第 21 卷第 8 期　　半 导 体 学 报　　Vol. 21, No. 8
2000 年 8 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　Aug., 2000

Fame: A Fast Detailed Placement Algorithm for Standard-Cell Layout Based on Mixed Mincut and Enumeration*

YAO Bo(姚　波), HOU Wen-ting(侯文婷),
HONG Xian-long(洪先龙) and CAI Yi-ci(蔡懿慈)

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)

Abstract: Rapid progress in manufacturing greatly challenges to the VLSI physical design in both speed and performance. A fast detailed placement algorithm, FAME is presented in this paper, according to these demands. It inherits the optimal positions of cells given by a global placer and exact position to each cell by local optimization. FM Mincut heuristic and local enumeration are used to optimize the total wirelength in y and x directions respectively, and a two-way mixed optimizing flow is adopted to combine the two methods for a better performance. Furthermore, a better enumeration strategy is introduced to speed up the algorithm. An extension dealing with blockages in placement has also been discussed. Experimental results show that FAME runs 4 times faster than RITUAL and achieves a 5% short in total wirelength on average.

Key words: detailed placement; layout; VLSI; mincut; optimization

EEACC: 2570　　**CCACC**: 7410D

CLC number: TN402　**Document code**: A　**Article ID**: 0253-4177(2000)08-0744-10

FAME: 一个标准单元模式下基于最小割和枚举的快速详细布局算法

姚　波　侯文婷　洪先龙　蔡懿慈

(清华大学计算机科学与技术系, 北京　100084)

摘要: 随着制造工艺的快速进步, 超大规模集成电路的物理设计技术在速度和质量上面临很大挑战. 提出了一个快速详细布局算法以适应这种要求. 算法继承总体布局得到的单元全局最佳位置, 然后采用局部优化将单元精确定位. FM 最小割和局部枚举方法分别用于优化 y 和 x 两个方向的连线长度, 这两个方向的优化在同一迭代过程中交替进行. 另外, 采用改进的枚举策略

* Project Supported by National Natural Science Foundation of China (Grant No. 69776027) and by National 973 Key Project of China (Contract No. G1998030413)

Received 30 December 1999, revised manuscript received 23 March 2000

加速算法,对于有障碍和宏模块情况下的布局也加以讨论.实例测试结果表明,FAME 的运行速度比 RITUAL 快 4 倍,并使总连线长度平均减小 5%.

关键词: 详细布局; 布图; 超大规模集成电路; 最小割; 优化
EEACC: 2570　　**CCACC**: 7410D
中图分类号: TN402　　**文献标识码**: A　　**文章编号**: 0253-4177(2000)08-0744-10

1 Introduction

In the flow of VLSI's physical design, placement is the stage in which positions are allocated to cells or modules subject to certain constraints. Some object functions, such as total wirelength and total chip area are optimized at the same time. Standard-cell based design is widely used nowadays. The object of placement in this mode is a number of cells, which are of the same height and may differ in widths. Placement is proved to be a NP-Complete problem[1]. According to the NTRS roadmap(1997 edition)[2], there will be an estimated 20M usable transistors available for auto-layout, which makes the placement even harder to carry out. Therefore, placement of a large number of cells remains to be one of the critical technologies for IC industry in the following decade[3].

Currently, the mainstream algorithm of placement for standard-cells is based on quadratic programming. This kind of method has been proved mathematically to have good performance, so it is widely accepted. RITUAL[4] and GORDIAN[5] are two famous placement systems based on this method. Due to the difficulty of dealing with distributed constraints, i.e. the requirements that there should be no overlaps between cells, and cells should align to rows, placement has to be carried out in two successive stages: global placement and detailed placement. In global placement, cells are given optimal positions in global view, while in detailed placement, distributed constraints are taken into account. In order to inherit the global optimized results of global placement, local optimizations are chosen for the detailed placement.

The existing work of detailed placement can be divided roughly into two categories. One is based on mathematical programming and the other is based on heuristic methods like Mincut. The mathematical programming methods usually assume a number of locations with the same width, which are named sites, in the optimizing region, and calculate the cost of putting each cell into each site. An integer programming is constructed and solved to get the cell placement in that region. The detailed placement of RITUAL applies the mathematical programming method. This kind of methods can get good results, but need longer CPU time.

