第 23 卷第 12 期 半 导 体 学 报 Vol. 23, No. 12
2002 年 12 月 CHINESE JOURNAL OF SEMICONDUCTORS Dec., 2002

Incremental Placement Algorithm for Standard-Cell Layout*

Li Zhuoyuan[1], Wu Weimin[1], Hong Xianlong[1] and Gu Jun[2]

(1 *Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)
(2 *Department of Computer Science, HongKong University of Science & Technology, Hongkong, China*)

Abstract: A new algorithm W-ECOP is presented to effect incremental changes on a standard cell layout automatically. This algorithm deals with cell inserting and cell moving based on rows instead of on cells as most placement algorithms usually do. An integer programming problem is formulated to minimize the adjustment on the initial placement and a heuristic method is presented to search for a shifting path so as to optimize the wirelength. Test of W-ECOP on a group of practical test cases shows that the algorithm can successfully accomplish incremental placement with good quality and high speed.

Key words: incremental placement; standard cell
EEACC: 2570 **CCACC**: 7410D
CLC number: TN47 **Document code**: A **Article ID**: 0253-4177(2002)12-1338-07

1 Introduction

As very large scale integration (VLSI) technology advancing fast, the scale of the IC layout expands at a surprising speed. However, the layout problem is so difficult that optimal layouts cannot be found in a reasonable time period[1]. Even a very simplified view of the layout optimization problem is NP-complete or NP-hard[2]. Thus, the layout problem is often broken up into several phases such as partitioning, floorplanning, placement, and routing. Each phase solves a sub-problem that is not quite as hard as the total layout problem. Placement is considered as the most important phase of the IC layout design because of its difficulty. The result of the placement directly affects the area and the performance of the chip, so placement algorithms often use iterative optimization methods to generate a solution that is close to optimum[3,4].

Sometimes the IC layout should be modified after placement to satisfy timing constraints or to complete clock net routing and the modification is always local or slight. General purpose placement algorithms cannot take advantage of these situations, because they are designed to generate a complete placement from scratch and thus very time consuming. Efficient algorithms are needed to obtain good incremental solutions in short time. The algorithm should preserve as much previous placement as possible so as to maintain the performance of the circuit. It is called incremental placement or ECO (Engineering Change Order) placement[5].

Choy *et al*. have presented two algorithms for incremental placement for gate array design[6]. It first searches the nearest empty slot for an added cell. Then it determines an optimal shifting path to move the cell to its neighboring slot by AUT (Algorithm Using Template) or AVP (Algorithm by

* Project supported by National Natural Science Foundation of China(No. 69776027) and 973 National Key Foundation Research Program of China(No. G1998030413)

Li Zhuoyuan male, was born in 1980, master candidate. His research interests focus on VLSI layout design.

Received 20 March 2002, revised manuscript received 7 July 2002 ©2002 The Chinese Institute of Electronics

Value Propagating). Because each cell can be only in its neighboring slots, the change to the initial placement can be minimal. It is a prior advantageous work on incremental placement. After that a scattered literature in the area of incremental placement appeared[8~12]. But as the prevailing design style in industry, standard cell layout design has received little attention in incremental placement.

In this paper, an incremental placement algorithm for standard cell layout is proposed. In Section 2, we present the objective function. The wirelength-driven incremental placement algorithm W-ECOP are described in Section 3. Experimental results are given in Section 4. Some issues, such as the efficiency of our algorithm and when the use of the algorithm is appropriate and the merits of algorithm implementation, are discussed in Section 5. The conclusion follows in Section 6.

2 Underlying principle

Incremental changes to a circuit usually involve the removal or the addition of logic elements from or to a finished circuit. Incremental placement algorithm is to complete the modification with the lowest cost. An algorithm ECOP is presented to deal with the problem automatically[7]. For an added/removed cell (which is called trouble cell) a row that can accept it can be easily found by comparing the width of the total free space in the row with the width of the trouble cell. Then the cell is inserted into the row vertically, while cell in the row should move to ensure no overlapping with the trouble cell. The total move-distance of all affected cells during the process can be got quickly so the optimal row with the minimum move-distance can be determined. The trouble cell should be inserted into this row. Because the total moving-distance is minimized during the process, the upset to the initial circuit is expected to be minimized so that the performance of circuit is preserved.

