h_i &\le y_j + H(1 + x_{ij} - y_{ij}) \\ y_i - h_j &\ge y_j - H(2 - x_{ij} - y_{ij}). \end{aligned} \quad (2)$$

It is easy to see that for each of the four possible choices of values for the integer variables, $(x_{ij}, y_{ij}) = (0, 0), (0, 1), (1, 0), (1, 1)$, only one of the four inequalities in (2) is active. For example, with $x_{ij} = 0$, $y_{ij} = 1$, only the third constraint applies, which allows module i to be anywhere below module j. The other three constraints are always satisfied for any permitted values of (x_i, y_i) and (x_j, y_j).

Let us assume that one dimension of the chip is known, say W, and that we need to determine a floorplan which corresponds to the minimal height. Then we can impose the following additional constraints:

$$\begin{aligned} x_i \ge 0, \quad y_i \ge 0, \quad & i \in K \\ x_i + w_i \le W, \quad & i \in K \\ y^* \ge y_i + h_i, \quad & \end{aligned}$$

where y^* is the height to be minimized. (3)

Hence, the first mixed integer programming formulation of the floorplanning problem for the case where all modules are rigid can be formulated as follows: *find minimum y^* subject to constraints (2) and (3).* This mixed integer programming formulation requires $2K$ continuous variables, $K(K - 1)$ integer variables, and $2K^2$ linear constraints. For example, if the number of modules $K = 25$, then the number of continuous variables is equal to 50, the number of integer variables is equal to 600, and the number of linear constraints is equal to 1250.

It is obvious that a better floorplan can be achieved if rotation of the rigid blocks is allowed. Our model can be easily extended to permit this. Let us introduce for each module i an additional 0-1 integer variable z_i, where $z_i = 0$ when rigid module i is placed in its initial orientation and $z_i = 1$ when module i is rotated 90°. The constraints of nonoverlapping for two rigid modules can be rewritten in the following form:

$$\begin{aligned} x_i + z_i h_i + (1 - z_i) w_i &\le x_j + M(x_{ij} + y_{ij}) \\ x_i - z_j h_j - (1 - z_j) w_j &\ge x_j - M(1 - x_{ij} + y_{ij}) \\ y_i + z_i w_i + (1 - z_i) h_i &\le y_j + M(1 + x_{ij} - y_{ij}) \\ y_i - z_j w_j - (1 - z_j) h_j &\ge y_j - M(2 - x_{ij} - y_{ij}) \end{aligned} \quad (4)$$

where

$$M = \max(W, H).$$

To keep the rotated rectangles within chip boundaries, the constraints (3) have to be rewritten as

$$\begin{aligned} x_i \ge 0, \quad y_i \ge 0, \quad & i \in K \\ x_i + (1 - z_i) w_i + z_i h_i \le W, \quad & i \in K \\ y^* \ge y_i + (1 - z_i) w_i + z_i h_i, \quad & \end{aligned}$$

where y^* is the height to be minimized. (5)

This initial model does not allow flexible modules, does not consider routing space and connection lengths, and requires substantial numbers of variables and constraints for an optimal solution. All these problems will be addressed next.

3.3. Flexible Modules in Integer Programming Formulation

The flexible model allows w_i and h_i to vary, but requires that the area S_i remain fixed, that is $w_i h_i = S_i$. We linearize this nonlinear relation about the point $w_{i\max}$, to give h_i as a linear function of w_i (see Fig. 1) by applying the first two members of the Taylor series:

$$h_i = \frac{S_i}{w_{i\max}} + (w_{i\max} - w_i) \frac{S_i}{w_{i\max}^2} \quad (6)$$

$$h_i = h_{i0} + \Delta w_i \lambda_i$$

where

$$h_{i0} = \frac{S_i}{w_{i\max}}, \quad \lambda_i = \frac{S_i}{w_{i\max}^2}, \quad \Delta w_i = w_{i\max} - w_i.$$