The heuristic methods usually begin with an initial legal placement, and a new placement with better object function value is tried to be obtained by exchanging or moving cells in local areas. The better solution is always kept at each optimization step. Mincut is a useful heuristic for partitioning. Depmin[6] applies the KL-Mincut[7] method to assign cells to rows in detailed placement. Mincut-based methods are simple to carry out and very fast in execution. Depmin, however, cannot give the satisfactory performance. It is partially because of the vulnerability of Mincut heuristics to local minimum, and the strict limitation of optimization to single direction in the flow.

In this paper, we propose a fast detailed placement algorithm, FAME. It adapts FM-Mincut heuristic[8], which is much faster than KL-Mincut, to assign cells into rows, and optimize the wirelength in y direction. Local enumeration of cell order within rows is used to minimize the wirelength in x direction. We interlace the two methods in the design flow and get good performance. To make the algorithm more faster, we find come better strategies for enumeration, like adjacent exchange and incomplete permutation. To deal with macro blocks or blockages, we extend FAME to cope with such constraints.

The rest of this paper is organized as follows. In section 2, the problem of detailed placement is defined and some assumptions are made. Section 3 describes the algorithm of FAME, including the flow and optimization steps. Section 4 focuses on the improvements to FAME: the better optimization flow, a speed up of enumeration and an extension to cope with the blockages or macro blocks. Experimental results are given in section 5. Conclusions are presented in section 6.

2 Problem Formulation

Given a design DS with p cells to be placed, we use $C=\{c_1, c_2, \cdots, c_p\}$ to denote the set of p cells in DS, $N=\{n_1, n_2, \cdots, n_q\}$ to denote the set of q nets in DS. To each cell c_i, the coordination (x_i, y_i) is set to the lower left point of the cell. The design area Π of DS is a rectangle one, determined by point $(0, 0)$ and (X_B, Y_B). K rows, $R=\{r_1, r_2, \cdots, r_k\}$, are defined in Π. We use Y_{ri} to denote the y coordinate of row r_i.

A detailed placement solution of the design DS satisfies the following conditions: 1) Each cell must be located within the placement area. 2) There is no overlap between cells. 3) Each cell must be assigned exactly to one row. i.e. if cell c_i is assigned to row r_j, $y_i=Y_{rj}$.

The object function that detailed placement is to be optimized in this paper is the total wirelength TWL of all nets. It is calculated as follows: $\mathrm{TWL}=\sum_{i=1}^{q} wl_j$, where wl_j is the connection length of net n_j. Let N_j be the number of cells connected by n_j and $\{c_{j1}, c_{j2}, \cdots, c_{jN_j}\}$ be the set of cells connected by n_j. Let (x_{jk}, y_{jk}) be the coordinate to cell c_{jk}. We use the coordinates of cells instead of pins to calculate the connected length of a net. Since the size of cell is small compared with the placement area, this approximation will not cause much error. wl_j is given as follows:

If $N_j<4$, bounding box model is used, i.e. $wl_j=x_{j\max}-x_{j\min}+y_{j\max}-y_{j\min}$.

If $N_j\geqslant 4$, a single trunk Steiner-tree model is used, i.e.

$$wl_j = \begin{cases} x_{j\max} - x_{j\min} + \sum_{k=1}^{N_j} |y_{jk} - \bar{y}_j| & \text{when } x_{j\max} - x_{j\min} \geqslant y_{j\max} - y_{j\min} \\ y_{j\max} - y_{j\min} + \sum_{k=1}^{N_j} |x_{jk} - \bar{x}_j| & \text{when } x_{j\max} - x_{j\min} < y_{j\max} - y_{j\min} \end{cases}$$

Where $x_{j\max} = \max_{k=1}^{N_j} x_{jk}$, $x_{j\min} = \min_{k=1}^{N_j} x_{jk}$, $y_{j\max} = \max_{k=1}^{N_j} y_{jk}$, $y_{y\min} = \min_{k=1}^{N_j} y_{jk}$, $\bar{x}_j = \sum_{k=1}^{N_j} x_{jk}/N_j$ and $\bar{y}_j = \sum_{k=1}^{N_j} y_{ik}/N_j$.

