ABSTRACT

In this paper, we propose a new hierarchical timing-driven standard-cell placement method, in which a new cluster placement model is introduced. The proposed method is divided into three phases, clustering cells, global placement and detailed placement. After clustering cells, in the global placement phase, we determine and improve cluster placement by a simulated annealing based method. In most of previous methods, the shape of any cluster was restricted to a square. In the proposed method, however, we remove this restriction, and propose a new placement model, called the Amoeba model, in which the shape of a cluster can be a collection of connected squares. Since the flexibility of cluster placement is increased by the amoeba model, we can obtain a high quality placement satisfying timing constraints. In the detailed placement phase, we assign cells to cell rows under the non-overlapping constraint of cells with a constructive approach. Experimental results show the effectiveness of the proposed method with the new cluster placement model.

1. INTRODUCTION

With the rapid progress of semiconductor technology, standard-cell placement has become a quite difficult task, since it must handle circuits containing more than $10^6$ cells under strict timing constraints. A well-known approach to this difficult problem is to introduce clustering of cells so that the problem size is effectively reduced. In particular, when a simulated annealing based placement algorithm is used, clustering is indispensable to get a good placement in a practical computation time [7]. There have been many clustering methods proposed so far [1, 4, 8].

Any placement method combining clustering and simulated annealing typically consists of three phases, i.e., clustering cells, global placement, and detailed placement. Global placement is to determine a placement of clusters, and detailed placement is to get a final standard cell placement. In the global placement phase, the shape of any cluster is generally assumed as a square, or sometimes, a circle. However, it may not be appropriate to restrict the shape of a cluster to a square or a circle, since, in the final cell placement, cells in a cluster would not form a square or a circle. Thus, it is quite reasonable to expect that, if the shape of a cluster in global placement can be flexibly changed, a better final placement would be obtained. This is our basic idea of the proposed method.

In this paper, we propose a clustering based standard cell placement method, which consists of three phases mentioned above. In the first clustering phase, we perform a connectivity-based clustering algorithm to form clusters of cells for the given netlist. In the global placement phase, we determine and improve cluster placement by simulated annealing. We propose a new cluster placement model, called the Amoeba model. In the amoeba model, the shape of any cluster can be flexibly changed under some constraints. Since the flexibility of cluster placement is increased, it is expected that a high quality placement satisfying timing constraints is easily obtained. Finally, in the detailed placement phase, we assign cells to cell rows under the non-overlapping constraint of cells with a constructive approach considering the minimization of estimated total wire length of all nets to get a final legal cell placement. Experimental results show that the proposed placement method produced better placements, compared with the placement method, in which the shape of any cluster is restricted to a square.

The remainder of this paper is organized as follows. In Section 2, we explain the interconnect delay model, and formulate the timing-driven standard-cell placement problem. In Section 3, we propose a timing-driven standard-cell placement method based on a new cluster placement model. In Section 4, experimental results are given. Finally, in Section 5, we address some conclusions and discuss future researches.

2. PRELIMINARIES

2.1. Layout Model

The layout model we assume in this paper is the row based standard cell model. A standard cell layout consists of rectangular modules with the same height called cells. The functionality and the electrical characteristics of each predefined cell are tested, analyzed, and specified in advance. Cells are placed in rows called cell rows, and none of them should overlap each other. I/O pads are placed around
the chip area. The space between two cell rows is called a *channel*, which is used to realize interconnection between cells.

### 2.2. Delay Model

In the proposed algorithm, timing constraints are taken into account, and thus an appropriate wire delay model is required. In this paper, the *Elmore delay* model [5] is adopted as the interconnect delay model.

![Figure 1: Elmore delay of wire $e_v$.](image)

For wire $e$, let $l_e$, $w_e$, $c_e$, and $r_e$ denote its length, width, capacitance, and resistance, respectively. Further, let $e_v$ denote the wire entering node $v$ from its parent, as shown in Figure 1. We use the following model for interconnect delay of $e_v$, denoted $D_{wire}(e_v)$.

$$
\begin{align*}
    c_e &= (c_{c0} \cdot w_e + c_f) \cdot l_e \\
    r_e &= r_0 \cdot \frac{l_e}{w_e} \\
    D_{wire}(e_v) &= r_{e_v} \cdot \left( \frac{l_v}{2} + c(T_v) \right)
\end{align*}
$$

where $c_{c0}$, $c_f$, and $r_0$ are area capacitance, fringing capacitance, and resistance for unit-width, unit-length wire, respectively. $T_v$ is the subtree rooted at $v$, and $c(T_v)$ is the capacitance of directed connected subtree in $T_v$ from $v$'s root $v$. The total Elmore delay $D_{Elmore}$ from source $s_0$ to sink $t_i$ is given as follows.

$$
D_{Elmore}(s_0, t_i) = \sum_{e_v \in \text{path}(s_0, t_i)} D_{wire}(e_v) 
$$

### 2.3. Problem Formulation

Let $L = (\mathcal{M}, \mathcal{N})$ be a logic circuit, where $\mathcal{M} = \{m_1, m_2, ..., m_M\}$ is a set of cells and $\mathcal{N} = \{n_1, n_2, ..., n_N\}$ is a set of nets. Any net $n_i$ can be represented as a subset of $\mathcal{M}$.

