A Fixed-die Floorplanning Algorithm Using an Analytical Approach

Yong Zhan, Yan Feng, and Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455, USA

***Abstract*— Fixed-die floorplanning is an important problem in the modern physical design process. An effective floorplanning algorithm is crucial to improving both the quality and the time-to-market of the design. In this paper, we present an analytical floorplanning algorithm that can be used to efficiently pack soft modules into a fixed die. The locations and sizing of the modules are simultaneously optimized so that a minimum total wire length is achieved. Experiments on the MCNC and GSRC benchmarks show that our algorithm can achieve above a 90% success rate with a 10% white space constraint in the fixed die, and the efficiency is much higher than that of the simulated annealing based algorithms for benchmarks containing a large number of modules.**

I. Introduction

Floorplanning is a crucial step in early stages of the physical design process. A high quality floorplan with small wire lengths and low white space will have a positive impact on both the performance and the yield of the final manufactured ICs. The high complexity of modern VLSI systems has made hierarchical design the preferred methodology even for the floorplanning stage. Hence, as pointed out in [1], the problem of fixed-die floorplanning, in which the outline of the floorplan is predetermined, has become more relevant than outline-free floorplanning, because at a lower level of the design hierarchy, the floorplan of a sub-system must be confined to the outline set by the higher level of hierarchy that is immediately above it.

During the past few years, several works have been performed in the direction of solving the fixed-die floorplanning problem. In [2], a simulated annealing based algorithm was presented, and slack-based moves were introduced to facilitate the reduction of the floorplan span in a given direction, and in [3], an evolutionary search approach was used to handle the fixed-die floorplanning problem, based on normalized Polish expressions. In [4], Chen and Chang proposed a novel cooling scheme for the simulated annealing process such that the runtime of the algorithm was significantly reduced, while at the same time, the quality of the resulting floorplan was improved. In [5], partitioning was effectively combined with the simulated annealing algorithm to make the later much more scalable with respect to problem size.

Another direction in the research of floorplanning problems is to study the representations of the geometric relationships among modules so that the algorithms such as the simulated annealing can be implemented more effectively. Examples of some of the recent works concerning the floorplan representations include the sequence pair method [6], BSG [7], O-tree [8], CBL [9], B*-tree [10], and TCG [11].

The fixed-die floorplanning problem can be considered as one of packing rectangular-shaped modules into a fixed outline. Two types of modules can be involved in a floorplanning problem, i.e., hard modules, whose shape cannot change during the floorplanning process, and soft modules, whose area remains the same but whose aspect ratio can vary. Over the years, the annealing-based algorithms have made remarkable progress in the floorplanning of hard modules. Nowadays, a state-of-the-art annealing-based floorplanning algorithm can achieve a good floorplan containing hundreds of hard modules within minutes. For soft modules, however, the results from the annealing-based algorithms are not as satisfactory. This either shows up as a long runtime to execute the algorithm or a low success rate[1]. One of the reasons for the low performance of annealing algorithms on the problem of floorplanning with soft modules is that the sizing of the modules adds more dimensions to the optimization problem, and hence increases the difficulty of the annealing process.

The floorplanning of soft modules remains an important problem because at the floorplanning stage, the detailed layout of modules has not usually been obtained yet. Hence, the rectangular-shaped modules still have certain flexibility in changing their aspect ratios. Several previous works have tackled the problem of effective floorplanning and sizing of soft modules [12] [13] [14]. However, these works were not performed in the fixed-die context. In [15], a highly efficient bipartitioning-based algorithm was proposed that can effectively deal with the soft modules in the fixed-die floorplanning problem. Nevertheless, the success rate of this algorithm is sensitive to the input benchmark and the constraints such as the maximum allowed aspect ratios of the modules. In this paper, we present a fixed-die floorplanning algorithm based on an analytical approach that can be used to efficiently pack soft modules into the fixed die while minimizing the total wire length. The success rate of the algorithm is benchmark and constraint-insensitive. The floorplanning problem is formulated into a constrained optimization problem, and the location and sizing for each of the modules are obtained simultaneously. The optimization algorithm is divided into two stages, i.e., rough floorplanning, followed by overlap reduction and final legalization. In the first stage, we have adopted a method similar to that used in [16] and [17] for the placement problem of standard cells to spread the modules relatively uniformly across the die. The difference between the floorplanning problem of soft modules and the placement problem of standard cells is that the modules in the floorplanning stage can have significant difference in both the widths and heights, and the aspect ratios of the modules can vary during the optimization process. Hence, besides the center coordinates of the modules, the widths of the modules also enter the floorplanning problem as optimization variables to take care of the module-sizing issue. In the second stage, we first use an optimization-based approach to effectively reduce the overlaps between modules in the rough floorplan. Then we send the improved floorplan, which already has little or no overlap between modules, to the $pl2sp()$ routine in Parquet-4 [2], whose function is to shift some of the modules so that a overlap-free floorplan is obtained. No further sizing or switching the order between modules is performed in $pl2sp()$. The operations that take place in the second stage of the algorithm are in contrast to the simple greedy algorithm used in [17] for the legalization of the placement of standard cells immediately after the rough placement, which usually fails to generate a legal floorplan in our situation because of the significant difference in both the widths and heights of the modules. Experimental results on the MCNC and GSRC benchmarks show that our method can achieve above a 90% success rate with a 10% white space constraint in the fixed die, while the runtime is almost linear with respect to the number of modules in the design.