Therefore, we need one additional continuous variable Δw_i for each flexible module. Then the condition of nonoverlapping of flexible module i and rigid module j can be rewritten in the following manner:

$$\begin{aligned} x_i + w_{i\max} - \Delta w_i &\le x_j, && i \text{ is to the left of } j \\ y_i + h_{i0} + \Delta w_i \lambda_i &\le y_j, && i \text{ is below } j \\ x_i - w_j &\ge x_j, && i \text{ is to the right of } j \\ y_i - h_i &\ge y_j, && i \text{ is above } j. \end{aligned} \quad (7)$$

To make only one of these constraints active, two 0-1 integer variables x_{ij}, y_{ij} should be introduced for each pair of modules. A system of constraints emerging in this situation is very similar to the one described by the system of constraints (2).

If both modules i and j have flexible shapes, then conditions of nonoverlapping will look as follows:

$$\begin{aligned} x_i + w_{i\max} - \Delta w_i &\le x_j, && i \text{ is to the left of } j \\ y_i + h_{i0} + \Delta w_i \lambda_i &\le y_j, && i \text{ is below } j \\ x_j + w_{j\max} - \Delta w_j &\le x_i, && i \text{ is to the right of } j \\ y_j + h_{j0} + \Delta w_j \lambda_j &\le y_i, && i \text{ is above } j. \end{aligned} \quad (8)$$

The same technique shown in (4) is used to make only one of these constraints active.

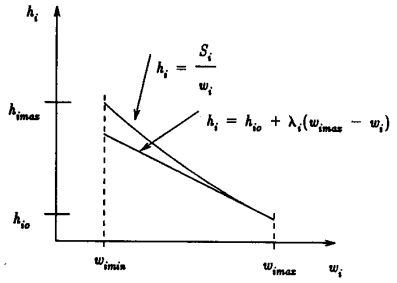


Fig. 1. Approximation function for h_i.

3.4. Constraints on Interconnection Length

A problem of identification of potentially critical paths and nets prior to physical design goes beyond the scope of this paper. This problem was addressed in [16]. We rely on the methodology described there to identify critical paths and nets. When critical paths are identified, this information can be used in floorplanning. The maximal allowed timing delay on interconnections for paths and nets is translated into maximal allowable length of interconnections for respective nets. Let L_s be a maximal allowable length of net s, and N_s be a set of pins connected by the net s. Let us introduce 6 new continuous positive variables for each net s $\bar{x}_s$, $\hat{x}_s$, $\bar{y}_s$, $\hat{y}_s$, x_s, y_s, such that

$$\begin{aligned} \bar{x}_s &= \max x_i, \quad \forall i \in N_s \\ \hat{x}_s &= \min x_i, \quad \forall i \in N_s \\ \bar{y}_s &= \max y_i, \quad \forall i \in N_s \\ \hat{y}_s &= \min y_i, \quad \forall i \in N_s \\ x_s &\geq \bar{x}_s - \hat{x}_s \\ y_s &\geq \bar{y}_s - \hat{y}_s. \end{aligned} \tag{9}$$

Then a half of perimeter $(x_s + y_s)$ of a covering rectangle can be used as an approximation of a Steiner tree connecting pins of the net s, and the length of the Steiner tree can be constrained by the L_s where:

$$x_s + y_s \leq L_s. \tag{10}$$

These constraints can be added for each critical net to the previously formulated sets of constraints for the floorplanning problem. Any increase in dimensionality of the model does not significantly affect the solution time for the following reasons. First, the number of critical nets represents only a small portion of all nets. Second, the time dependence on the number of continuous variables in a mixed integer linear programming problem is at most linear.