ECOP has an obvious drawback. The connectivity between the trouble cell and the rest of the circuit is completely ignored. As a result, the routing paths to and from this cell may be unnecessarily long. This will seriously affect the performance of the circuit. A better solution is to place the new element to where its overall connection length to the rest of the circuit is minimal. However, the place may not be in any row and its neighboring rows may have not enough free space to accept the cell. How to adjust the positions of the rest cells is a difficult problem. Our goal is to find an efficient scheme to solve it.

The underlying principle of our work is we assume that the initial placement is a good placement. So the algorithm should not only minimize the wirelength but also modify the circuit with minimum adjustment. On these basis, some criteria are used to estimate the result:

(1) The trouble cell should be positioned to minimize the wirelength;

(2) The maximal moving-distance of a cell, generally classified into x-distance and y-distance, should be minimized;

(3) On these two bases, the number of moved cells should be minimized.

As a result, most cells are on their initial positions after incremental placement and cell moving tends to be restricted to its neighboring area. Thus the increase in wirelength is minimized and the circuit performance is preserved.

3 Algorithm

3.1 Outline

The algorithm W-ECOP has three main steps:

(1) Determine the optimal position: Determine the optimal position for a trouble cell;

(2) Insert Trouble Cell: Place a trouble cell to the optimal row and eliminate overlap;

(3) Shift Path Searching: If free space in the row cannot accept the trouble cell, move some cells out and search for an optimal shifting path.

3.2 Determine optimal position

A heuristic method is used to determine the optimal position for a trouble cell. Assuming that cells connected to the cell C_i are fixed, the wire-length of the nets connected to C_i is computed as follows:

$$L_p = \sum_{n \in N_i, i \neq j} w_n [(x_i - x_j)^2 + (y_i - y_j)^2] \quad (1)$$

where w_n represents the weight of net_n.

Let the differential coefficient of L_p equal zero:

$$\frac{\partial L_p}{\partial x_i} = 0, \quad \frac{\partial L_p}{\partial y_i} = 0 \quad (2)$$

Then the optimal position of C_i to minimize the wirelength is as follows:

$$\begin{aligned} x_i &= \left[\sum_{n \in N_i, i \neq j} w_n x_j \right] \Big/ \sum_{n \in N_i, i \neq j} w_n (n - 1) \\ y_i &= \left[\sum_{n \in N_i, i \neq j} w_n y_j \right] \Big/ \sum_{n \in N_i, i \neq j} w_n (n - 1) \end{aligned} \quad (3)$$

where n represents the number of connect pins of net_n.

Note that we use the bounding-box model as an approximation of wire-length. Other models such as Steiner-tree also can be used here to get a more precise approximation. Because it just affects the position of a simple trouble cell, experiments show that it does not determine the final results.

3.3 Insert trouble cell

Let us consider the row R_k beside position (x_i, y_i). If free space in R_k can accept C_i, C_i should be inserted into R_k and the rest cells in R_k should move to ensure no overlapping with C_i.

Definitions of some parameters are as follows:

W_i: the width of C_i

C_{ki}: the i-th cell in R_k counted from the left

G: Assume that C_i is inserted between C_{kj} and $C_{k(j-1)}$ according to its x-coordinate, G is the width of free space between C_{kj} and $C_{k(j-1)}$.

If G is larger than W_i, C_i should be inserted to the free space between C_{kj} and $C_{k(j-1)}$ directly and the positions of the rest cells remain. Otherwise the free space should extend to accept C_i, we hope to find a scheme with the least cost to adjust the positions of the rest cells. The optimization object is to minimize the total number of moved-cells. Some other parameters are used here:

ME: the width of free space needed to extend

$$ME = W_i - G \quad (4)$$

L, R: the total number of cells on the left and right of the free space, respectively