The rest of the paper will be organized as follows. Section II formulates the fixed-die floorplanning problem and presents the two-stage optimization algorithm using an analytical approach. Section III shows the experimental results in terms of the runtime, success rate, and the total half perimeter wire length (HPWL) in the final floorplan, and the conclusions are provided in section IV.

This work was supported in part by DARPA under grant N66001-04-1-8909, SRC under contract 2003-TJ-1092, and NSF under award CCR-0205227.

[1]The success rate of a set of annealing runs is defined as the ratio of the number of runs that result in a floorplan that can fit into the fixed die, to the total number of annealing runs.

II. Problem Formulation and the Analytical Floorplanning Algorithm

A. Problem formulation

As stated previously, our goal is to develop an efficient algorithm to pack rectangular-shaped soft modules into a fixed die while minimizing the total wire length. The input to the algorithm includes the width and height of the die, the area of each module, and the maximum allowed aspect ratios of the modules. The output of the algorithm is the final floorplan within the fixed die which includes the location and sizing for each of the modules. Hence, our fixed-die floorplanning problem for soft modules is formulated as

Given the dimensions of the fixed die, L_x and L_y, and assuming that the area and the maximum allowed aspect ratio of the i^{th} module are A_i and R_i, respectively, find the location and sizing for each of the modules such that the total wire length is minimized and the final floorplan is within the fixed die.

Our algorithm achieves the objective in two stages, i.e., rough floorplanning, followed by overlap reduction and final legalization. The objective of the first stage is to find the approximate location and sizing for each of the modules such that the total wire length is small, while the modules are spread relatively uniformly across the fixed die. Some overlap is allowed in this stage. The objective of the second stage is to fine-tune the location and sizing for each of the modules such that a floorplan without any overlap between modules is obtained.

B. Rough floorplanning

In the rough floorplanning stage, we try to minimize the total wire length, *WL*, and spread the modules as uniformly as possible across the fixed die. The spreading of modules is characterized by a penalty term P_D, which describes the non-uniformity of the module distribution.

Let x_i, y_i, and w_i be the center coordinates and the width of module i, respectively. Then the optimization problem that we solve in the rough floorplanning stage is

$$\begin{aligned} \text{minimize} \quad & WL + \alpha \cdot P_D \\ \text{such that} \quad & 0 \le x_i \le L_x \\ & 0 \le y_i \le L_y \\ & \sqrt{\frac{A_i}{R_i}} \le w_i \le \sqrt{A_i R_i} \\ & i = 1, 2, \ldots, n \end{aligned} \tag{1}$$

where n is the total number of modules, and the weighting factor α is used to determine the relative significance of the penalty term P_D with respect to the wire length *WL*. The first two constraints state that the center of each module should not exceed the boundary of the fixed-die while the last constraint ensures that the real aspect ratio of each module is within its maximum allowed range. Note that these constraints do not exclude the possibility that some of the modules may lie partly outside the fixed-die. This issue is taken care of by the P_D term in our implementation and will be discussed further when we present the calculation of P_D.

The constrained optimization problem (1) can be conveniently transformed into an unconstrained optimization problem and solved using the Sequential Unconstrained Minimization Technique (SUMT) [18]. The unconstrained optimization problem has the form

$$\text{minimize} \quad WL + \alpha \cdot P_D + \beta \cdot B \tag{2}$$

where B is the barrier term created to take care of the removed constraints in (1) and β is the weighting factor associated with B. One form of the barrier term B is given by

$$\begin{aligned} B = & \sum_{i=1}^{n} \frac{1}{x_i} + \sum_{i=1}^{n} \frac{1}{L_x - x_i} \\ & + \sum_{i=1}^{n} \frac{1}{y_i} + \sum_{i=1}^{n} \frac{1}{L_y - y_i} \\ & + \sum_{i=1}^{n} \frac{1}{w_i - \sqrt{\frac{A_i}{R_i}}} + \sum_{i=1}^{n} \frac{1}{\sqrt{A_i R_i} - w_i} \end{aligned} \tag{3}$$

The key property of the objective function in (2) and the barrier function B in (3) is that if we start from a point within the feasible region determined by the constraints in (1) and use any kind of iterative searching algorithm, e.g., the conjugate gradient method, to solve the unconstrained optimization problem, the points being searched will always remain feasible. This is because B approaches ∞ as the boundary of the feasible region is approached, and hence, the search process will never go outside the feasible region. A convenient starting point is obtained by choosing x_i, y_i, and w_i randomly within their corresponding bounds.