Assume that a net $n_j$ is a set of cells, $n_j = \{m_{n_j,0}, m_{n_j,1}, ..., m_{n_j,k}\} \subset \mathcal{M}, k = |n_j| - 1$, where $m_{n_j,0}$ is the source cell and others are sink cells. For each net $n_j$, the required arrival time $T_{req_j}$ is given. Let $d_j$ denote the largest delay time from the source to a sink of a net $n_j$. Then, the timing constraint violation $V_j$ of a net $n_j$ is defined as follows.

$$
V_j = \begin{cases} 
0 & (d_j \leq T_{req_j}) \\
 1 & (d_j > T_{req_j}) 
\end{cases}
$$

### 3. THE PROPOSED ALGORITHM

#### 3.1. Overview

The proposed method is divided into three phases, i.e., clustering of cells, global placement and detailed placement. After clustering cells, in the global placement phase, cluster placement is determined and improved with a simulated annealing based algorithm. Then, in the detailed placement phase, cells are assigned to cell rows under the non-overlapping constraint of cells. In the following, we give the overview of each phase.

First, clustering of cells is performed to reduce the size of the problem so as to shorten the computation time of the simulated annealing based global placement method in the global placement phase (Fig. 2(b)).

In most of previous clustering-based placement methods, in the global placement phase, the shape of each cluster is assumed to be a square, and each cluster is placed in the chip area, allowing the overlaps among clusters. However, it may be difficult to obtain a high quality cell placement, since cells in a cluster will not necessarily form a square in the final cell placement. Making the size of a cluster small
would lead to a better placement, but it would also incur the large increase of computation time if the placement method was based on simulated annealing.

To resolve this difficulty, we propose a new cluster placement method. In the global placement phase of the proposed method, the chip area is divided into a set of global bins (Fig. 2(c)). Then, all clusters are placed on global bins by a simulated annealing based method (Fig. 2(d)), allowing the overlaps among clusters. The area of each cluster is k times as large as the area of a global bin, where k is a positive integer less than 20 (in the current implementation, we set k=10). Let \( C \) be a set of all clusters, \( C = \{c_1, c_2, ..., c_C\} \), and let \( B \) be a set of all global bins, \( B = \{b_1, b_2, ..., b_B\} \). Each cluster is placed on more than one global bin (Figure 3). Therefore, let \( A_i \) be the size of a cluster \( c_i \) and let \( A_{bin} \) be the size of a global bin, then a cluster \( c_i \) is placed on \( N_i \) global bins, where \( N_i = \lceil A_i / A_{bin} \rceil \).

We call \( N_i \) global bins composing a cluster \( c_i \) the cluster bins of \( c_i \). And for a set of global bins, \( B \), two global bins \( b_s \in B \) and \( b_t \in B \) is said to be 8-adjacent, if they share a common vertex. Moreover, if the reflexive transitive closure of the 8-adjacent relation on \( B \) is equal to the universal relation of \( B \), we say that \( B \) is 8-connected.

We formally define the amoeba cluster placement model as follows.

**Amoeba Model** For any cluster \( c \in C \), \( c \) is placed on a set of 8-connected global bins in the chip area.

3.4. Phase 2: Global Placement

As described in Section 3.1, in the global placement phase of the proposed method, the set of all clusters in \( C \) is placed in the chip area with the simulated annealing based
placement method with the amoeba model. Outline of the global placement phase is shown in Figure 4.

Initial placement of clusters is generated randomly under the condition that the shape of any cluster is a square and overlaps among clusters are permitted. We define the shape of a cluster as a rectilinear polygon by eliminating bins from the top-right corner of a square if the cluster can’t be placed in a square shape.

We give some notations to explain the detail of the proposed algorithm in the following. When a cluster \( c_i \in \mathcal{C} \) was placed with the amoeba model, a set of global bins on which cluster bins of \( c_i \) have been placed is represented as \( P(c_i) \subset \mathcal{B} \). Moreover, a set of clusters placed on global bin \( b_j \) is represented as \( Q(b_j) \), that is, \( Q(b_j) = \{ c_i \mid b_j \in P(c_i), c_i \in \mathcal{C} \} \).

