A One-Pass Standard Cell Placement Algorithm Using Multi-Stage Graph Model

(다단(多段)그래프 모델을 이용한 빠른 표준셀 배치 알고리즘)

趙 煥 奎*, 慶 宗 旻*

(Hwan Gue Cho and Chong-Min Kyung)

要 約

표준 셀을 이용한 고밀도 집적회로 설계시에 필요한 자동배치 알고리즘을 제시한다. 제안된 알고리즘은 두 단계로 구성되어 있으며, 첫 단계에서는 다단 그래프 모델을 이용하여 각 셀을 적당한 열(row)에 배치하고 만일 via셀이 필요하면 각 해당되는 열에 넣어준다. 둘째 단계에서는 모든 셀의 위치를 그 해당된 열 내에서 주어진 비용함수가 되도록 최소가 되게 유지하면서 왼쪽에서 오른쪽으로 한개씩 정해나간다. 실험결과에 의하면 이 방법은 종래의 쌍교환(pairwise interchange) 방식이나 일반화된 힘방향 이완(generalized force-directed relaxation) 등의 반복수행 방식에 비해 약 100배정도의 빠른 속도로 동작되며, via 셀의 갯수와 수평트랙의 갯수에 대해서는 본 알고리즘에 의해 근사최적 값을 얻을 수 있었다.

Abstract

We present a fast, constructive algorithm for the automatic placement of standard cells, which consists of two steps. The first step is responsible for cell-row assignment of each cell, and converts the circuit connectivity into a multi-stage graph under the constraint that sum of the cell-widths in each stage of the multi-stage graph does not exceed maximum cell-row width. Generation of feed-through cells in the final layout was shown to be drastically reduced by this step. In the second step, the position of each cell within the row is determined one by one from left to right so that the cost function such as the local channel density is minimized. Our experimental result shows that this algorithm yields near optimal results in terms of the number of feed-through cells and the horizontal tracks, while running about 100 times faster than other iterative procedures such as pairwise interchange and generalized force directed relaxation method.

I. Introduction

As various CAD (Computer-Aided Design) tools are developed in recent VLSI design process, automatic layout generation and verification steps especially require efficient design aids since layout tasks are extremely labor-intensive, error-prone and time consuming without CAD tools. While there are several major design strategies in VLSI implementation, we concentrate on the standard cell approach where the information on each cell such as its electrical behavior, detailed layout and external pin position and characteristics

*正會員, 韓國科學技術院 電氣 및 電子工學科
(Dept. of Elec. Eng., KAIST)
**正會員, 韓國科學技術院 電算學科
(Dept. of Computer Science, KAIST)
接受日字 : 1987年 5月 9日

are prestored in the cell library. Standard cells are typically designed to fit together in rows, where common signals such as power, ground and clocks run through the cells horizontally at fixed position. The purpose of the cell placement in the standard cell method is to ease the ensuing routing task in as small as possible by minimizing such objective functions as total wiring length, the number of feed-through cells and total channel density.

Placement algorithms for standard cells can be basically categorized into two types, i.e., constructive placement and iterative improvement. In practical implementation, the constructive placement is used for an initial placement and the result is refined using various iterative improvement methods. For the initial placement there are several strategies such as graph cluster[1], graph partitioning based on min-cut scheme[2,3], estimation of relative cell position[4] and linear ordering/folding[5,6,7]. For iterative improvement, pairwise cell exchnage[2,6] and force directed relaxation[4,7] methods and simulated annealing are generally used[8].

This paper has focused on obtaining a one-pass (noniterative) solution for a standard cell placement, which does not have the time-consuming trial-and-error nature of the iterative improvement approach. In the first step of our procedure, each cell is assigned its row in the first step using a multi-stage graph model which is rather a natural abstraction of the standard cell layout and explained in Section II. The specific position of each cell within the row is determined in the second step based on the evaluation of an objective function reflecting the local channel density at the cell rectangle being crossed by a hairline cursor which sweeps from the left end of the row to the right. In both steps, the asymptotic time complexity is αn^2, where n is the total number of cells; however, the actual CPU time consumed is nearly minimal since the propotionality constant, α is very small due to the fact that both procedures are of one-pass, noniterative nature, and the respective cell position is not disturbed any more once determined.