The total wire length *WL* is calculated using a quadratic formula. Since the quadratic formula only works with two-pin nets, we first use the *clique* model to decompose all of the multi-pin nets into two-pin nets. Assume modules i_1, i_2, ..., i_j are connected by multi-pin net i with weight k_i. The clique model removes the multi-pin net and reconnects every pair of modules in i_1, i_2, ..., i_j with a two-pin net with a weight of $\frac{2k_i}{j(j-1)}$. After all of the multi-pin nets have been replaced by the corresponding two-pin nets, the total quadratic wire length *WL* can be calculated using

$$WL = \sum_{i,j} k_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2] \tag{4}$$

where the indices i and j run through all of the modules connected by two-pin nets, and the k_{ij}'s are the net weights obtained from the clique model.

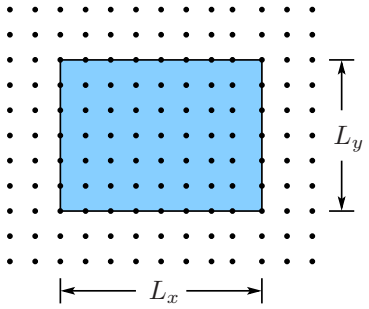


Fig. 1: Grid for calculating the penalty term P_D. The shaded region represents the fixed die. The points inside the die constitute the set C_{in}, and the points outside the die constitute the set C_{out}.

As stated previously, the penalty term P_D is used to characterize the non-uniformity in the distribution of modules across the fixed die. We have adopted a method similar to that presented in [16] and [17] for the placement problem to calculate P_D. Specifically, we first define the concept of area density, D_R, within a region R, which is given by

$$D_R = \frac{S_A}{S_R} \tag{5}$$

where S_R is the area of region R, and S_A is the sum of the overlap areas of all of the modules with region R. Note that as $S_R \to 0$, we obtain the area density of a point. Under this definition, if the modules are uniformly distributed within the fixed die, and a legal floorplan is obtained, the area density inside the die should be $\bar{D} = (\sum_{i=1}^{n} A_i)/(L_x L_y)$ while the area density outside the die should be 0.

During the optimization process, to keep track of the non-uniformity

of the area density, we superimpose an array of monitoring points over the die area and its surrounding region as shown in Fig. 1. The penalty term P_D is then calculated as the sum of squares of the variations of the densities from their ideal values, over all of the monitoring points. Specifically, let C_{in} and C_{out} represent the sets of monitoring points inside and outside the die area, respectively, and let D_i be the actual area density at monitoring point i, then the penalty term P_D can be calculated by

$$P_D = \sum_{i \in C_{in}} (D_i - \bar{D})^2 + \sum_{i \in C_{out}} D_i^2 \tag{6}$$

The second term in the expression of P_D penalizes the modules that go partly outside the fixed die during the optimization process.

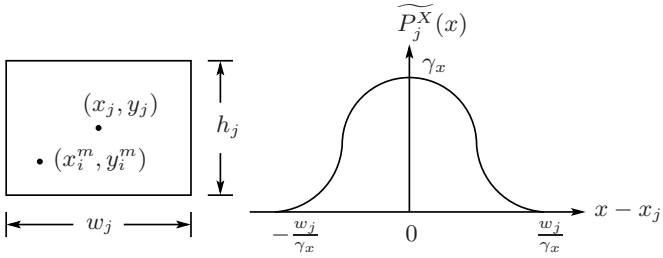


Fig. 2: Calculation of the area density function (a) coordinates of the module and the monitoring point (b) bell-shaped function used to approximate the contribution to the area density from a module.

In equation (6), the actual area density D_i at monitoring point i is calculated by summing up the contributions from all of the modules in the floorplan, i.e.,

$$D_i = \sum_{j=1}^{n} P_j(x_i^m, y_i^m) \tag{7}$$

where $P_j(x, y)$ is the contribution to the area density distribution from module j and (x_i^m, y_i^m) is the location of monitoring point i as shown in Fig. 2(a). Ideally, $P_j(x, y)$ is 1 if module j overlaps with point (x, y), and 0 otherwise, i.e.,

$$P_j(x,y) = \begin{cases} 1 & \text{if } |x - x_j| \le \frac{w_j}{2} \text{ and } |y - y_j| \le \frac{h_j}{2} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where $h_j = \frac{A_j}{w_j}$ is the height of module j. However, the D_i obtained this way is not a smooth function of the optimization variables x_j and y_j, which may cause practical difficulties in applying gradient-based optimization algorithms. To resolve this difficulty, we can use a bell-shaped smooth function $\widetilde{P_j}(x, y)$ to approximate the effect of module j on the area density distribution [16] [17]. The function $\widetilde{P_j}(x, y)$ can be decomposed into

$$\widetilde{P_j}(x,y) = \widetilde{P_j^X}(x) \times \widetilde{P_j^Y}(y) \tag{9}$$

where

$$\widetilde{P_j^X}(x) = \begin{cases} \gamma_x \times \left(1 - \frac{2(x-x_j)^2}{(\frac{w_j}{\gamma_x})^2}\right) & \text{if } 0 \le |x - x_j| \le \frac{w_j}{2\gamma_x} \\ \gamma_x \times \frac{2(|x-x_j| - \frac{w_j}{\gamma_x})^2}{(\frac{w_j}{\gamma_x})^2} & \text{if } \frac{w_j}{2\gamma_x} < |x - x_j| \le \frac{w_j}{\gamma_x} \end{cases} \tag{10}$$