C_{li}: the i-th cell on the left of the free space counted from the right

W_{li}: the width of C_{li}

G_{li}: the width of free width between $C_{l(i-1)}$ and C_{li}

C_{ri}: the i-th cell on the right of the free space counted from the left

W_{ri}: the width of C_{ri}

G_{ri}: the width of free width between C_{ri} and $C_{r(i+1)}$

The order of cells is remained when adjusting the positions of cells. Cells on the left of the free space are moved to the left one by one. First C_{l1} moves to C_{l2}, the width of free space can extend G_{l1}. Then C_{l1} and C_{l2} move to C_{l3}, the width of free space can extend $G_{l2} + G_{l1}$. When they move to C_{lp}, the width of free space extended is ML_p.

$$ML_p = \sum_{i=0}^{p} G_{li} \quad (5)$$

In the same way we can get the width of free space extended when cells on the right move:

$$MR_p = \sum_{i=0}^{p} G_{ri} \quad (6)$$

So the integer programming (IP) formulation of the problem is as follows:

$$\begin{cases} \text{Minimize} \quad l + r \\ \text{s.t.} \\ 0 \leqslant l < L, \ 1 \in N; \\ 0 \leqslant r < R, \ r \in N; \\ ML_l + MR_r \geqslant ME \end{cases} \quad (7)$$

A special method named dual-diagonal searching (DDS) is applied for the special IP problem. It is named so because searching for the optimal solution is accomplished along two diagonal directions in a $(L+1)(R+1)$-matrix $\boldsymbol{M}$, whose element m_{ij} e-

qual $ML_i + MR_j$:

$$\boldsymbol{M} = \begin{bmatrix} ML_L & ML_L + MR_1 & \cdots & ML_L + MR_R \\ \cdots & \cdots & \cdots & \cdots \\ ML_2 & ML_2 + MR_1 & \cdots & ML_2 + MR_R \\ ML_1 & ML_1 + MR_1 & \cdots & ML_1 + MR_R \\ 0 & MR_1 & \cdots & MR_R \end{bmatrix} \tag{8}$$

The algorithm has three substeps:

(1) Search for a feasible solution along the main diagonal

The searching list is as follows:

$$\begin{cases} m_{00}, m_{11}, \cdots, m_{RR}, m_{(R+1)R}, \cdots, m_{LR} & L > R \\ m_{00}, m_{11}, \cdots, m_{LL}, m_{(L+1)L}, \cdots, m_{RL} & L \leqslant R \end{cases} \tag{9}$$

The first element m_{pq} that is prior to ME can be the feasible solution. And the minimum total number of moved-cells N_u equal $(p+q)$.

(2) Search left branch for a possible optimal solution

Assume that the current feasible solution is m_{lr}, the next feasible solution $m_{l'r'}$ is determined as follows:

$$l' = \begin{cases} l-1 & (m_{1r} \geqslant ME) \quad \text{or} \\ & (m_{1r} < ME \quad \text{and} \quad r' + l > N_u) \\ l & (m_{1r} < ME \quad \text{and} \quad r' + l \leqslant N_u) \end{cases}$$

$$r' = \begin{cases} r & (m_{1r} \geqslant ME) \\ r+1 & (m_{1r} < ME) \end{cases} \tag{10}$$

until $l' < 0$ or $r' > R$. When we get a new feasible solution $m_{l'r'}$, N_u should be updated to $(l' + r')$.

(3) Search right branch for another possible optimal solution

It also begins from m_{pq}. Assume that the current feasible solution is m_{lr}. The next feasible solution $m_{l'r'}$ is determined as follows:

$$r' = \begin{cases} r-1 & (m_{1r} \geqslant ME) \quad \text{or} \\ & (m_{1r} < ME \quad \text{and} \quad l' + r > N_u) \\ r & (m_{1r} < ME \quad \text{and} \quad l' + r \leqslant N_u) \end{cases}$$

$$l' = \begin{cases} l & (m_{1r} \geqslant ME) \\ l+1 & (m_{1r} < ME) \end{cases} \tag{11}$$

until $r' < 0$ or $l' > L$. N_u should also be updated to $(l' + r')$ when we get a new feasible solution $m_{l'r'}$.

After an optimal solution m_{pq} is determined, cells $C_{l1}, C_{l2}, \cdots, C_{lp}$ should move to the left and have no free space between them. Cells $C_{r1}, C_{r2}, \cdots, C_{rq}$ should move to the right in the same way. Then the width of free space between $C_{k(j-1)}$ and C_{kj} is sufficient for accepting C_i. C_i can be inserted without overlapping.