3.5. Constraints on Routability

First, we will consider design styles where routing is performed over the functional modules. Sea-of-gates design style and standard cells design style belong to this group. It is a well-known fact in the physical design that a chip is routable if the length of routing tracks available for routing is 1.5–2.0 times higher than the actual length of routed interconnections. It is also well known that the Manhattan distance between two terminals presents a lower bound on the net length. Therefore, the number of the tracks available for routing should be at least 2 times higher than the number of required tracks for Manhattan routing. This means that bin-packing algorithms cannot be used for floorplanning when the number of available routing tracks for routing over the modules is not sufficient to accommodate interconnections. Empty area around the modules should be reserved for routing. Let us introduce two additional continuous variables ξ_{ij1} and ξ_{ij2} for each pair of modules i and j such that $|x_i - x_j| \leq \xi_{ij1}$ and $|y_i - y_j| \leq \xi_{ij2}$, i.e.:

$$\begin{aligned} -\xi_{ij1} &\leq x_i - x_j \leq \xi_{ij1} \\ -\xi_{ij2} &\leq y_i - y_j \leq \xi_{ij2}. \end{aligned} \tag{11}$$

Then the conditions of balancing for supply and demand of routing tracks in both dimensions will have the following forms:

$$\begin{aligned} \sum_i \sum_j c_{ij}\xi_{ij1} &\leq 0.5 y^* n_h W \\ \sum_i \sum_j c_{ij}\xi_{ij2} &\leq 0.5 y^* n_v W. \end{aligned} \tag{12}$$

Here c_{ij} is the number of common nets between modules i and j. n_h and n_v are the numbers of routing tracks per unit length in horizontal and vertical directions respectively. y^* is the unknown height of the chip and W is the given width of the chip. The conditions (12) should be added to the set of conditions for the floorplanning problem.

If the width of the chip W is also unknown, then the desired aspect ratio $a = w/y^*$ for the chip can be used. The constraints (12) can be rewritten as

$$\begin{aligned} \sum_i \sum_j c_{ij}\xi_{ij1} &\leq 0.5 a (y^*)^2 n_h \\ \sum_i \sum_j c_{ij}\xi_{ij2} &\leq 0.5 a (y^*)^2 n_v. \end{aligned} \tag{12a}$$

To preserve linearity of the constraints (12a), $(y^*)^2$ can be presented as $(y^*)^2 = y_{\min}^2 + (y^* - y_{\min})y_{\min}$, where $y_{\min}$ is the minimal reasonable height of the chip, which is consistent with the aspect ratio a. Then balancing conditions can be rewritten again:

$$\begin{aligned} \sum_i \sum_j c_{ij}\xi_{ij1} &\leq 0.5 a n_h [y_{\min}^2 + (y^* - y_{\min})y_{\min}] \\ \sum_i \sum_j c_{ij}\xi_{ij2} &\leq 0.5 a n_v [y_{\min}^2 + (y^* - y_{\min})y_{\min}]. \end{aligned} \tag{12b}$$

In this form, constraints (12b) are linear with respect to y^*.

3.6. Floorplanning with Routing Around the Modules

Very often modules are designed in such a way that all terminals are located on the periphery and interconnec-

tions are expected to be positioned in channels around the modules. In this situation additional space for routing should be included in the evaluation of the chip size. To keep this additional requirement to a minimum we include in the objective function the member F_r with the weight α_r:

$$F_r = \sum_i \sum_j c_{ij}(\mu_h \xi_{ij1} + \mu_v \xi_{ij2}) \tag{13}$$

where c_{ij}, ξ_{ij1}, and ξ_{ij2} are, as before, the connectivity, the horizontal, and the vertical components of the interconnections between modules i and j. They obey the constraints (11) given in Section III-3.5. μ_h and μ_v are widths of tracks on horizontal and vertical layers, respectively.

3.7. Optimization of the Floorplan with the Given Topology

One of the often mentioned formulations of the floorplanning problem assumes that the topology of the chip is given and only shapes of the modules should be optimized. When the mixed integer programming formulation is applied to this problem, it results in elimination of all integer variables. These variables will assume values 0 or 1 according to the mutual positions of modules. It means that it is known for any two modules i and j which one of them is on the top of another, or to the right of another. For example, if module j is on the right from module i according to the given topology, then $x_{ij} = 0$, $y_{ij} = 0$, and only one inequality $x_i + w_i \leq x_j$ is needed. As a result, the number of integer variables for this formulation is equal to zero. The *NP*-hard problem of floorplanning becomes polynomially solvable when topology is defined. The number of continuous variables is $2K$ and the number of linear constraints is $O(K)$ with the coefficient dependent on the average number of neighbors for each module for the given topology. A solution for such a problem can be easily obtained by an application on any standard linear programming software.