From the calculations above, TWL is the sum of two parts: one part involves x coordination only and the other involves y coordination only. We use the terms "total wirelength in the x direction" and "total wire length in the y direction" to refer to them respectively.

3 Algorithm of FAME

3.1 Outline of FAME

The flow of FAME is composed of three successive phases, as shown in Figure 1. In the first phase, we try to get an initial legal detailed placement from the global placement results. In the second phase, refining row assignment (RRA) and cell permutation within rows (PCR) are interlaced to optimize the total wirelength in y and x directions respectively. Making rows even (RE) and removing overlap (ORR) are used to make the interlacing possible. In the third phase, cell permutation is carried out more intensively to optimize the total wirelength in the x direction. Phase 2 and phase 3 work as the refinement to the initial solution generated by phase 1.

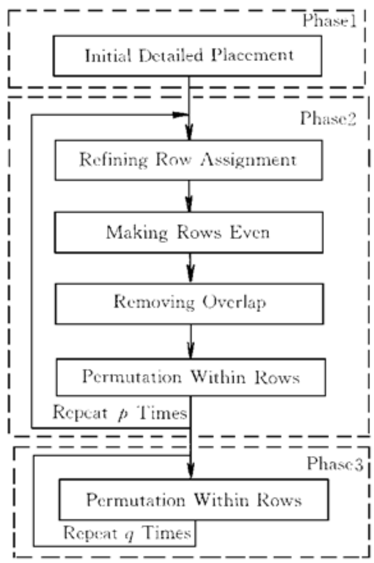

FIG. 1 Outline of FAME

3.2 Initial Detailed Placement

As the first phase of detailed placement, initial detailed placement (IDP) inherits the results of global placement and generates an initial legal detailed placement. Since we have several optimization steps in the following phases, the quality of final results does not depend on initial solution.

A simple partition and mapping heuristic is adopted: we partition the placement area hierarchically into $m \times n$ subareas until each subarea can hold hundreds of cells. We then map all the cells into these rooms according to the cells' original positions and the capacity of each room, with one cell belonging to one room exactly. Within each room, cells are first assigned to rows in order of increasing value in y coordinate. We begin the assignment from the bottom row, then the second row from the bottom, until all the cells are assigned. Then we merge all the segments of the same row from different subareas, and place cells in each row side by side from the left to the right in order of increasing value in x coordinate. The initial detailed placement is shown in Figure 2(a).

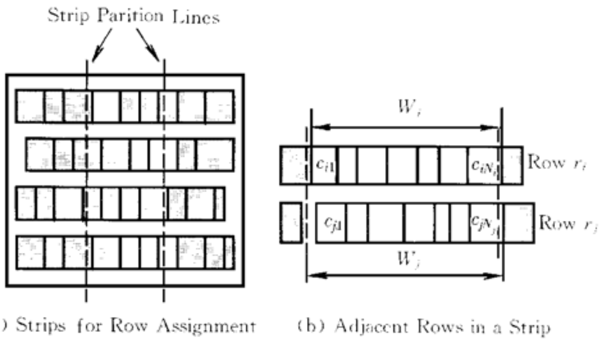

FIG. 2 Refined Row Assignment

3.3 Refining Row Assignment

Refining row assignment is the main step to optimize the assignment of cells into rows and reduce the total wirelength in y direction. In order to restrict the movement of cells in local areas, the whole placement area is divided into several vertical strips. As is shown in Figure 2(a), partition lines are given to the placement area and cells with the same filling patterns are in the same strip. Cells move or exchange with others only within the strips.

