Floorplanning Method Based on Liner Programming

Changhong Zhao, Jian Chen, Xiaofang Zhou, Ming’e Jing, Dian Zhou
ASIC & System State Key Lab.
Fudan University
Rd. Zhangheng No.825 Shanghai P.R.China 201203

Abstract: This paper analyses the floorplanning methods based on linear programming and presents a linear replacement of the non-linear objective function. It also presents the sub-section linearization method to replace the original nonlinear items in the constraint inequalities. Compared with former floorplanning methods based on linear programming, the solutions of the method in this paper always lie in the feasible region of the original floorplanning problem. Experimental results show that the method in this is competitive.

Key words: floorplanning, linear programming, VLSI

*Parameter:

- \( b_i \): the \( i \)th module
- \( x_i \): the horizontal coordinate of the bottom-left corner of \( b_i \)
- \( y_i \): the vertical coordinate of the bottom-left corner of \( b_i \)
- \( h_i \): the height of \( b_i \)
- \( w_i \): the width of \( b_i \)
- \( a_i \): the area of \( b_i \)
- \( \tau_i = h_i / w_i \)
- \( \tau_i, \tau_j \): \( \tau_i \leq \tau_j \leq \tau_i \)
- \( \tau_c = h_i / w_c \)
- \( \tau_c, \tau_c \): \( \tau_c \leq \tau_c \leq \tau_c \)

0 INTRODUCTION

Floorplanning of VLSI is a key research point and a lot of excellent work has been done in recent years. Nowadays most research work in floorplanning is focused on the aspects such as interconnect driven floorplanning[8][9], bus driven floorplanning[10] and so on. But the research of the traditional floorplanning is never interrupted. The research of this paper belongs to the traditional floorplanning and the floorplanning based on linear programming method is presented to deal with the adjustment of soft modules.

Floorplanning of VLSI is a NP hard problem
and heuristic algorithms are usually adopted. The method in this paper is divided into two steps. The first step is to adjust the position of all modules with their widths and heights unchanged by means of simulated annealing and sequence pair representation. After the first step, the relative positions of all modules are determined. In the second step, the linear programming is used to adjust the soft modules to further improve the quality of floorplanning with the relative position of all modules unchanged.

In the second step, the total wire length will change little because of the fixed relative position of all modules. Therefore in this paper, the objective function of the second step only includes the chip area.

1 Preliminaries

1.1 Problem Formulation of the Traditional Floorplanning

Let $B = \{b_1, b_2, ..., b_N\}$ be a set of $N$ rectangular modules. $b_i \in B$ is associated with a three tuple $\{h_i, w_i, a_i\}$ and $\tau_i$. A floorplanning is an assignment of rectangular modules with the coordinates of their bottom-left corners being assigned to $(x_i, y_i)$’s so that there are no overlaps between any two modules. The objective of placement/floorplanning is to minimize a specified cost metric such as the area of the chip, the total wire length or the combination of both chip area and total wire length.

1.2 Sequence Pair[1]

In this paper, sequence pair is adopted in the first step. A sequence pair is a pair of combinations of module names. It represents the relationship between any two modules $b_i$ and $b_j$ as follows:

- If $b_i$ is ahead of $b_j$ in both sequences, $b_i$ is to the left of $b_j$. The corresponding sequence pair is $(\cdots b_j \cdots, \cdots b_i \cdots)$.
- If $b_i$ is ahead of $b_j$ in the first sequence while behind $b_j$ in the second sequence, module $b_i$ is above module $b_j$. The corresponding sequence pair is $(\cdots b_i \cdots, \cdots b_j \cdots)$.

2 Problem formulation

The mathematical programming model of the second step in this paper is given below.

P1:

Minimize $f_i = h_i w_c$  

Subject to:

\[
\begin{align*}
\tau_i \leq h_i / w_i \leq \tau_i \quad & \text{or} \quad \tau_i w_i \leq h_i \leq \tau_i w_i \quad (2) \\
x_i + w_i \leq x_j \quad (b_i \text{ is to the left of } b_j) \quad (3) \\
y_i + h_i \leq y_j \quad (b_i \text{ is above } b_j) \quad (4) \\
w_i \cdot h_i \geq a_i \quad (\forall i) \quad (5) \\
x_i + w_i \leq w_c \quad (\forall i) \quad (6) \\
y_i + h_i \leq h_c \quad (\forall i) \quad (7) \\
x_i \geq 0 \quad y_i \geq 0 \quad w_i \geq 0 \quad h_i \geq 0 \quad (8) \\
\tau_c w_c \leq h_i \leq \tau_i w_c \quad (9)
\end{align*}
\]
In P1, (1) and (5) are non-linear items and further analysis proves that (1) is a non-convex function. Therefore (1) and (5) should be approximated in order to solve P1 efficiently.