3.4 Shifting path searching

If free space in R_k cannot accept C_i, some cells in R_k should move out to ensure that C_i can be placed without overlapping. Which cell should be moved out and where to place it are the difficult problems. We want to find an efficient scheme to solve it.

For each cell in R_k, its optimal position (x_i, y_i) to minimize the wirelength will be computed according to equation (3) and compared with its position (x, y) and the y-coordinate of $R_{k-1}(y_{k-1})$ and $R_{k+1}(y_{k+1})$. The result is stored into a flow-tendency information library, which is defined by a two-dimensional array BUCK[n][2], where n is cell index in R_k and BUCK[i][0] equal 1, −1 or 0 corresponding to x_i smaller than, larger than or equal to x, and BUCK[i][1] equal 1, −1 or 0 corresponding to y_i larger than y_{k+1}, smaller than y_{k-1} or else.

Let us consider cells on the left of C_i. Assume cell C_{kr} is exactly the cell C_{li} (just see the definition in 3.3) and it satisfies the following two conditions:

(1) BUCK[r][1]! = 0

(2) $ML_j + W_{lj} \geqslant W_i$

Then it can move to R_{k-1} or R_{k+1} (according to the value of BUCK[r][1]) and cells between C_i and C_{kr} move to the left so as to erase overlap.

A placement cost, cost(C_{kr}) is introduced to minimize the wierlength and the upset to the initial placement, cost(C_{kr}) consists of three cost c_1, c_2 and c_3 and is defined as

$$\text{cost}(C_{kr}) = w_1 c_1(C_{kr}) + w_2 c_2(C_{kr}) + w_3 c_3(C_{kr}) \tag{12}$$

where $w_i (i = 1, 2, 3)$ is a weight for $c_i (i = 1, 2, 3)$

and c_i is defined as bellow.

The cost $c_1(C_{kr})$ is computed based on the flow-tendency of cells between C_{kr} and C_i. It is computed as

$$c_1(C_{kr}) = \sum_{m=r}^{i} w'_m \mathrm{BUCK}[m][0] \qquad (13)$$

where w'_m is a weight according to the number of nets connected to C_{km}. The more nets connect to C_{km}, the larger w'_m is, because moving C_{km} will cause wirelength increase/decrease more. Larger $c_1(C_{kr})$ demenstrates that more cells have a tendency to move to the left.

The cost $c_2(C_{kr})$ is computed based on the width of C_{kr}, it is defined as

$$c_2(C_{kr}) = -wW_{kr} \qquad (14)$$

where w is a weight. Because C_{kr} will be inserted to another row (for example, R_{k+1}), the larger $c_2(C_{ki})$ is, the smaller W_{kr} is, then the less upset to cells in R_{k+1} is expected.

The cost $c_3(C_{kr})$ is computed as

$$c_3(C_{kr}) = -|i-r| \qquad (15)$$

Notice that C_i is inserted between C_{kj} and $C_{k(j-1)}$. $|i-r|$ demonstrates the number of moved-cells during the course. If the cost of $c_3(C_{kr})$ is large, the number is small, and the upset to R_k is small.

The process is the same to cells on the right of C_i and C_{kr} can be determined by maximize the cost fucntion. After a move-out cell C_{kr} is determined, C_i is inserted into R_k and the cells between C_i and C_{kr} moved to C_{kr} to erase overlap. C_{kr} is regarded as a new trouble cell. Then we redo the same process for C_{kr} and the flow-directions in y-orientation of latter move-out cells are consistent to C_{kr}. Finally the shifting path is available.

Let us reconsider the two conditions above. They are the basis of the whole process. If there is no any cell satifying the conditions, or the process is not end automically (because the free space in the next row always can not accept the move-out cell), the incremental placement will fail or trace into a timeless process. A huristic method is presented to deal with these cases:

When meeting a row having no cell satisfying the two conditions or the search-and-move process has carried on for five times, our algorithm searches an optimal row for current trouble cell. The optimal row should have enough free space to accept the move-out cell and can be determined by the following two functions:

F_1= the total moving-distance in inserting and moving cells;

F_2= the number of moved cells.

and the cost function relating to the best row is given by

$$\text{Cost Function} = w_1F_1 + w_2F_2 \qquad (16)$$

where w_1 and w_2 are weights of F_1 and F_2, respectively and the DDS method is used when computing the value of F_1 and F_2. Thus, the optimal row can be obtained by minimizing the Cost Function and the trouble cell is inserted.