3.8. More on Objective Functions for Mixed Integer Programming Formulations

a) The optimal solution of the floorplanning problem is characterized by the minimal chip area $F = x^* \times y^*$, where x^* and y^* are the optimal width and height of the chip, respectively. However, very often the desired aspect ratio $a = x^*/y^*$ is given for the chip. This allows a description of the chip area as a function of one variable $F = a(y^*)^2$. The minimum of the function $F(y^* \geq 0)$ is achieved for the same value of y^* as the minimum of the function $F_1 = y^*$. This fact allows the use of a linear objective function instead of the quadratic one and presents the constraints (3) from original integer programming problem formulation in the following form:

$$\begin{aligned} x_i \geq 0, \quad y_i \geq 0, \qquad & i \in K \\ x_i + w_i \leq a y^*, \qquad & i \in K \\ y^* \geq y_i + h_i. & \end{aligned} \tag{14}$$

All other constraints on nonoverlapping will be the same as listed before.

b) One possible definition of the problem can be to find a floorplan described by the minimal resulting interconnection length. This can be achieved by using an objective function $F = \Sigma_s (x_s + y_s) \rightarrow \min$ and constraints on nonoverlapping given by (2) and (3), and the constraints on interconnection length (10). Here x_s and y_s are defined by (10). Another reasonable objective function for floorplanning is a linear combination of the chip size and the total interconnection length $F = y^*W + \alpha \Sigma_s (x_s + y_s)$, where α is a weight coefficient. This formulation can be extended to emphasize critical nets. Then $F = \Sigma_s \alpha_s (x_s + y_s)$, where α_s is the weight coefficient for the net s.

c) As was mentioned in Section III-3.6, an additional term can be included in the objective function to provide space for interconnections. This term (13) can be added to all other terms in the objective function mentioned above.

IV. Solution Method for Mixed Integer Programming Model of Floorplanning

The solution for the mixed integer linear programming models described in the previous paragraph can be obtained by the standard mathematical programming software. There are several software packages available. We applied the widely used LINDO [17] package for this purpose. This version of LINDO allows the introduction of up to 200 0-1 integer variables and a few thousand linear constraints. As was mentioned earlier, the number of integer variables used by the floorplanning model is significant. It was shown in the example given in Section III-3.2 that this number is 600 for a floorplan problem with 25 modules. The solution time for the integer programming problems grows exponentially (in the worst case) with the number of integer variables. In practical cases this growth is not exponential, but still can be very fast. This dictates a necessity for a technique that will consider only a limited number of modules (say 10-12) at a time. This technique is called successive augmentation (see Fig. 2). The main idea of the successive augmentation is to create a final floorplan by adding a new group of modules to a partial floorplan in an optimal way until all the modules are positioned. The successive augmentation does not guarantee an optimal solution for the original mixed integer programming problem. It guarantees optimality on each step of the process and, as a result, a suboptimal solution of the original problem. This method represents generalization of the greedy idea from one variable to a group of variables. Application of the successive augmentation in conjunction with the mixed integer programming requires solutions for several problems:

1) how to select a new group of modules to be added to the partial floorplan?