and

$$\widetilde{P_j^Y}(y) = \begin{cases} \gamma_y \times \left(1 - \frac{2(y-y_j)^2}{(\frac{h_j}{\gamma_y})^2}\right) & \text{if } 0 \le |y - y_j| \le \frac{h_j}{2\gamma_y} \\ \gamma_y \times \frac{2(|y-y_j| - \frac{h_j}{\gamma_y})^2}{(\frac{h_j}{\gamma_y})^2} & \text{if } \frac{h_j}{2\gamma_y} < |y - y_j| \le \frac{h_j}{\gamma_y} \end{cases} \tag{11}$$

are the shape factors along the x and y directions, respectively, and γ_x and γ_y are two parameters used to control the spreading of the bell-shaped function $\widetilde{P_j}(x, y)$. Smaller values of γ_x and γ_y will cause the bell shaped function to become wider, which can make the spreading of highly clustered modules faster. However, if γ_x and γ_y are too small, then the approximation to the real area density function $P_j(x, y)$ will be very inaccurate. Hence, tradeoff values are chosen for these two parameters in our implementation of the algorithm. The functions $\widetilde{P_j^X}(x)$ and $\widetilde{P_j^Y}(y)$, as shown in equation (10), (11), and Fig. 2(b), have the attractive property of being normalized. Specifically, the integral of $\widetilde{P_j^X}(x)$ with respect to x over the entire real axis $\Re$ gives w_j while the integral of $\widetilde{P_j^Y}(y)$ with respect to y over $\Re$ gives $h_j = \frac{A_j}{w_j}$. Hence, the total contribution of module j to the overall area density distribution is still A_j although the smooth bell-shaped function $\widetilde{P_j}(x, y)$ is used to approximate the original rectangular-shaped density function $P_j(x, y)$.

The optimization problem (2) is solved using the conjugate gradient method [19]. There are numerous descriptions of this method in the literature and interested readers are referred to [19] for details. We emphasize here that in our implementation, the conjugate gradient algorithm is executed multiple times. Each run is terminated if the improvement between consecutive iterations becomes insignificant or a pre-determined maximum number of iterations is reached. The final solution of each run is used as the starting point of the new run. The original value of the parameter α is chosen to be small such that a higher weight is assigned to the wire length term *WL* to ensure the quality of the solution. Then, in successive runs, the value of α is increased, while the value of β is decreased. This makes the module distribution more uniform, and at the same time, allows the exploration of the optimum to go closer to the boundary of the feasible region.

C. Overlap reduction and final legalization

After the rough floorplanning stage, the modules are relatively uniformly distributed across the fixed die. However, some overlaps between modules are often present after this stage. Although increasing the density of the monitoring-point grid in the rough floorplanning stage is beneficial to reducing the overlap, it will invariably increase the computational cost. In addition, the overlap can rarely be removed completely in the rough floorplanning stage even if a rather dense grid of monitoring points is used. Hence, instead of using a very dense grid in rough floorplanning and sacrificing the runtime, we add an explicit overlap reduction and final legalization stage to the algorithm such that a legal floorplan within the given fixed die is obtained.

Due to the fact that the modules in the floorplanning stage can have significant difference in both the widths and heights and the allowed white space with respect to the die area is relatively small, the simple greedy algorithm used in [17] for legalizing the placement of standard cells will usually not be able to result in a legal floorplan within the fixed die. In our work, we choose to first reduce the overlap between modules through solving the following optimization problem

$$\text{minimize} \quad WL + \eta \cdot P_O + \beta \cdot B \tag{12}$$

where the definitions of *WL* and B are the same as those shown in section II B. The term P_O in (12) is used to represent the penalty due to the overlaps between modules and the overlap between a module and the outside of the fixed die. Hence, P_O can be written as

$$P_O = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} S_{ij}^M + \sum_{i=1}^{n} (A_i - S_i^D) \tag{13}$$

where S_{ij}^M is the overlap area between modules i and j, S_i^D is the overlap area between module i and the fixed die, and $A_i - S_i^D$ is the overlap area between module i and the *outside* of the fixed die. Note that both S_{ij}^M and S_i^D can be considered as the overlap area between two rectangles. Hence, we discuss below in detail how to quickly find this overlap area given the center coordinates and dimensions of the two rectangles, and how to use smooth functions to approximate the overlap area.