4 Experimental results

To show the effectiveness and utility of the proposed algorithm, we apply it to a set of circuits from industry (test cases from Ultima Company). The experiments are done on a Sun Ultra Sparc-1 Work-Station with 4GB memory. The information of test cases are given in Table 1. In each case the number of trouble cells is less than twentieth of all the cells in the circuit. Results compared with E-COP are reported in Table 2. Note that the wirelength is calculated in Steiner trees model and based on the exact interconnected pins. As one can see, there is a wirelength reducing between 0.1 to 1 percent compared to ECOP. And total wirelength is even reduced about 0.1 percent compared to the initial circuit. As expected, the run time of W-E-COP is very short and increases linearly related to the total number of cells in circuit.

It can be seen from Table 1 that the run time of W-ECOP is much shorter than ECOP. Because ECOP searches each row in the chip to place a trouble cell. But in W-ECOP trouble cells are placed in their neighboring rows and some small

cells move out to search for shifting path. The searching area is limited to part of the chip in W-ECOP. It leads to a high effectiveness.

Table 1 Information of circuits

Circuits	# Cells	W/mm	# TCells*
Cnt3	16	572.8	1
Cnt100	760	46293.2	32
Cnt1000	8150	1684053	357
M16_4	2870	205308	146
M32_my	7150	961634	350
Gfsm20	268	15373.7	16
Gfsm300	4154	727274	176
Sony_1	24847	9922919	1229
Toshiba	16444	4252005	1062

* # TCells: number of trouble cells

Table 2 Comparing W-ECOP with ECOP

Circuits	ECOP		W-ECOP		WL*	WL'**
	W'/mm	Runtime/s	W'/mm	Runtime/s	Dec. %	Dec. %
Cnt3	556.1	0.01	548.3	0.00	4.27	1.36
Cnt100	46376.7	0.06	46209.7	0.01	0.18	0.37
Cnt1000	1684936	9.30	1682999	0.29	0.06	0.11
M16_4	206065	1.11	204964	0.07	0.16	0.50
M32_my	962017	8.74	960653	0.25	0.10	0.10
Gfsm20	15422.6	0.02	15370.5	0.00	0.02	0.34
Gfsm300	727760	1.81	726650	0.10	0.08	0.15
Sony_1	9940513	106.49	9927345	4.12	-0.04	0.13
Toshiba	4264968	54.74	4251982	2.93	0.00	0.30

* WL: Wirelength comparing to the initial placement; ** WL': Wirelength comparing to ECOP

5 Discussion

Our algorithm is proposed to deal with incremental layout changes. Incremental placement is usually caused by adding cells/nets, reducing congestion or meeting timing constraints. These adjustments are known to only after the layout design is proposed so it is unpredictable and sometimes frequent. So the algorithm should be with high efficiency. This is one of the most important objects of our algorithm. Our algorithm deals with one single added/moved cell at a time. It tries to get an optimal adjusting scheme for it, and an iteration of the algorithm is necessary to deal with a series of the cells. Therefore, it is obvious that our algorithm is suitable for small design changes including a small portion of cells. If the cells that are needed to be moved are of a large number, it is necessary to redo the completely placement.

In incremental placement, the modification to the placement is separated into two steps: First, the designer presents his opinion on modifying the design and some constraints. The schemes of modification can also be carried out by a particular algorithm designed for timing or congestion. Second, our algorithm carries on the practical modification according to the opinions/schemes and constraints. So the other aim of our algorithm is to preserve the performance of the initial placement. We try to achieve it by minimizing the number of moving cells (as described in 3.2) and constraining the moving bound of a cell to its neighboring area (as described in 3.3).

How to adjust cell position to erase overlapping is of fundamental importance for any incremental placement algorithm. If free space in the initial circuit is enough in any part of the circuit, it is easy to find a free space to place the added cell. The algorithm ECOP will achieve a good result. But when the free space is absent and there is little free space around the optimal position for the added cell, the shifting path searching process becomes very important. It can constrain cell moving in its neighboring area to preserve the performance and get high speed as the results shown in Table 2.