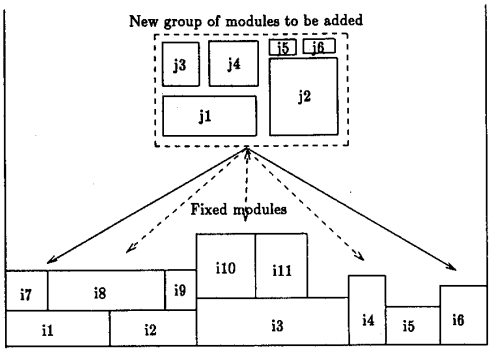


Fig. 2. Floorplanning by successive augmentation.

```
Procedure FloorplanDesign;
begin
(1)  Select a group of m modules as a seed;
(2)  Formulate a system of linear constraints for these unpositioned modules;
(3)  Call an integer programming procedure to obtain the first partial floorplan;
(4)  while (m ≤ k) do /* k is the total number of modules */
(5)     Select a new group of e modules based on the connectivity to the
           already fixed modules in the partial floorplan and timing considerations;
(7)     Find a set of d covering rectangles for the partial floorplan, where d ≤ m;
(8)     Formulate a system of linear constraints for d covering rectangles
           and e unpositioned modules;
(9)     Call an integer programming procedure to obtain a new partial floorplan;
(10)    m = m + e;
(11) endwhile
(12) Perform global routing;
(13) Adjust floorplan;
end; /* Procedure FloorplanDesign */
```

Fig. 3. An algorithm for floorplan design.

2) how to minimize the number of integer variables for each solved integer programming problem?

These questions will be answered in the following sections.

4.1. An Algorithm for Solving the Floorplanning Problem

As it was stated above a new group of modules is added to the partial floorplan on each step of the successive augmentation algorithm. Different criteria can be applied to their selection. One popular technique is to perform clustering of modules based on the connectivity prior to floorplanning and then add a cluster to the partial floorplan at each step of the process. Another technique is to apply a linear ordering procedure [18], using as a seed one of the modules with the highest number of I/O connections. A new group of modules for floorplanning is selected according to their position in the linear order. Both clustering and linear ordering techniques allow weighting factors to emphasize critical nets. As a result, timing problems can be addressed on three steps of the algorithm. The first time is when a new group of modules is selected. Modules of the critical nets may have priority in the order. The second time is in the formulation of the integer programming problem by assigning weight coefficients to critical nets. The third time is by routing critical nets first when global routing is performed. All of these techniques are available to users of our package. The description of the floorplanning algorithm is given in Fig. 3.

4.2. Covering Rectangles for the Partial Floorplan

The number of integer variables in the problem depends on the number of modules. Their mutual positions should be described by constraints. One resource in reducing dimensionality for the integer programming problem is replacement of already positioned modules by a set of covering rectangles. If the number of covering rectangles produced is less than the number of modules replaced, then reduction of the number of integer variables is achieved and the execution time is reduced.

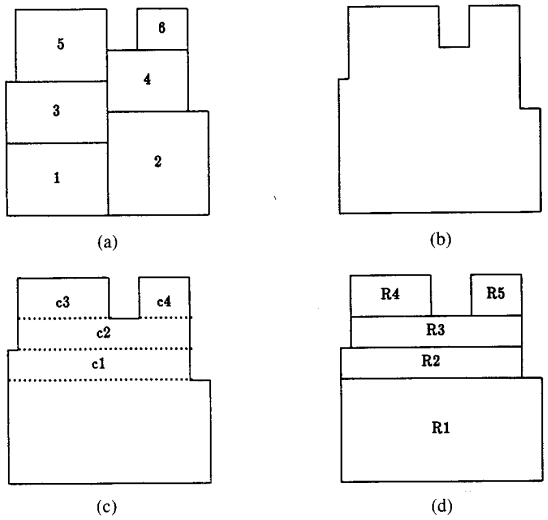


Fig. 4. Partitioning of a partial floorplan. (a) A partial floorplan. (b) A covering polygon for the floorplan. (c) Horizontal edge-cuts. (d) Partitioning of the polygon.

Fig. 4(a) shows a set of six fixed modules which can be represented as a hole-free polygon in Fig. 4(b). This polygon is then partitioned in the horizontal direction as shown in Fig. 4(c). A set of five covering rectangles of this polygon is presented in Fig. 4(d).

The following theorems show that the number of the covering rectangles produced by decomposition of the polygon is always less than or equal to the number of fixed modules forming the polygon. In our case, the polygon of the partial floorplan always has a special feature—a flat bottom. Holes at the bottom of the polygon are ignored because new modules are added only from the open side of the chip. Some properties of this procedure are given below.

Let N be a number of the fixed modules of the partial floorplan and n be a number of horizontal edges of the polygon covering the partial floorplan. A horizontal edge-

cut is a horizontal line that comprises at least one horizontal edge and cuts from the polygon one rectangle.