In Fig. 3, we show two overlapping rectangles, located at (x_i, y_i) and (x_j, y_j), and with dimensions $w_i \times h_i$ and $w_j \times h_j$, respectively. The overlap area can be calculated as

$$S = \Delta x \cdot \Delta y \tag{14}$$

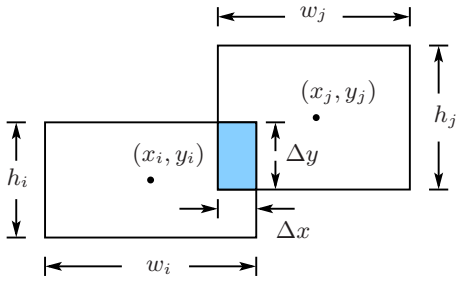


Fig. 3: Overlap between two rectangles.

where

$$\Delta x = (\min\{x_i + \frac{1}{2}w_i, x_j + \frac{1}{2}w_j\} - \max\{x_i - \frac{1}{2}w_i, x_j - \frac{1}{2}w_j\}) \times U(\min\{x_i + \frac{1}{2}w_i, x_j + \frac{1}{2}w_j\} - \max\{x_i - \frac{1}{2}w_i, x_j - \frac{1}{2}w_j\}) \quad (15)$$

and

$$\Delta y = (\min\{y_i + \frac{1}{2}h_i, y_j + \frac{1}{2}h_j\} - \max\{y_i - \frac{1}{2}h_i, y_j - \frac{1}{2}h_j\}) \times U(\min\{y_i + \frac{1}{2}h_i, y_j + \frac{1}{2}h_j\} - \max\{y_i - \frac{1}{2}h_i, y_j - \frac{1}{2}h_j\}) \quad (16)$$

Here, $U(x)$ is the unit step function, i.e.,

$$U(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (17)$$

Equation (14)-(17) can be used to calculate the overlap area between two rectangles exactly. However, none of the functions min{}, max{}, or $U(x)$ is smooth, and this will again create problems for the gradient-based optimization algorithm. These non-smooth functions are replaced by the following smooth approximating functions in our implementation of the floorplanner.

$$\min\{x, y\} \approx \frac{x \cdot e^{k(y-x)} + y \cdot e^{k(x-y)}}{e^{k(x-y)} + e^{k(y-x)}} \quad (18)$$

$$\max\{x, y\} \approx \frac{x \cdot e^{k(x-y)} + y \cdot e^{k(y-x)}}{e^{k(x-y)} + e^{k(y-x)}} \quad (19)$$

$$U(x) \approx \frac{1}{2}(1 + \tanh(k'x)) \quad (20)$$

where k and k' are parameters used to control the accuracy of the approximating functions. Larger values of k and k' will increase the accuracy of the approximation, but at the same time reduce the smoothness of the approximating functions, which may increase the difficulty of the optimization process. These two parameters are tuned in our implementation to achieve the best balance between the runtime of the algorithm and the quality of the resulting floorplans. The optimization problem (12) is again solved using the conjugate gradient method.

After solving the optimization problem (12), the resultant floorplan will contain little, if any, overlap between modules. We then send this improved floorplan to the $pl2sp()$ function in Parquet-4 such that a sequence pair corresponding to a floorplan without any overlap between modules is obtained. The core functionality of the $pl2sp()$ routine is to legalize a floorplan without performing module sizing or switching the order between modules. The reason why we cannot call $pl2sp()$ directly after the rough floorplanning stage is that the overlap between modules is not small enough at that time, and the simple legalization procedure used in $pl2sp()$ will generally not be able to fit the floorplan into the fixed-die.

D. Time complexity analysis

The runtimes for solving the unconstrained optimization problems (2) and (12) are determined primarily by the time required to calculate the objective functions and their gradients. Let n, N, N_2, and M be the total number of modules, the total number of nets in the original netlist, the total number of decomposed two-pin nets, and the total number of monitoring points in the grid superimposed on the die area and its surrounding region, respectively. The time complexity of calculating the barrier term B and its gradient is $O(n)$, and the time complexity of calculating the wire length term *WL* and its gradient is $O(N_2)$. However, under the reasonable assumption that the fanout of each net in the original netlist is upper-bounded by a constant, we obtain $O(N_2) = O(N)$.

From (6) and (7), it may seem that the time complexity of calculating P_D is $O(nM)$. However, a better analysis will show that the actual complexity is only $O(M)$. This is because to calculate P_D, we use a two dimensional array to store the D_i's corresponding to the monitoring points. Initially, all of the D_i's are set to 0. Then we go through the n modules, and for each module, we *only* update the D_i's corresponding to the monitoring points covered by this module. Since the monitoring points form a regular grid, it takes constant time to find the range of the array indices corresponding to the points covered by each module, and each updating of a D_i also takes constant time. Hence, if M' represents the total count of the monitoring points covered by all of the modules (note that if a point is covered by multiple modules, it should be counted multiple times.), then the total cost of calculating all of the D_i's is $O(M')$. Finally, we sum up all of the D_i's to obtain P_D, which takes $O(M)$ time. Since $M' = O(M)$, the total cost of calculating P_D becomes $O(M') + O(M) = O(M)$. Similarly, we can show that the time complexity of calculating the gradient of P_D is also $O(M)$.