The objective of determining the shape of \( c_i \) is to minimize the total delay considering the number of overlaps among clusters. Therefore, we should use the sum of delay

3.4.1. Moving clusters

In this section, we describe how to change the cluster placement in the proposed method when the cluster placement is improved with simulated annealing. There are two cases. One is to move one cluster, and the other is to interchange locations of two clusters. But, the latter can be realized as an extension of the former. Therefore, we mainly explain how to move a cluster using an example shown in Figure 5.

We assume that all clusters are placed in the chip area based on the amoeba model. First, one of them is randomly selected and the location of this cluster is changed as follows. Let this cluster be \( c_i \in \mathcal{C} \), and only \( c_i \) is shown in Figure 5(a). Then, \( c_i \) is removed from the chip area, and let \( P(c_i) = \emptyset \) (Figure 5(b)). Next, one of the global bins, \( b_s \in \mathcal{B} \), in the chip area is selected randomly. One of the cluster bins in \( c_i \) is placed on \( b_s \), and let \( P(c_i) = \{ b_s \} \) (Figure 5(c)). Finally, the shape of \( c_i \) is determined by adding \((N_i-1)\) cluster bins to \( P(c_i) \) one by one while keeping them 8-connected (Figure 5(d)∼(f)). We explain how to determine the shape of a cluster in Section 3.4.2.

Here, before changing the location of \( c_i \), if there isn’t any cluster at \( b_s \), i.e., \( Q(b_s) = \emptyset \), or \( c_i \in Q(b_s) \), then, the procedure terminates. Otherwise, the procedure to interchange locations of two clusters starts. Another cluster, denoted \( c_j \), is randomly selected from \( Q(b_s) \) except \( c_i \). Then, \( c_i \) is removed from the chip area, and let \( P(c_j) = \emptyset \). We randomly select one of the global bins on which \( c_j \) was placed before changing the location of \( c_i \). Let this global bin be \( b_t \). One of the cluster bins in \( c_j \) is placed on \( b_t \), and let \( P(c_j) = \{ b_t \} \). Similar to the case of \( c_i \), the shape of \( c_j \) is determined by adding \((N_j-1)\) cluster bins to \( P(c_j) \) one by one while keeping them 8-connected.

3.4.2. Determining the shape of a cluster

We explain how to determine the shape of a cluster when moving it. As described in Section 3.4.1, for the cluster \( c_i \) to be moved, first, one global bin is randomly selected as the destination of move of one cluster bin of \( c_i \), and then, remaining cluster bins are moved one by one so that the shape of cluster \( c_i \) is determined. The following is the procedure to determine the shape of a cluster.

**Step 1** Let \( P(c_i) = \{ b_s \} \).

**Step 2** Let \( B_8(c_i) \subset \mathcal{B} \) be a set of global bins which are 8-adjacent with \( P(c_i) \).

**Step 3** Evaluate the cost of each \( b \in B_8(c_i) \).

**Step 4** Add \( b_{min} \in B_8(c_i) \) with the minimum cost to \( P(c_i) \), that is, \( P(c_i) = P(c_i) \cup \{ b_{min} \} \).

**Step 5** Repeat Step 2∼Step 4 until \( |P(c_i)| = N_i \).
3.4.3. Cost function

A current solution in simulated annealing (SA) is perturbed by the methods presented in Sections 3.4.1 and 3.4.2. Then, a neighborhood solution is generated. We define the cost function to evaluate a neighborhood solution as follows.

\[
f_{\text{cost}} = L_{\text{wire}} + \beta N_{\text{overlap}} + \gamma T_{\text{vio}}
\]  

(6)

where \(N_{\text{overlap}}\) is the quadratic sum of the number of cluster overlaps on each global bin, defined as follows.

\[
N_{\text{overlap}} = \sum_{b \in B} (|Q(b)| - 1)^2
\]

\(L_{\text{wire}}\) is the total estimated wire length of all nets and \(T_{\text{vio}}\) is the total violation time of all nets. \(L_{\text{wire}}\) and \(T_{\text{vio}}\) are calculated by an extension of the SERT algorithm [3] described in Section 3.4.4. \(\beta\) and \(\gamma\) are user specified parameters.

3.4.4. Routing Estimation

To evaluate interconnections among clusters by Equation (6), we need to assign the pins of a net on each cluster and generate routing patterns of interconnections. We determine them by an extension of the SERT algorithm [3].

In the proposed method, we assign the pin positions for each cluster and determine global routes among clusters, simultaneously. First, we must assign the pin position of the source to the source cluster. We assign the pin position of the source to the cluster bin which is the nearest to the center of a net. The center of a net is determined by calculating the arithmetic mean of locations of all clusters connecting to the net. Next, instead of seeking terminals in each step of adding a new edge, we seek cluster bins which haven’t been added to a Steiner tree. And then, the pin of the cluster connecting to a net is assigned to a cluster bin included in the cluster. For the lack of space, the details are omitted. For details, refer to [6].