Using the following heuristic, we reassign the cells in the strip each time we choose two adjacent rows, r_i and r_j to operate as shown in Figure 2(b). Let W_i and W_j be the width of r_i and r_j in the strip respectively, $\{c_{i1}, c_{i2}, \cdots, c_{iN_i}\}$ be the N_i cells in r_i and $\{c_{j1}, c_{j2}, \cdots, c_{jN_j}\}$ the N_j cells in r_j. As for cell c_{ik}, let w_{ik} be its width and $\{n_{ik1}, n_{ik2}, \cdots, n_{ikT}\}$ be the T nets connected to it. The gain, g_{ik}, of putting c_{ik} into row r_j, is $g_{ik}=\sum_{s=1}^{T_{ik}}(wl_{iks}-wl'_{iks})$, where wl_{iks} is the connect length of net n_{iks} when c_{ik} is in row r_i (i.e. $y_{ik}=Y_i$) and wl'_{iks} is the length when c_{ik} is in row r_j (i.e. $y_{ik}=Y_j$). The gain, g_{jk}, of putting cell c_{jk} into row r_i can be determined in the same way. A positive gain means a reduction in wirelength when cell is moved to the other row. We try to move as many as possible cells with positive gain into their adjacent row to make the total gain maximum. The width constraints of each row can be satisfied at the same time.

Strips are processed one by one from the left to the right. Within each strip, adjacent rows are processed from the bottom pair to the top. When cells are moved between rows, only the y coordinates of them are modified, thus only wirelength in the y direction is optimized.

This algorithm is a kind of FM-Mincut heuristic in the sense that each time cells are partitioned into two sets, one for each row, and wirelength functions are used instead of the cut number in partitioning. The time complexity is $o(n)$ because of the computation of the moving gain for all cells, where n is the number of cells involved. FM-Mincut is faster than KL-Mincut since it moves two sets of cells for once, while KL-Mincut exchanges pairs of cells for many times and has a time complexity of $o(n^2)$ to complete the partition.

3.4 Making Rows Even

Because of the discreteness of cell widths, RRA itself cannot guarantee that the total width of cells in a row is not larger than the width of that row, as may result in the illegal detailed placement. We make rows even (RE) to adjust cell allocations between rows to eliminate the possible overflows.

RE contains two scans for each row. The first scan goes from the bottom row to the top. When row r_i with more cells than it can hold is obtained, we move several cells from r_i into row r_{i+1}. The limitation of strip in RRA is abandoned here and each cell in row r_i can be moved. Cell with higher gain after moving is chosen with higher priority. The gain of moving is same as in RRA. Cells don't move until the total width of cells in r_i is no larger than the width capacity. The second scan goes from the top row to the bottom. Similar to the first scan, except that cells are moved to the row below it if necessary. All possible overflows can be eliminated by RE, as long as the placement area is large enough.

3.5 Removing Overlap Within Rows

Cells are all assigned to a row after RRA. We remove the possible overlap between cells and make ready for the following permutation. This step is called removing overlap within rows (ORR). Since rows are located without overlap in the y direction, cells are also free from overlapping each other in the y direction.

A mapping method is used to remove the overlap between cells and keep the order of cells in x direction in each row. As for row r_i, $(0, X_B)$ is considered as the boundary of cell position in x coordinate for the sake of simplicity. We sort the set of cells in r_i, $\{c_{i1}, c_{i2}, \cdots, c_{iN_i}\}$ first, so that $x_{i1} \leqslant x_{i2} \leqslant, \cdots, \leqslant x_{iN_i}$. Let x'_{ik} be the new x coordinate after ORR of cell c_{ik}, and w_{ik} be its width. Cells are then located one by one from the left to the right. To cell c_{ik}, we set a left boundary lb_{ik} and a right boundary rb_{ik} as follows: $lb_{ik} = x'_{i(k-1)} + w_{i(k-1)}$ and $rb_{ik} = X_B - \sum_{t=k+1}^{N_i} \left\lceil \frac{w_{it}}{Rl} \right\rceil$; where Rl is a preset value between $\left\lceil \sum_{i=1}^{N_t} w_{it}/X_B \right\rceil$ and (1) Left boundary of the leftmost cell is 0, (i.e. $lb_{i0} = 0$) the right boundary of the rightmost cell is X_B(i.e. $rb_{iN_i} = X_B$).