To calculate the P_O term in (12) using expression (13), the apparent time complexity is $O(n^2)$, because of the double summation involved. However, this cost can be reduced significantly by observing that two modules i and j that are separated far apart from each other after the rough floorplanning stage will have no interaction in the overlap removal stage that follows. Hence, the S_{ij}^M term corresponding to these two modules can be dropped from the double summation in (13). Our strategy of calculating the P_O term efficiently is to associate with each module an interaction range box as shown in Fig. 4, and the S_{ij}^M term enters the double summation in (13) only if the interaction range boxes associated with modules i and j overlap with each other after the rough floorplanning stage. There is no unique way of determining the size and shape of each of the n interaction range boxes. A good heuristic is to associate with each module i a square-shaped interaction range box with the same center coordinate as the module itself and a side length of $2\sqrt{A_i R_i}$. An interaction list is established for each module after the rough floorplanning stage to store the indices of the modules that have interactions with it. The time it takes to build all of the lists is $O(n \times \log(n) + K)$ using the interval tree and range tree [20], where K is the total number of the pairs of modules that have interactions with each other. However, since the lists only need to be built once, and then they can be used many times in solving the optimization problem (12), the amortized cost of this step of the algorithm can be practically ignored, considering the problem sizes encountered in the floorplanning stage of the design, i.e., a few hundred to a few thousand modules. After the lists are established, each calculation of P_O and its gradient is reduced from $O(n^2)$ to approximately $O(n)$ time.

From the above analysis, we see that the calculation of the objective function and its gradient in the optimization problem (2) has a time complexity of $O(N + n + M)$, and the corresponding cost for the optimization problem (12) is $O(N + n)$ after the interaction lists are established. The building up of the interaction lists takes $O(n \times \log(n) + K)$ time but has an extremely small amortized cost that can be ignored in practice. We emphasize here that because the operations involved in the calculations of *WL*, B, and their gradients are all very simple, the actual costs of solving the problems (2) and (12) are dominated by the calculations of P_D, P_O, and their gradients.

Finally, the $pl2sp()$ function from Parquet-4 that we use to obtain the final overlap-free floorplan has a time complexity of $O(n^3)$, where n is the total number of modules. However, in practice, we find that the runtime associated with this function call is negligibly small compared with that of solving the optimization problems (2) and (12). This is because all of the operations involved in the $pl2sp()$ function are very

Die Aspect Ratio	1:1		2:1		3:1		4:1	
	Analytical	Parquet-4	Analytical	Parquet-4	Analytical	Parquet-4	Analytical	Parquet-4
ami33	10/10	4/10	9/10	2/10	10/10	1/10	10/10	0/10
ami49	10/10	5/10	10/10	2/10	10/10	1/10	9/10	0/10
n100	10/10	2/10	10/10	1/10	10/10	0/10	10/10	0/10
n200	9/10	0/10	10/10	0/10	9/10	0/10	9/10	0/10
n300	10/10	0/10	10/10	0/10	10/10	0/10	9/10	0/10

(a)

Die Aspect Ratio	1:1		2:1		3:1		4:1	
	Analytical	Parquet-4	Analytical	Parquet-4	Analytical	Parquet-4	Analytical	Parquet-4
ami33	10/10	5/10	10/10	6/10	10/10	2/10	10/10	2/10
ami49	10/10	8/10	10/10	6/10	10/10	4/10	9/10	1/10
n100	10/10	4/10	10/10	2/10	10/10	2/10	10/10	1/10
n200	10/10	7/10	10/10	3/10	10/10	1/10	9/10	2/10
n300	10/10	1/10	10/10	2/10	10/10	1/10	10/10	1/10

(b)

TABLE I: Success rate of the floorplanning algorithms under (a) a 10% white space constraint and (b) a 15% white space constraint.

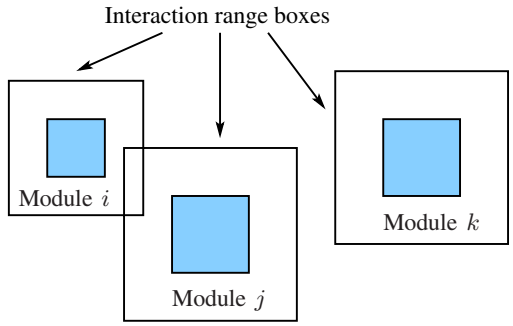


Fig. 4: Interaction range boxes of modules. The shaded areas represent modules. S_{ij}^M is included in the double summation in (13) because the corresponding interaction range boxes overlap with each other. S_{ik}^M and S_{jk}^M are excluded from the double summation in (13) because the corresponding interaction range boxes do not overlap.

simple, and n is generally much smaller than N and M, which are the total numbers of nets and monitoring points, respectively.

III. Experimental Results

Our algorithm is implemented in C++, and the experiments are performed on a desktop with a 3.2GHz Intel(R) Pentium(R)-4 CPU running the Red Hat Linux 8.0 operating system.

As stated previously, this work has focused on the problem of floorplanning with soft modules. The maximum allowed aspect ratio of each module is assumed to be 3 in our experiments. We have tested our algorithm on the two largest MCNC benchmarks ami33 and ami49 and the three largest GSRC benchmarks n100, n200 and n300, and the aspect ratio of the fixed die ranged from 1:1 to 4:1. The experimental results have been compared with those obtained from Parquet-4, which we see as a representative of the current state-of-the-art fixed-die floorplanners. Note that Parquet-4 is specifically tuned for the efficient floorplanning of hard modules, although it can also deal with soft modules [21]. The reason why we choose Parquet-4 is that, to the best of our knowledge, Parquet is the only free floorplanning package online that can handle both fixed die and soft modules, and Parquet-4 is the newest release in the Parquet series.