Figure 1. $f_1 = h_c w_c$

3 Methods based on linear programming

If P1 can be approximated to a linear programming, it can be solved efficiently. The process to transform P1 to a linear programming is the process to linearize (1) and (5). The linearization of (1) will be discussed in subsection 3.1 and the linearization of (5) will be discussed in subsection 3.2. In subsection 3.3, the linear programming model will be given.

3.1 Replacement of (1)

First the replacement of (1) will be discussed.

Let $f_2 = \sqrt{h_c w_c}$ (10)

It is clear that the minimization of (1) is equivalent to that of (10). Further analysis proves that (10) is a convex function (as figure 2).

Figure 2. $f_2 = \sqrt{h_c w_c}$

It is clear that $h_c, w_c > 0$, so

$$\frac{h_c + w_c}{2} \geq \sqrt{h_c w_c} \quad (11)$$

Therefore the perimeter of the chip is used to replace (1) in [3] and the objective of P1 is changed as:

Minimize $f_2 = h_c + w_c \quad (12)$

In [4], Taylor expansion is used to linearize the objective function of P1. The corresponding function is given below.

$$\min f(w_c, h_c) = w_c^0 h_c^0 + h_c^0 (w_c - w_c^0) \approx w_c^0 h_c^0 + h_c^0 (w_c - w_c^0) + w_c^0 (h - h_c^0) \quad (13)$$

(13) is equivalent to (14)

Minimize $f^*(w_c, h_c) = h_c^0 w_c + w_c^0 h_c \quad (14)$

In this paper, a new replacement function of (1) is presented as follows.

From (11), when $\tau_c = 1 \quad (h_c = w_c)$

$$0.5 h_c + 0.5 w_c = \sqrt{h_c w_c} \quad (15)$$

When $\tau_c \neq 1$, it is assumed that

$$\delta_1 h_c + \delta_2 w_c = \sqrt{h_c w_c} \quad (16)$$

In order to reduce the error of the estimation of the chip area, $\delta_1$ and $\delta_2$ should be continuous. Let

$$\delta_1 + \delta_2 = 1 \quad (17)$$

From (16) and (17)

$$\delta_1 = 1/\sqrt{\tau_c} + [\square] \quad (18)$$

$$\delta_2 = \sqrt{\tau_c}/(1 + \sqrt{\tau_c}) \quad (19)$$
During the process of floorplanning, $\tau_c$ is usually uncertain. Because the change of $\tau_c$ during the process of adjusting the soft module is slight, $\tau_c^{(0)}$, which is obtained after the first step, is used to approximate $\tau_c$. Also we can use $(\tau_c + \tau_c)/2$ to approximate $\tau_c$. In this paper we use $\tau_c^{(0)}$ to approximate $\tau_c$. After $\tau_c$ is approximated, $\delta_1, \delta_2$ is calculated and the final objective function is

$$
\delta_1 h_c + \delta_2 w_c
$$

(20)

In this paper, (20) is used to be the replacement of (1).

3.2 Replacement of (5)

The replacement of (5) in [3] is given below.

$$
2(w_i + h_i) \geq 3 \sqrt{a_i + \lambda_i s_i}
$$

(21)

Here $s_i = \max(w_i^b, h_i^b)$ and $\lambda_i$ is an adjustment parameter. Different modules have different $\lambda_i$’s. The floorplanning result will be worse if $\lambda_i$ is too big whereas the solution will be infeasible if $\lambda_i$ is too small. Therefore, (21) is not a good replacement inequality of (5).

In [4], Taylor expansion is used to replace (5). The replacement inequality is:

$$
h_i \geq h_i^{(j)} - \frac{h_i^{(j)}}{w_i^{(j)}}(w_i - w_i^{(j)})
$$

(22)

In order to approximate the area constraints accurately, the approximation is done at several $w_i^{(j)}$ values. The approximation of (1) in [4] is shown below:

From figure 4, it is clear that the new feasible region is enlarged after the approximation. In fact the solution in [4] is always in the infeasible region of P1 because the solutions of a linear programming always lies on the edge of its feasible region.

To overcome the defects of the replacement inequality of (5) in [3] and [4], a subsection linearization method is presented in this paper. Details of this method are given below.

1. Select several special point $(w_i^{(0)}, h_i^{(0)})$, $(w_i^{(1)}, h_i^{(1)})$, ...... $(w_i^{(m)}, h_i^{(m)})$, $(w_i^{(0)} < w_i^{(1)} < ... < w_i^{(m)})$ on the curve of
\[ w_i \cdot h_i = a_i. \] Let \( h_i^{(0)}/w_i^{(0)} = \tau_i \) and \( h_i^{(m)}/w_i^{(m)} = \tau_i. \)

2. Connect \( ((w_i^{(0)}, h_i^{(0)}), (w_i^{(1)}, h_i^{(1)})) \) \( ((w_i^{(1)}, h_i^{(1)}), (w_i^{(2)}, h_i^{(2)})) \) \( \ldots \) \( ((w_i^{(m-1)}, h_i^{(m-1)}), (w_i^{(m)}, h_i^{(m)})) \) to form a new curve.