Theorem 1: $n \leq N + 1$.

Proof: There are $2N$ horizontal edges for a set of N rectilinear modules. According to the floorplan design procedure, each module is placed either on the bottom edge of the chip or on top of another module. Then the bottom edge of each module is overlapping with the top edge of the module below, except for the modules placed on the bottom of the chip. The overlapping edges are merged into one edge which is a union of component edges. The number of horizontal edges produced by such procedure is at most $2N - (N - 1) = N + 1$. The number of horizontal edges n of the covering rectangle cannot exceed this number.

Let N^* be the number of partitions produced by the partitioning procedure.

Theorem 2: $N^* \leq n - 1$.

Proof: The procedure starts from the bottom of the chip and produces a cut line for the first from the bottom horizontal edge of the covering. This procedure separates from the polygon a rectangle where the bottom edge is the bottom of the chip and the top edge is an edge-cut line. This process is repeated for each horizontal edge with an increased y-coordinate. Because this process starts not from the bottom but from the first horizontal edge with $y > 0$, then at most $n - 1$ steps will be executed and $n - 1$ rectangles are generated. Therefore, $N^* \leq n - 1$.

Corollary: $N^* \leq N$.

Proof: Since $n \leq N + 1$ and $N^* \leq n - 1$, it follows that $N^* \leq N$.

Reduction of the number of rectangles for the integer programming step is usually very substantial. A further reduction can be achieved if a set of overlapping partitions is used instead of nonoverlapping partitions to cover the partial floorplan.

4.3. Global Routing and Final Chip Area Computation

It is clear that a workable floorplan should allocate an area for interconnections. Different floorplans should be compared not by the area produced by close packing of modules, but by the area, which includes modules and space occupied by routing. This is especially true for design styles with routing around the modules. However, there are very serious obstacles for the accurate prediction of space required for interconnections. We will list some of these obstacles. Flexible modules are given only by their areas, bounds on the aspect ratios, and their connectivity to other modules. Dimensions of such modules are not known prior to floorplanning, and therefore, it is not known how many pins can be assigned to each side of the module. This situation cannot be described analytically because many additional factors are involved in the pin assignment. A major role here is played by the internal structure of each module.

In our floorplanning methodology it is assumed that eight potential locations are available for generalized pins (two locations on each side). Preliminary pin assignment takes place during global routing. Instead of considering a center of a module as a generalized pin position for all nets, we consider one of these eight generalized pins on sides for each net. A combination of pin positions on different modules which corresponds to the shortest net length is selected. A number of nets connected to one generalized pin does not exceed $\lceil n_i l_{ik}/2L_i \rceil$, where n_i is a number of nets connected to modules i, l_{ik} is a length of side k of module i ($k = 1, 2, 3, 4$), and L_i is a perimeter of module i. This model provides a more realistic evaluation of the routing space required for interconnections than the one with pins located at the center of the module.

The next important question is how to define positions and widths of routing channels. In our methodology this problem is solved in several steps. As the first step, we include the area for interconnections F_r given by (13) in the objective function of the integer programming problem. Then the minimized objective is a combined area of the modules and interconnections. In addition, if pin assignment is known, each module is placed into an envelope, which exceeds the initial size of each side on the value $\mu_\alpha \Sigma_i c_{igp}$, where μ_α ($\alpha = h, v$) is metal width plus spacing for one routing track in horizontal (vertical) direction. c_{igp} is a connectivity of module i ($i = 1, \cdots, K$) to side p of module g ($p = 1, 2, 3, 4$).

For example, if 6 pins were assumed on the top side of the module and 4 pins on the bottom side, then an envelope exceeds the vertical dimension of the module on the value $10\mu_h$, and the widths of the channels on the top and on the bottom would be $6\mu_h$ and $4\mu_h$, respectively. These enhanced dimensions of modules are used to solve a placement problem for the partial floorplans by an integer programming technique. On the next step of the algorithm a global routing is performed for the system of channels defined by the use of extended or original modules. Our global router is graph based. It uses the channel position graph [11], [12] obtained from the floorplan produced by the integer programming step. It uses the shortest path algorithm to find a route between two generalized pins. It also uses a penalty function for utilization of a heavily used channel. Nets with the tight timing requirements are routed first. On the final step of the algorithm, widths of channels are adjusted to accommodate results of the global routing and final chip area, and wirelength is computed.