II. Multi-Stage Graph Model for Row Assignment

In the channel routing step following the cell placement, the cost of interconnection can be reduced 1) if the cells to be interconnected are located in the same row or in the adjacent rows, and/or 2) if the cell positions within each row is adjusted such that the channel density is minimal. Condition 1) can be satisfied by incorporating a multi-stage graph model, which is as follows, while the line sweep method explained in Section III is responsible for meeting condition 2).

We draw a graph by mapping each cell into vertex and drawing edges between vertices if the corresponding cells are sharing a signal net, and modify the resultant graph into a multi-stage graph by creating dummy nodes (feed-through cells), if necessary. Multi-stage graph is denoted by $G(V_1, V_2, ..., V_n)$. We call V_1 as stage 1, V_2 as stage 2, and V_n as stage n. Each stage is a subset of V, which is the total vertex set of the multi-stage graph, G. An important feature of the multi-stage graph is that every edge in G connects either the vertices belonging to the same stage or those in the neighboring stages such that no edges can cross over any stage without creating dummy nodes in the intermediate stages in between. For a more formal definition, if we let (p, q) be an edge of multi-stage graph G, where $p \in V_i$ and $q \in V_j$, then the difference between the two stage indices is at most 1, i.e.,

$$|j - i| \leqslant 1.$$

As an example, Fig. 1 (a) shows a graph where each vertex represents cell, and edges denote the inter-cell connection, while (b) shows its multi-stage graph representation. We note that an extra node denoted as F in stage 3 represents a feed-through cell in the cell placement obtained from this multi-stage graph.

Actually, in constructing a multi-stage graph having a constant number of stages from a given circuit, we may insert a dummy vertex (feed-through cell) into the stage V_j to maintain the property of the multi-stage graph, when edges crossing over the stage V_j exist. The process of transforming a given graph into a multi-stage graph begins by forming the base row which is the stage 1. Let the average

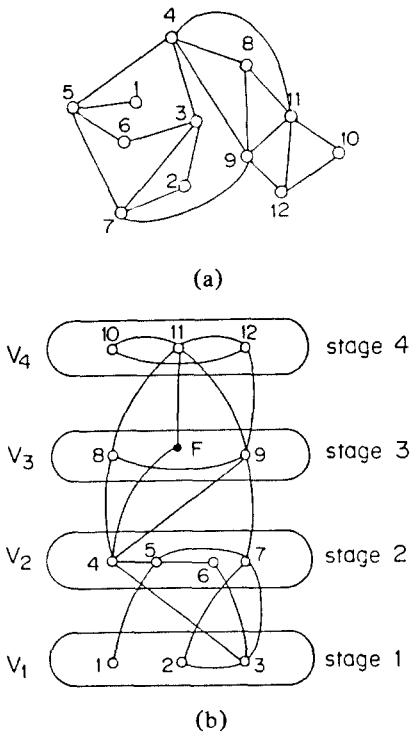

Fig. 1. A graph and its multi-stage graph.

row width be W_{avg} which can be approximately computed from the sum of total cell width and the number of cell rows. We choose a seed cell which is least connected to other cells. Subsequently, cells are incorporated into the base row one by one until the total width of base row exceeds W_{avg}. To describe the row assignment process, we divide the whole cells into two groups: one group called staged group consist of cells which were assigned their respective rows and the other group called unstaged group consist of cells which were not yet assigned the row. Having selected the seed cell, we select a cell among the unstaged group which maximizes the following objective function $F_1(c)$ and add it to base row:

$$F_1(c) = C_1 * N_1 - C_2 * N_2$$

, where c is a candidate unpartitioned cell. N_1 and N_2 denote the number of cells in the staged group adjacent to c, and the number of cells in the unstaged group adjacent to c, respectively, while the coefficients C_1 and C_2 are empirical constants.