The new position of cell c_{ik} is determined by

$$x'_{ik} = \begin{cases} lb_{ik} & \text{when } x_{ik} < lb_{ik} \\ rb_{ik} - w_{ik} & \text{when } (x_{ik} + w_{ik}) > rb_{ik} \\ x_{ik} & \text{otherwise} \end{cases} \tag{1}$$

Since $lb_{ik} \leqslant rb_{i(k-1)}$, we have got that $rb_{ik} - lb_{ik} \geqslant w_{ik}/Rl \geqslant w_{ik}$, thus x'_{ik} has a unique value in (1). It is obvious that $x'_{ik} \geqslant x'_{i(k-1)} + w_{i(k-1)}$, so the placement of such mapping is free from overlaps between cells. All rows are processed by OOR before cell permutation.

3.6 Permutation of Cells Within Rows

Permutation of cells within rows (PCR) is the main optimization step in phase 3. We try to change the order of cells in rows to make the total wirelength in x direction shorter. Rows are processed one by one. As for row r_i, we choose a bucket of m successive cells

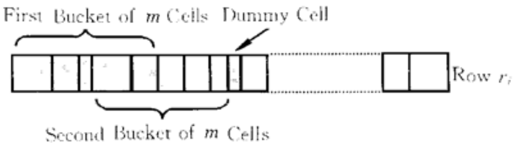

FIG.3 Permute Cells Within a Row

each time, as is shown in Figure 3. All possible orders of the m cells are checked and the one with the shortest wirelength is kept. Cells are placed side by side, so it is easy to derive the position of cells in the bucket from their order. If there are gaps between cells, we should fill them with movable dummy cells first. The bucket slides in the row from the left to the right, until all the cells in the row are processed. There are k cells in common for the two successive buckets to make the cells move freely in the row. For example, in Figure 3, the first bucket of cells contains the first to the fifth cell in r_i and the second bucket contains the fourth to the eighth, with two cells in common.

PCR is a simple but effective method. The enumeration-based methods can achieve better performance than Mincut heuristics when fewer than 10 cells are involved of the final placement[9]. Our experimental results also show that PCR reduces the total wirelength in x direction efficiently. Since all possible orders are enumerated for m cells, the time complexity is $o(m!)$ for each local permutation, which makes PCR time-consuming for a larger m. We will carry out some improvements in accelerating PCR in section 4.

4 Some Key Techniques in FAME

4.1 Two-Dimensional Minimization

RRA is used to minimize the total wirelength in y direction while PCR in x direction. In Depmin[6] RRA is applied in phase 2 and PCR in phase 3. This separated one-dimensional optimization may trap into local minimization and limit the placement quality. We mix them up and take the advantage of a two-dimensional optimization. The result is the new flow for phase 2 as shown in Figure 1. In the new flow, RRA and PCR are interlaced within one loop. Thus optimization in both directions are combined.

The interlacing of RRA and PCR improves the performance in the following ways: After carrying out PCR, cell orders in rows are changed. In consequence, the set of cells in each strip for RRA is also modified. This modification provides RRA with a larger space of solution, since cells are possibly put into other strips. On the other hand, cell movement in rows is also a kind of disturbance to RRA, because cells in each strip do not remain the same any longer between the repetition of RRA. Such disturbance helps Mincut heuristic to get out of local minimum and approach the better solution.

To verify this idea, we have compared the total wirelengths in y direction after RRA between two flows. One is the flow of FAME that interlaces RRA and PCR in phase 2, the other is the noninterlaced flow used in Depmin, with only RRA and RE in phase 2. The result of comparison to the test case C7 is shown in Figure 4. Though PCR does not reduce the wirelength in y direction itself, the mixture of RRA and PCR in the interlaced flow results in a 10 percent shorter in total wirelength in y direction. Experiments in other test cases also show such advantages of the interlaced flow.