V. Experimental Results

Our methodology of floorplanning was implemented in a Fortran77 program running on Apollo DN3550 workstations (4 Mips). It performs periodic calls to the LINDO

TABLE I
RESULTS ON PACKING OF THE BENCHMARK AMI33 (PRIM BBL2)

Algorithm	Chip Utilization (%)	Wirelength (m)
GORDIAN	79.5	76.26
BEAR	90.09	106.07
Minnesota	96.23	71.27

TABLE II
RESULTS ON FINAL FLOORPLAN WITH ROUTING SPACE

Algorithm	Chip Area (mm^2)	Wirelength (mm)
MOSAICO	3.16	151.824
VITAL	3.12	134.599
Seattle Silicon	2.94	125.000
BEAR	2.83	131.244
Delft (Manual Placement)	2.60	151.656
Minnesota	2.54	139.810

[17] package, which runs on the same computer and is executed as a procedure. For easy comparisons of our results with those published earlier, a benchmark provided by the Workshop on Physical Design 1988, has been used. This benchmark, AMI33, (also known as PrimBBL2), includes 33 modules and 123 nets. As was mentioned in Section II, this benchmark is widely used by many researchers [11]–[13], and others.

The first set of experimental data describes application of different algorithms to packing of the benchmark AMI33. It is assumed that routing is performed over the modules. Analogous assumption is made in [11] and [13]. Wiring length is measured as a net half perimeter. Comparative data on obtained solutions are given in Table I. A solution provided by our system, called Minnesota, obtained the highest level of area utilization and smallest wiring length by effectively utilizing the flexible shape option. It took 6 min to construct this 33-module floorplan.

The second set of experimental data on a floorplan with around-the-module routing, which was assembled at the 1988 Workshop on Placement and Routing, was published in [11]. We added our experimental data to the data presented there and merged them into Table II. The solution is also presented in the Fig. 5. The Minnesota system took 16 min to construct the floorplan and route it.

VI. CONCLUSION

A system described in this paper utilizes a standard mathematical software as a major subroutine for solving floorplanning problems. Experiments demonstrated the effectiveness of this approach. Results can be further improved by an application of detailed routing procedure followed by a compaction.

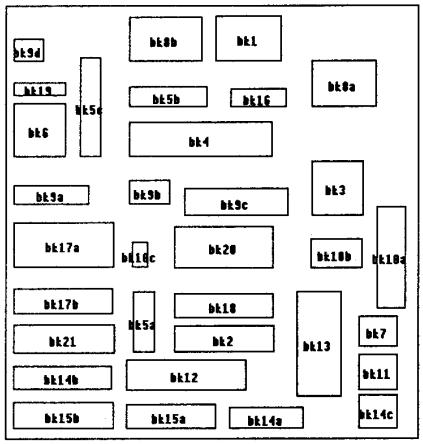


Fig. 5. The final floorplan with routing space.

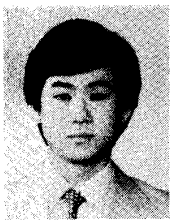
ACKNOWLEDGMENT

The authors would like to thank Dr. K. Kozminski from MCNC for providing information on benchmarks and the reviewers for their useful suggestions.