After the cell assignment in the base row is completed, the assignment of remaining cells in the subsequent rows is performed. Let us assume that all the rows from the 1st stage (base row) to the i-th stage has been occupied with cells and we will describe how to construct the (i + 1)-th stage. Let Adj_row(i) be the set of unstaged cells which is connected to one or more cells in the i-th stage(row). Adj_row(i) then becomes a pool of cells which are candidates to be positioned in the (i + 1)-th row. If the sum of the widths of cells in Adj row(i) is greater than W_{avg}, then some cells in Adj row (i) are to be deleted. When we delete a cell from Adj_row(i), the cell whose removal causes the generation of minimal number of feed-through cells are removed first. We define a discarding function $F_2(c)$ to select a cell to be removed from Adj_row(i):

$$F_2(c) = |N_a| - |N_b|$$

, where N_a is the set of nets linking cell c to some cell in the (i)-th stage and N_b is the set of nets among N_a, to which another cell in Adj_row(i) belongs. It is easy to see that $F_2(c)$ is the number of feed-through cells to be generated to make a multi-stage graph, if the cell c is removed from Adj_row(i). This deletion process continues until all the reamining cells in Adj_row(i) can fit in the (i + 1)-th row. In Fig. 2, we show the staged cells in the i-th stage and the cells in Adj_row(i). Based on this removal strategy, we choose to remove the cell *a* and *b* since the removal of cell *a* or *b* does not incur the generation of feed-through cells, while removing cell *c*, creates a feed-through cell in the (i+1)-th stage.

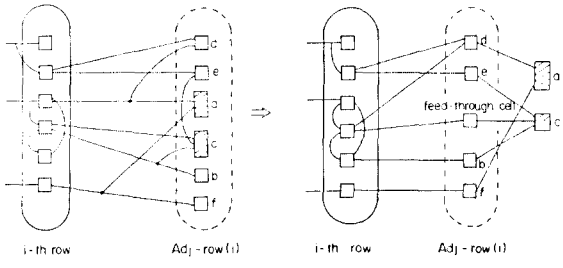

Fig. 2. Cell deletions which cause the feed-through or not from Adj-row(i).

On the other hand, if the sum of the widths of the cells in Adj_row(i) is less than W_{avg}, additional cells are to be recruited into the Adj_row(i) to fill in the empty slots in Adj_row(i). We choose a cell from the unstaged cells based on the selecting function $F_1(c)$ and insert the selected cell into Adj_row(i). This procedure is shown as an abstract code in Fig. 3. Time complexity of the row assignment step is $O(n^2)$ where n denotes the total number of cells.

```
Multi-stage Graph Algorithm for Row Assignment

Input  : a graph denoting the electrical circuit
Output : a multi-stage graph (row assignment result)
(1) Compute the number of rows and average row width, W_avg.
(2) Make the base row (stage 1) and let i := 1.
(3) Get Adj_row(i).
(4) If the total width of Adj_row(i) > W then
      while (the total width of Adj_row(i) ≥ W_avg) do
        begin
          delete a cell from Adj_row(i) which minimizes
                a discarding function F1(c);
          if the deletion causes feed-through cell generation then
                insert feed-through cells into Adj_row(i);
        end;
    else
      while (the total width of Adj_row(i) < W_avg) do
          choose a cell out of unpartitioned cells which maximizes
                the selecting function F1(c) and
          insert it into Adj_row(i);
(5) Assign the cells in Adj_row(i) to the (i+1)-th stage.
(6) If the last row was constructed, stop here.
    Otherwise let i := i+1 and go to (3). //
```

Fig. 3. Row assignment algorithm using multi-stage graph.

III. Line Sweep Method for Intra-Row Cell Placement

After each cell is assigned its row to be placed in, the second step determines the exact physical location of each cell within the row. Initially, we calculate the left end and right end position of each row within the chip assuming that all rows are centered symmetrically at a position to minimize the total chip area as shown in Fig. 4, where B_i and E_i denote left end and right end position of the i-th row, respectively.