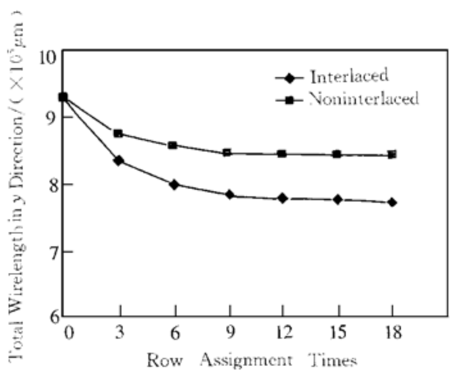

FIG. 4 Comparision of Total Wire Length in y Direction After RRA for Case C7

4.2 Speedup to Enumeration

PCR takes up 80 percent of the running time of FAME and the most time-consuming operation in PCR is involved in the calculation of total wirelength for a cell arrangement. Supposing n cells are chosen for permutation, a simple strategy of enumeration is used, all n cells may change their position when we got a new order. Therefore, the wirelength of all m nets connected can be changed. So m nets must be checked to calculate the total wirelength after each permutation. The time complexity of this operation is $o(m)$. In fact, each time we only exchange the order of two adjacent cells to generate a new order. For instance, once the order of n cells is $(c_1, c_2, c_3, \cdots, c_n)$, but the next time the order becomes $(c_2, c_1, c_3, \cdots, c_n)$. Only the order of c_1 and c_2 are exchanged at this time. Since cells are placed side by side, only the positions of c_1 and c_2 are modified, with those nets connected to c_1 and c_2 being checked to get the total wirelength after permutation. The time complexity of wirelength calculation is reduced to $o(\bar{m})$, where $\bar{m}$ is the average number of nets connected to one cell. A method is provided in Ref. [10] that exchanging only the order of two adjacent elements a time can generate all possible permutation of n elements. We apply the method to PCR and get a speedup to FAME.

The second refinement aims at a better trade off to the speed and performance. If n cells are chosen for every permutation in PCR, the time complexity is $o(n!)$ for all possible orders. But it is unacceptable when n goes higher than 5 in experiments. On the other hand, if n is too small, the permutations are limited by the local areas and the performance deteriorates. So there is a dilemma in choosing n. Our solution is that each time we choose n cells and change the order of cells in odd positions only. For example, 5 cells $\{c_1, c_2, c_3, c_4, c_5\}$ are chosen, the original order is $(c_1, c_2, c_3, c_4, c_5)$. During the permutation, only c_1, c_3 and c_5 are changed their order in position while c_2 and c_4 are always in the second and the fourth position respectively. In this way, it is possible to move the cells in a span of n cells within each permutation in time complexity of $o(n/2)$. Appropriate n is chosen through experiments to get a good trade off between speed and performance.

4.3 Extend to Cope with Blockages

Lots of VLSI designs contain preplaced macro blocks like memory or register groups. Forbidden areas may also be marked in the placement area. It's a practical constraint that makes cells be kept out of such forbidden areas and not overlap with the preplaced blocks, but detailed placement algorithms that can deal with such basic constraints are rarely found in the papers. We demand that forbidden areas should be determined before detailed placement and have fixed positions. The filling of fixed dummy cell are introduced to deal with the two constraints in common.

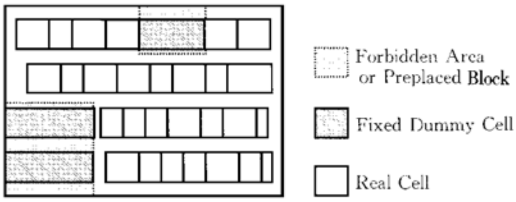

FIG. 5 Add Fixed Dummy Cells to Cover Forbidden Areas

As shown in Figure 5, dummy cells are filled to occupy all the areas where flat cells haven't entered in rows. To deal with the fixed cells, more considerations are added. In RRA and ER, fixed dummy cells do not move between rows. In ORR, fixed dummy cells leave many gaps between them, which can be used as placement sections. If there is no fixed dummy cells in a row, the whole row is a section itself. First we allocate cells to these sections in rows, and remove the overlaps between cells within each section. As for PCR, the bucket of cells is truncated when fixed dummy cells are met. In other words, fixed cells are not involved in permutation.

5 Experimental Results

We have implemented the basic idea in this paper along with all the improvements on it in a tool called FAME, in C on a SUN Enterprise 450 workstation running on Solaris. We ran FAME on a variety of testcases with 500 to 20000 cells, and compared the results with the famous programming- based tool called RITUAL and another Mincut-based tool called Depmin. RITUAL is a package with both global and detailed placements, but we use only the detailed placement part for comparison. Same global placement results and same wirelength estimation method are used in all the three tools. Some characteristics of the 8 testcases are tabulated in Table 1. The first 7 testcases are MCNC benchmarks with no blocks in design, while the last one, blk4, is an industrial testcase with 4 preplaced blocks. The results are listed in Table 2. RITUAL and Depmin cannot run on blk4 since they do not accept preplaced blocks.