Tables I (a) and (b) show the comparison results of the success rate between our algorithm and Parquet-4 for the 10% and 15% white space constraints, respectively. To obtain each data value in the two tables, the corresponding floorplanning program was executed 10 times, the number of the resulting floorplans that could fit into the fixed die was counted, and the success rate was calculated. We see clearly from the tables that our analytical floorplanning algorithm can achieve above a 90% success rate for both the 10% and 15% white space constraints, and the success rate improves as more white space is allowed in the floorplan[2]. As a comparison, Parquet-4 achieves a maximum of 80% success rate and it fails most of the tests for the GSRC benchmarks when the white space constraint is set to 10%. In Fig. 5, we show some examples of the final floorplans of the n100 benchmark obtained by our algorithm under a 10% white space constraint. The outer rectangles represent the outlines of the fixed dies. Each floorplan is shifted towards the lower-left corner of its corresponding fixed die for the purpose of easy comparison. Due to the space limit, the floorplan examples of other benchmarks are omitted here.

In Fig. 6, we show the average runtimes of both our algorithm and Parquet-4 with respect to the number of modules in the floorplan. Parquet-4 performs better for the small MCNC benchmarks but its runtime increases rapidly as the number of modules increases. On the contrary, the runtime of our algorithm increases at a much slower rate and it beats Parquet-4 by about 2X for n200 and 4X for n300. Note that Fig. 6 only compares the runtime of a single execution of the algorithms. The evaluation of the real efficiency of each algorithm should also be based on the success rate because any time spent on the unsuccessful runs is wasted. Considering the much higher success rate of the analytical-based approach, we can see that our algorithm can achieve an order of magnitude effective improvement in runtime, as compared with Parquet-4, for the large GSRC benchmarks.

In Table II, we compare the average total wire length obtained from our analytical approach and that from Parquet-4 for the successful runs. Because Parquet-4 fails to find a legal floorplan for n200 and n300 when the white space constraint is set to 10%, we only compare the two algorithms for the case of 15% white space constraint. We can see from the table that our algorithm achieves better wire length than Parquet-4 and the average improvement is about 12%.

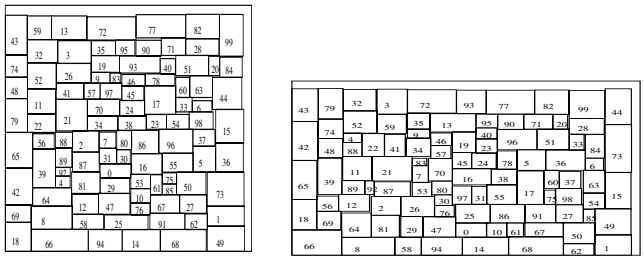
IV. Conclusions

In this paper, we presented a soft-module floorplanning algorithm based on an analytical approach. The algorithm is divided into two stages, i.e., rough floorplanning , followed by overlap reduction and final legalization. In the rough floorplanning stage, an optimization problem is solved where the objective function is a linear combination of the total wire length and the area distribution density of modules. In the overlap reduction and final legalization stage, we first solve an optimization problem to minimize the linear combination of the total wire length and the overlap area, then we call the $pl2sp()$ function in Parquet-4 to obtain a sequence pair corresponding to a floorplan without any overlap between modules. Experimental results on the MCNC and GSRC benchmarks show that our algorithm can achieve

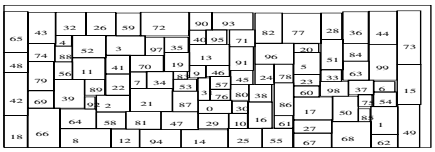
[2]For our algorithm, the initial values of the optimization variables, i.e., x_i, y_i, and w_i are chosen randomly. Hence, due to the inherent non-convexity of the problem, the obtained final floorplans also differ from run to run of the algorithm. To take this effect into consideration, we use the success rate, the average runtime, and the average total wire length to characterize the performance of our algorithm.

Die Aspect Ratio	1:1			2:1			3:1			4:1		
	Analytical	Parquet-4	Improve	Analytical	Parquet-4	Improve	Analytical	Parquet-4	Improve	Analytical	Parquet-4	Improve
ami33	74072	82149	9.8%	75168	79131	5.0%	75180	91721	18.0%	79529	101274	21.5%
ami49	799239	928597	13.9%	829888	942117	11.9%	880387	1092771	19.4%	939049	1003220	6.4%
n100	291628	342103	14.8%	290158	351542	17.5%	298894	351338	14.9%	313060	392118	20.2%
n200	572145	630014	9.2%	565927	645219	12.3%	583282	639803	8.8%	608074	685057	11.2%
n300	702822	770354	8.8%	722527	780406	7.4%	793771	838600	5.3%	858346	872501	1.6%

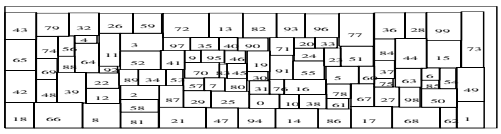
TABLE II: Average total wire length (HPWL) of the floorplans obtained using the analytical approach and Parquet-4 under a 15% white space constraint.