3. Let new curve replace \( w_i \cdot h_i = a_i. \)

The feasible region in this paper lies in the upper region of the new curve.

P2 is a linear programming and the number of variables is \( 4N + 2 \).

As in [3] and [5], redundant items corresponding to (3) \( (4) \) (6) \( (7) \) should be deleted before linear programming is used.

### 4 Experimental results

It is assumed that \( 0.1 \leq \tau_i \leq 10 \) and \( 0.9 \leq \tau_i \leq 1.1 \). In the first step, simulated annealing algorithm based on sequence pair is used to determine the relative position of all modules. In the second step, P2 is adopted to adjust the soft modules. We select 19 points \( ((q/(10-j), q/(10-j)), (q/(10-j), q/(10-j))) \) \( (j = 0, 1, 2, \ldots, 9) \) on \( w_i \cdot h_i = a_i \) in the subsection linearization process.

In the experiment, the method is tested on five MCNC benchmark examples. It is also tested on a big benchmark test196 which is generated from four ami49s. The first step is implemented in c++ language and the liner programming is implemented in matlab language in a 3.8G PC with 1G bytes memory. Linear interior point solver is adopted in the linear programming process. The experimental results are shown in table 1 and figure 6-7.

Table 2 and table 3 are the results in [3] and [4]. Compared with table 2-3, the chip area of some benchmarks in table 1 are bigger than that of table 2-3. This is because that the area of a module may be decreased in table 2-3. An obvious example is that the chip area usage of apt in table 3 exceeds 100%.

By our method, the area of a module after linear programming may be bigger than the original one when (5) is replaced by (23), which guarantee the feasibility of the solutions.
5 Conclusion

This paper analyses the floorplanning method based on linear programming. Compared with the method in [3-4], the solution of the method in this paper always lies in the feasible region of the original programming problem P1, even though some of the floorplanning results are worse than that of [3-4] in appearance. In [3-4] the area of each module may be reduced after their process, while these areas should not be reduced during the floorplanning process. Therefore, the method of this paper is competitive.

Reference

Table 1: Experimental results in this paper

<table>
<thead>
<tr>
<th>benchmark</th>
<th>Input chip area</th>
<th>Output chip area</th>
<th>Area usage</th>
<th>The error of the sum area of all modules</th>
<th>Run time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ami33</td>
<td>1.1936</td>
<td>1.1675</td>
<td>99.05%</td>
<td>0.9%</td>
<td>0.4370</td>
</tr>
<tr>
<td>Ami49</td>
<td>37.27</td>
<td>35.987</td>
<td>98.5%</td>
<td>1%</td>
<td>0.8750</td>
</tr>
<tr>
<td>Apte</td>
<td>47.08</td>
<td>47.29</td>
<td>98.46%</td>
<td>0.25%</td>
<td>0.094</td>
</tr>
<tr>
<td>Hp</td>
<td>9.21</td>
<td>9.00</td>
<td>98.06%</td>
<td>1%</td>
<td>0.11</td>
</tr>
<tr>
<td>Xerox</td>
<td>19.87</td>
<td>19.59</td>
<td>98.78%</td>
<td>1.2%</td>
<td>0.1</td>
</tr>
<tr>
<td>Test196</td>
<td>156.46</td>
<td>144.32</td>
<td>98.24%</td>
<td>1.17%</td>
<td>141.25</td>
</tr>
</tbody>
</table>

Table 2: Experimental results in [3]

<table>
<thead>
<tr>
<th></th>
<th>Ami33</th>
<th>Ami49</th>
<th>Apte</th>
<th>Hp</th>
<th>Xerox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>1.1578</td>
<td>35.5206</td>
<td>46.5616</td>
<td>8.8306</td>
<td>19.3503</td>
</tr>
<tr>
<td>Run time (s)</td>
<td>16260</td>
<td>68995</td>
<td>496.5</td>
<td>1659.9</td>
<td>4872.0</td>
</tr>
</tbody>
</table>

The test in [3] was done on a 500M PC. Run time includes simulated annealing process and linear programming process. \( 0.1 \leq \tau_i \leq 10 \) \((i = 1,\ldots,N)\)

Table 3: Experimental results in [4] (The total area of the modules in a circuit is normalized to 1)

<table>
<thead>
<tr>
<th></th>
<th>Ami33</th>
<th>Ami49</th>
<th>Apte</th>
<th>Hp</th>
<th>Xerox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Area</td>
<td>1.31</td>
<td>1.21</td>
<td>1.11</td>
<td>1.25</td>
<td>1.24</td>
</tr>
<tr>
<td>Output area</td>
<td>1.03</td>
<td>1.00</td>
<td>0.96</td>
<td>1.02</td>
<td>1.00</td>
</tr>
<tr>
<td>Run time (s)</td>
<td>18s</td>
<td>34</td>
<td>4s</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

The test in [4] was done on a 233M PC with 64M bytes memory. \( 0.3333 \leq \tau_i \leq 3 \) \((i = 1,\ldots,N)\)