As mentioned above, there are many constraints such as KEEPOUT areas for routing, fixed cells and different height rows and overlap rows in Stand-cell layout. How to deal with these constraints is also a problem. A row-partition method is adopted to maintain the inner-integrality of each row and assure the validity of our algorithm. Because this is not the main point of our algorithm, it is not given here.

6 Conclusion

This paper presented a new wirelength driven algorithm for incremental placement of standard cell layout. The proposed algorithm automatically

adjusts the placement of a finished layout to effect any change to the circuit. Thus, a designer need not re-execute the whole placement process after modifying the circuit. The algorithm determines the optimal position for a trouble cell and constructs integer programming problem to insert it to its neighboring row. If necessary, it searches for an optimal shifting path when moving cells to ensure perturbing the initial placement the least. Experimental results demonstrate the effectiveness of the new approach.

References

[1] Preas B, Lorenzetti M. Physical design automation of VLSI systems. The Benjamin-Cummings Publishing Company, 1988

[2] Garey M, Johannes D. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: Freeman, 1979

[3] Hauge P, Nair R, Yoffa E. Circuit placement for predictable performance. In: Proc ICCAD, 1987: 88

[4] Ou S, Pedram M. Timing-driven placement based on partitioning with dynamic cut-net control. In: Proc Design Automation Conf, 2000: 427

[5] Cong J, Fang J, Khoo K Y. An implicit connection graph maze routing algorithm for ECO routing. In: Proc ACM/IEEE International Symposium on Physical Design, 1999: 214

[6] Choy C, Cheng T, Wong K. Incremental layout placement modification algorithms. In: IEEE Trans Computer-Aided Des Integr Circuits Syst, 1996, 15(4): 437

[7] Yu Hong, Yao Bo, Hong Xianlong, et al. ECOP: an incremental placement algorithm based on row partitioning for standard cell layout. Chinese Journal of Semiconductors, 2001, 22: 96(in Chinese) [于泓, 姚波, 洪先龙, 等. ECOP: 一种基于单元行划分的标准单元模式增量式布局算法. 半导体学报, 2001, 22: 96]

[8] Togawa N, Hagi K, Yanagisawa M, et al. An incremental and global routing algorithm for field-programmable gate arrays. In: Proc ICCAD, 1998: 519

[9] Cong C, Huang H. Technology mapping for field programmable gate arrays with incremental changes. In: Proc 37th Design Automation Conference, 2000: 290

[10] Cong J, Sarrafzadeh M. Incremental physical design. In: International Symposium on Physical Design, 2000

[11] Emmert J M, Bhatia D. Incremental routing in FPGAs. In: IEEE Inl ASIC Conference And Exhibit, 1998: 217

[12] Liu Yi, Hong Xianlong, Cai Yici, et al. A mew clock-driven ECO placement algorithm based on table-lookup delay model. Chinese Journal of Semiconductors, 2002, 23: 473

一种新的标准单元增量式布局算法*

李卓远[1] 吴为民[1] 洪先龙[1] 顾 钧[2]

(1 清华大学计算机科学与技术系, 北京 100084)

(2 香港科技大学计算机科学与技术系, 香港)

摘要: 提出了一种新的增量式布局方法 W-ECOP 来满足快速调整布局方案的要求. 与以前的以单元为中心的算法不同, 算法基于单元行划分来进行单元的插入和位置调整, 在此过程中使对原布局方案的影响最小, 并且尽可能优化线长. 一组从美国工业界的测试例子表明, 该算法运行速度快, 调整后的布局效果好.

关键词: 增量式布局; 标准单元

EEACC: 2570 **CCACC**: 7410D

中图分类号: TN47 **文献标识码**: A **文章编号**: 0253-4177(2002)12-1338-07

* 国家自然科学基金(批准号: 69776027)和 973 国家重大基础研究(批准号: G1998030413)资助项目

李卓远 男, 1980 年出生, 硕士研究生, 主要从事超大规模集成电路布图算法的研究.

2002-03-20 收到, 2002-07-07 定稿

©2002 中国电子学会