REFERENCES

[1] R. H. M. Otten, "Automatic floorplan design," in *Proc. 19th Design Automation Conf.*, 1982, pp. 261–267.
[2] ——, "Efficient floorplan optimization," in *Proc. Int. Conf. on Computer-Aided Design*, 1983, pp. 499–502.
[3] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proc. 23rd Design Automation Conf.*, 1986, pp. 101–107.
[4] D. F. Wong and P. S. Sakhamuri, "Efficient floorplan area optimization," in *Proc. 26th Design Automation Conf.*, 1989, pp. 586–589.
[5] T. Mueller, D. F. Wong, and C. L. Liu, "An enhanced bottom-up algorithm for floorplan design," in *Proc. 24th Design Automation Conf.*, 1987, pp. 524–527.
[6] W. R. Heller, G. Sorkin, and K. Maling, "The planar package planner for system designers," in *Proc. 19th Design Automation Conf.*, 1982, pp. 253–260.
[7] K. A. Kozminski and E. Kinnen, "An algorithm for finding a rectangular dual of a planar graph for use in area planning for VLSI integrated circuits," in *Proc. 21st Design Automation Conf.*, 1984, pp. 655–656.
[8] ——, "Rectangular dualization and rectangular dissections," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1401–1416, Nov. 1988.
[9] J. Bhasker and S. Sahni, "A linear algorithm to find a rectangular dual of a planar triangulated graph," in *Proc. 23rd Design Automation Conf.*, 1986, pp. 108–114.
[10] B. Lokanathan and E. Kinnen, "Performance optimized floor planning by graph planarization," in *Proc. 26th Design Automation Conf.*, 1989, pp. 116–121.
[11] B. Eschermann, *et al.*, "Hierarchical placement of macrocells: A meet in the middle approach," in *Proc. Int. Conf. on Computer-Aided Design*, 1988, pp. 460–463.
[12] H. Cai, "Connectivity biased channel construction and ordering for building-block layout," in *Proc. 25th Design Automation Conf.*, 1988, pp. 560–565.
[13] J. M. Kleinhans, G. Sigl, and F. M. Johannes, "GORDIAN: A new global optimization/rectangle dissection method for cell placement," in *Proc. Int. Conf. on Computer-Aided Design*, 1988, pp. 506–509.
[14] L. Sha and T. Blank, "ATLAS—A technique for layout using analytic shapes," in *Proc. Int. Conf. on Computer-Aided Design*, 1987, pp. 84–87.

[15] L. Markov, J. R. Fox, and J. H. Blank, "Optimization techniques for two-dimensional placement," in *Proc. 21st Design Automation Conf.*, 1984, pp. 652–654.
[16] H. Youssef, E. Shragowitz, and L. Bening, "Critical path issue in VLSI design," in *Proc. Int. Conf. on Computer-Aided Design*, 1989, pp. 520–523.
[17] L. Schrage, *LINDO: Linear Interactive Discrete Optimizer.* Readwood City, CA: Scientific, 1982.
[18] S. Kang, "Linear ordering and application to placement," in *Proc. 20th Design Automation Conf.*, 1983, pp. 457–464.



Suphachai Sutanthavibul received the B.S. degree in computer engineering from Chulalongkorn University in Thailand in 1983 and the M.S. and Ph.D. degrees in computer science from the University of Minnesota in 1987 and 1990, respectively.

Currently he is working with IBM, Kingston, NY. His research interests include timing and physical design aspects of CAD for VLSI.

Dr. Sutanthavibul is a member of ACM.

Eugene Shragowitz received the M.S. degree in electrical engineering from the College of Electrical Engineers, Leningrad, U.S.S.R., in 1958 and the Ph.D. degree from the National Scientific Research Institute, Moscow, U.S.S.R., in 1971.

From 1981 to 1986 he worked as the Technical Consultant in the Advanced CAD Department of Control Data Corporation, Minneapolis, MN. He is currently an Associate Professor of Computer Science at the University of Minnesota. His research interests are in CAD of VLSI, nonlinear network theory, combinatories, and parallel algorithms.

J. B. Rosen received the undergraduate degree in electrical engineering from Johns Hopkins University in 1943, and the Ph.D. degree in applied mathematics from Columbia University in 1952.

He is currently a Professor of Computer Science and a Fellow of the Minnesota Supercomputer Institute, University of Minnesota. His current research interests include large scale numerical optimization methods and parallel computation for continuous and discrete problems. He recently edited a volume on supercomputers and large-scale optimization.