To describe the line sweep method for the cell positioning within the row, we define the

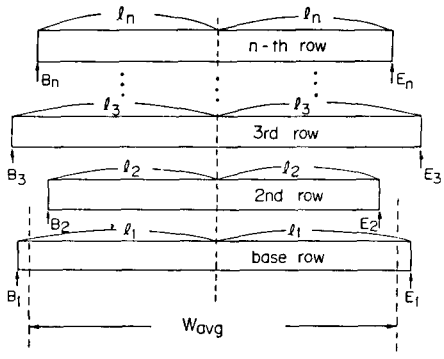

Fig. 4. Global chip configuration.

i-th point P_i as the x-coordinate of the interface line between the placed cells and the unplaced cells in the i-th row. Fig. 5 shows a snapshot taken during the cell positioning process using line sweep method, where placed cells are those which already found its final position and denoted as shaded boxes, while unplaced cells are those still remaining in the waiting pool of the relevant row and denoted as white boxes. The current sweep line shown as a dotted vertical line at $x = x_s$ is hopping rightward from its current x-position (P_2 in Fig. 5) to the next position (P_3), which becomes the active front being ceterminded as $min_i \{P_i\}$.

Fig. 5 shows an intermediate state of this process, where $P_1 = 10$, $P_2 = 7$, $P_3 = 9$, $P_4 = 14$. At the beginning of the seonc step, P_i is set to B_i Suppose we have placed several cells in rows such as in Fig. 5; we then find the m-th row which satisfies $P_m = min_i \{P_i\}$. The next cell to be placed is then chosen from the group of unplaced cells in the m-th row such that a selecting function is maximized. This is shown

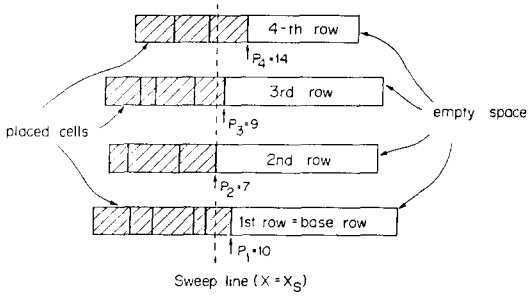

Fig. 5. Line sweeping for intra-row cell placement.

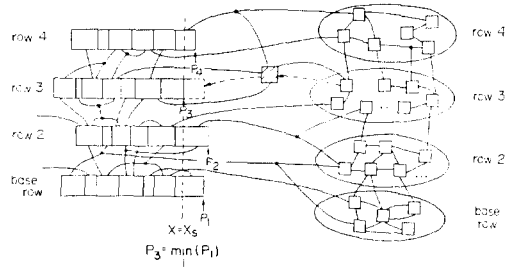

Fig. 6. An intermediate state of step2 : shaded cell is selected for row 3.

Line Sweep Algorithm for Intra-Row Cell Placement

Input : the row partition obtained from the first step.
Output : the final standard cell placement layout.

(1) Compute B_i and E_i for all rows.
(2) Let $P_i := B_i$ for all rows.
(3) Find m-th row such as $P_m = min\{P_i\}$.
(4) For the unplaced cells in m-th row
choose a cell ,c which maximizes the selecting function $F_3(c)$.
(5) Advance the position of P_m the position such that
$P_m := P_m$ + the width of the cell c.
(6) If there still remain upnlaced cell remained, then go to step(3).
Otherwise stop here. //

Fig. 7. Intra-Row cell placement algorithm by line sweeping.

in Fig. 6 where the placed cells are shown in the cell rows, while the unplaced cells are still in the multi-stage graph. A cell (shaded) selected from the stage 3 of the multi-stage graph. A cell (shaded) selected from the stage 3 of the multi-stage graph is shown to be joining the cell row 3, when the sweep line is at $x = x_s$. We prepare the selecting function $F_3(c)$ which computes the number of nets which are connected to cell c and cross over P_m. The function $F_3(c)$ is defined more formally as follows:

$$F_3(c) = N_c \cap N_p$$

, where N_c is the set of nets attached to the cell c, and N_p is the set of nets which should interesect the current vertical sweep line $x = P_m$. In addition, several tiebreaking schemes are provided for cases when a tie occurs in the values for F_3 (c) among many cells unplaced. This process continues for all rows simultaneously until all the cells are placed within the corresponding row. The intra-row placement procedure is briefly shown as an abstract code in Fig. 7.

IV. Conclusion

Since time complexities of both the multi-stage graph formation and the intra-row cell placement are $O(n^2)$, the overall time complexity of the standard cell placement procedure proposed is $O(n^2)$, where n is the number of standard cells. However, since both the multi-stage graph formation and line sweep process are inherently of one-sweep nature requiring no iteration the proportionality factor α(in αn_2) is very small and the actual CPU time is significantly less compared to other methods based on iterative improvement such as GFDR (Generalized Force Directed Relaxation) or PI (Pairwise Interchange).