3.5. Phase 3: Detailed Placement

In the detailed placement, a final standard cell placement is determined from the global placement of clusters. The cell placement is done with the step-by-step refinement procedure as follows. This procedure consists of three steps. In the first step, cluster bins in each cluster are assigned to global bins in the chip area without any overlaps among clusters, considering the minimization of total wire length. For each cluster, the corresponding global bins in the chip area thus determined in this step are called target global bins. In the second step, each cluster is decomposed into a set of original cells, and each cell is assigned to one of target global bins, which were determined in the first step, considering the minimization of total wire length. Finally, in the third step, each cell in the global bins are assigned
to cell rows, also considering the minimization of total wire length. For the lack of space, the details are omitted. For details, refer to [6].

4. EXPERIMENTAL RESULTS

The proposed method has been implemented in C language, and run on a 450MHz Ultra SPARC-II processor. To evaluate the effectiveness of the proposed cluster placement model, we compare the proposed method with the placement method, in which all behaviors are the same as the proposed method except that the shape of any cluster is restricted to a square.

For test data, we use the ISPD98 benchmark data, which have been originally proposed as the partitioning benchmark suite [2]. We modify the data by specifying the shape of each cell so that they can be used as the benchmark of standard cell placement. Table 1 shows the characteristics of the ISPD98 circuits used in the experiments, where #Cells, #Pads, #Nets, and #Pins are the numbers of cells, pads, nets, and pins of data, respectively. Parameters used in the proposed methods, such as constants in the cost functions, the initial temperature of SA, etc., were appropriately set based on the preliminary experiments. For timing constraints, 800ps was set to all nets.

Table 1: Test data

<table>
<thead>
<tr>
<th>circuit</th>
<th>#Cells</th>
<th>#Pads</th>
<th>#Nets</th>
<th>#Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm01</td>
<td>12506</td>
<td>246</td>
<td>14111</td>
<td>50566</td>
</tr>
<tr>
<td>ibm05</td>
<td>28146</td>
<td>1201</td>
<td>28446</td>
<td>126308</td>
</tr>
<tr>
<td>ibm07</td>
<td>45639</td>
<td>287</td>
<td>48117</td>
<td>175639</td>
</tr>
</tbody>
</table>

Table 2: Experimental results

<table>
<thead>
<tr>
<th>circuit</th>
<th>model</th>
<th>$L_{wire}$</th>
<th>$T_{vio}$</th>
<th>$N_{vio}$</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm01</td>
<td>New</td>
<td>69</td>
<td>77</td>
<td>437</td>
<td>90772</td>
</tr>
<tr>
<td></td>
<td>Old</td>
<td>65</td>
<td>126</td>
<td>631</td>
<td>65494</td>
</tr>
<tr>
<td>ibm05</td>
<td>New</td>
<td>209</td>
<td>3865</td>
<td>109088</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Old</td>
<td>227</td>
<td>3865</td>
<td>109088</td>
<td></td>
</tr>
<tr>
<td>ibm07</td>
<td>New</td>
<td>441</td>
<td>6820</td>
<td>8922</td>
<td>38111</td>
</tr>
<tr>
<td></td>
<td>Old</td>
<td>458</td>
<td>4800</td>
<td>10596</td>
<td>41787</td>
</tr>
</tbody>
</table>

Table 2 shows experimental results. In this table, New and Old mean the proposed placement method with the new amoeba cluster placement model and the placement method with the existing non-amoeba cluster placement model. $L_{wire}$ shows the total wire length of all nets [m], and $T_{vio}$ and $N_{vio}$ represent the total of timing constraint violations [ns] and the total number of unsatisfied timing constraints, respectively. CPU shows the CPU time [seconds].

From experimental results, the proposed method produces better placements than the previous method in terms of timing violation. The total wire length of the proposed method was also better than the previous one. However, there are still many timing violations remained. This might be partly due to the fact that unrealistically severe timing constraints were given to all nets. Nevertheless, further reduction of timing violation is required. Since the CPU time is large, reduction of the CPU time is also needed.

5. CONCLUSIONS

In this paper, we have proposed a clustering based, timing-driven standard-cell placement method with a new cluster placement model, called the amoeba model. In the proposed method, the shape of any cluster, which was used to be a square in the previous method, can be flexibly changed. Since the flexibility of cluster placement is increased, we can obtain a high quality placement satisfying timing constraints. Experimental results were quite promising.

There are several future works. First, in the clustering phase, some timing influenced clustering method like [8] is sought, since in the current implementation, no timing constraints are considered in cell clustering. Second, in the global placement phase, some effective mechanism to remove overlaps among clusters is required, since, in the current implementation, a considerable amount of overlaps exists in general. Also, an efficient algorithm to determine global routes among clusters is needed to reduce the computation time. Finally, in the detailed placement, a timing-driven cell assignment method to cell rows is sought.

REFERENCES