(a) (b)



(c)



(d)

Fig. 5: The floorplans of the n100 benchmark under a 10% white space constraint with the aspect ratio of the fixed die set to (a) 1:1, (b) 2:1, (c) 3:1, and (d) 4:1. Each floorplan is shifted towards the lower-left corner of its corresponding fixed die for the purpose of easy comparison.

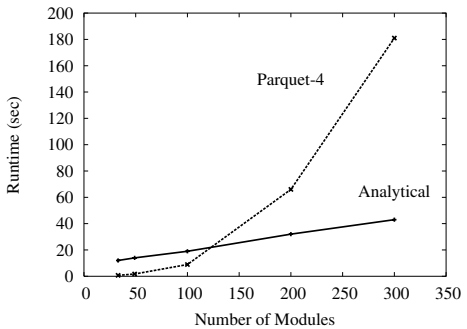


Fig. 6: Comparison of the runtimes of our algorithm and Parquet-4.

above a 90% success rate for a white space constraint of as low as 10%, while the average improvement in wire length is about 12% compared with Parquet-4. In addition, our algorithm can achieve an order of magnitude effective speedup compared with Parquet-4 for the large GSRC benchmarks.

REFERENCES

[1] A. B. Kahng, "Classical Floorplanning Harmful?" *Proceedings of the ACM International Symposium on Physical Design*, pp. 207-213, Apr. 2000.

[2] S. N. Adya and I. L. Markov, "Fixed-outline Floorplanning: Enabling Hierarchical Design," *IEEE Transactions on VLSI Systems*, vol. 11, no. 6, pp. 1120-1135, Dec. 2003.

[3] C. T. Lin, D. S. Chen, and Y. W. Wang, "Robust Fixed-outline Floorplanning through Evolutionary Search," *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference*, pp. 42-44, Jan. 2004.

[4] T. C. Chen and Y. W. Chang, "Modern Floorplanning Based on Fast Simulated Annealing," *Proceedings of the ACM International Symposium on Physical Design*, pp. 104-112, Apr. 2005.

[5] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov, "Unification of Partitioning, Placement, and Floorplanning," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 550-557, Nov. 2004.

[6] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-Packing-Based Module Placement," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 472-479, Nov. 1995.

[7] S. Nakatake, K. Fujiyoshi, H. Mirata, and Y. Kajitani, "Module Packing Based on the BSG-Structure and IC Layout Applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 6, pp. 519-530, Jun. 1998.

[8] P Guo, C. Cheng, and T. Yoshimura, "An O-Tree Representation of Non-Slicing Floorplan and Its Applications," *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 268-273, Jun. 1999.

[9] X. Hong, S. Dong, G. Huang, Y. Ma, Y. Cai, C. Cheng, and J. Gu, "A Non-Slicing Floorplanning Algorithm Using Corner Block List Topological Representation," *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 833-836, Dec. 2000.

[10] Y. C. Chang, Y. W. Chang, G. M. Wu, and S. W. Wu, "B*-Tree: A New Representation for Non-Slicing Floorplans," *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 458-463, Jun. 2000.

[11] J. M. Lin and Y. W. Chang, "TCG: A Transitive Closure Graph-Based Representation for Non-Slicing Floorplans," *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 764-760, Jun. 2001.

[12] A. Ranjan, K. Bazargan, S. Ogrenci, and M. Sarrafzadeh, "Fast Floorplanning for Effective Prediction and Construction," *IEEE Transactions on Very Large Scale Integration (VLSI)*, vol. 9, no. 2, pp. 341-351, Apr. 2001.

[13] T. C. Wang and D. F. Wong, "Optimal Floorplan Area Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 8, pp. 992-1001, Aug. 1992.

[14] F. Y. Young, C. C. N. Chu, W. S. Luk, and Y. C. Wong, "Handling Soft Modules in General Non-Slicing Floorplan using Lagrangian Relaxation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 5, pp. 687-692, May 2001.

[15] J. Cong, M. Romesis, and J. R. Shinnerl, "Fast Floorplanning by Look-Ahead Enabled Recursive Bipartitioning," *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference*, pp. 1119-1122, Jan. 2005.

[16] W. Naylor *et al.*, "Non-Linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer," *US Patent 6301693*, Oct. 2001.

[17] A. B. Kahng and Q. Wang, "Implementation and Extensibility of an Analytical Placer," *Proceedings of the ACM International Symposium on Physical Design*, pp. 18-25, Apr. 2004.

[18] F. S. Hillier and G. J. Lieberman, "Introduction to Operations Research," McGraw-Hill, New York, NY, 1995.

[19] W. H. Press *et al.*, "Numerical Recipes in C++," Cambridge University Press, Cambridge, UK, 2002.

[20] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, "Computational Geometry," Springer-Verlag, Berlin, Germany, 2000.

[21] I. Markov, Private communication, 2005.