We implemented this procedure using C language on VAX 11/750 and applied it to three example circuits prepared, shown as ALU, COM and CAL in Table 1.

It compares the CPU time, number of horizontal tracks and the number of feed-through cells generated among our procedure explained in this work, the procedure based on LOF (linear ordering and folding) for initial placement and nearly exhaustive PI (Pairwise Interchange) [6] for iterative improvement (LOF + PI) and LOF followed by GFDR (General Force Directed Relaxation)[7] for iterative placement improvement (LOF+ GFDR).

It is worth pointing out that the CPU time and the number of feed-through cells is reduced drastically, while the number of horizontal tracks generated is still comparable. Fig. 8 shows the result of channel routing using greedy algorithm for the placement result of CAL example obtained from the proposed placement procedure. Based on this experimental result, we like to propose that the row partitioning procedure using the multi-stage graph is more natural than the other circuit partitioning procedure such as Min-Cut and linear ordering/folding in standard cell placement problems. Since the global consideration

Table 1. Experimental results

Circuit name	No. of cells	CPU(time (Sec.)			No. of horizontal tracks			No. of feed-thru cells		
		LOF+ PI	LOF+ GFDR	this work	LOF+ PI	this GFDR	LOF+ work	LOF+ PI	LOF+ GFDR	this work
ALU	67	1445	407	4	34	33	32	7	5	0
COM	144	4424	1451	12	46	45	49	5	5	0
CAL	270	12350	4921	40	82	72	70	50	19	0

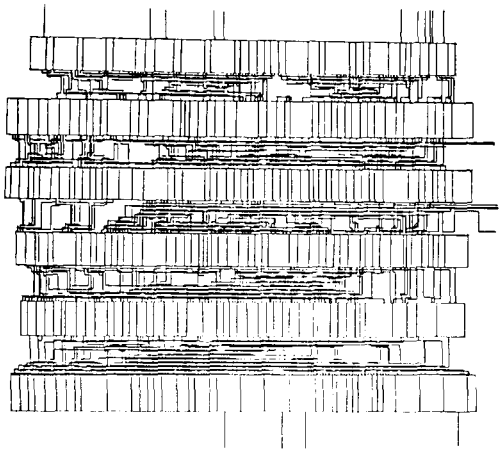

Fig. 8. Automatically generated layout for circuit, CAL, with 270 cells.

for the cell placement was already given in the multi-stage graph formulation step, the ensuing cell positioning procedure was able to produce a cost-effective and near-optimal solution, although the cell positioning is considered only within its row.

References

[1] B. D. Richard, "A Standard Cell Initial Placement Strategy", *Proc. 21th Design Automation Conf.*, pp. 392-398, 1984.

[2] M. Murakata et. al., "A Standard Cell Placement Algorithm with Predictive Row Width Equalization", *Proc. ICCAD.*, pp. 374-377, 1986.

[3] A. E. Dunlop and B. W. Kernighan, "A Procedure for Placement of Standard Cell VLSI Circuits", *IEEE Trans. CAD*, vol. CAD-4, no. 1 pp. 92-98, Jan. 1985.

[4] K. M. Just, J. M. Kleinhans and F. M. Johannes "On the Relative Placement and The Transportation Problem For Standard-Cell Layout", *Proc. 23rd Design Automation Conf.*, pp. 308-312, 1986.

[5] S. H. Kang, "Linear Ordering and Application to Placement", *Proc. 20th Design Automation Conf.*, pp. 457-463, 1983.

[6] G. S. Kang, *"A Study on the Automatic Placement System for Standard Cell"*, M.S. thesis, KAIST, Feb. 1986.

[7] N. W. Eum, *"A Study on the Automatic Layout System for Standard Cell"*, M.S. thesis, KAIST, Feb. 1987.

[8] M. R. Hartoog, "Analysis of Placement Procedures For VLSI Standard Cell Layout", *Proc. 23rd Design Automation Conf.*, pp. 314-319, 1986.