Table 1 Characteristics of Test Example

Circuit Name	Number of Cells	Number of Nets	Number of Pads	Number of Blocks
C2	590	963	373	0
C5	1586	1887	301	0
C7	2150	2465	315	0
s13207	4267	5757	1490	0
s2	9906	11514	1608	0
s3	15545	17271	1726	0
acq	21854	21928	74	0
blk4	10044	10840	791	4

Compared with RITUAL, FAME runs 4 times faster in most cases and achieves a 5 percent shorter in total wirelength on average. As for large testcases like avq, FAME runs even 12 times faster that it. We notice that the CPU time increases almost linearly with the example size for FAME. Another comparison with Depmin shows that FAME runs at a speed comparable to the Depmin's while the total wirelength is 11 percent shorter in the final placement.

Table 2　Results Comparison Between Depmin, RITUAL and FAME

Circuit Name	Total Wire Length After Global Lacement /$10^5\mu$m	Depmin		RITUAL		FAME	
		Total Wire Length, Final /$10^5\mu$m	CPU Time /s	Total Wire Length, Final /$10^5\mu$m	CPU Time /s	Total Wire Length, Final /$10^5\mu$m	CPU Time /s
C2	4.599	4.482	5.5	4.245	23.1	3.954	6.8
C5	13.194	12.916	20.3	11.807	79.2	11.227	17.4
C7	19.265	17.616	36.5	16.44	85.9	16.034	22.7
s13207	70.178	67.733	40.9	64.503	241.7	63.001	48.3
s2	202.05	195.96	89.5	189.90	825.2	180.56	99.1
s3	290.94	275.81	139.6	283.25	1597.8	258.53	150.4
avq	107.96	124.52	160.7	106.17	3272.0	95.78	263.1
blk4	24.561	N/A	N/A	N/A	N/A	22.806	99.6

6　Conclusions

A detailed placement algorithm for standard-cell layout is presented. The algorithm adopts FM-Mincut heuristic and local enumeration for wirelength reduction in y and x directions respectively. A new flow that interlaces the operations in two directions is introduced to take the advantage of two-dimensional optimization. Using a new strategy of exchange in adjacence in enumeration, the time complexity of gain calculation is reduced. The incomplete permutation in enumeration also provides a better trade off between time and performance in enumeration. Results obtained from testcases demonstrate that FAME is superior to a programming based tool, RITUAL, both in speed and performance, and achieves much better performance than other heuristic based tools, such as Depmin, in speed.

References

[1] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, San Francisco, CA: Freeman, 1979.

[2] "The National Technology Roadmap for Semiconductors", Semiconductor Industry Association, U.S.A, 1997.

[3] J. Parkhurst *et al*., "SRC Physical Design Top Ten Problems", in Proc. ISPD'99, 1999, 55—58.

[4] A. Srinivasan, K. Chaudhary and E. S. Kuh, "RITUAL: A Performance Driven Placement Algorithm," IEEE Trans. on Circuits and Systems Ⅱ, 1992, **39**(11): 825—840.

[5] J. M. Kleinhans, G. Sigl, F. M. Jonhannes and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", IEEE Trans. on CAD, 1991, **10**(3): 356—365.

[6] X. Yuan, C. Qiao and X. Hong, "Detailed Placement Based on Mincut", in Proc 10th National CAD/CG Conference, Guilin, China, 1998, 10.

[7] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure and Its Partitioning Graphs", Bell Syst. Tech. J., 1970, 291—307.

[8] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", in ACM/IEEE Proc. 19th DAC, 1982, 175—181.

[9] A. E. Caldewll, A. B. Kahng and I. L. Markov, "Optimal Partitioners and End-case Placers for Standard-Cell Layout", in Proc. ISPD'99, 1999, 90—94.

[10] K. Lu, "Combinatorics", Qinghua University Press, 2nd Ed., 1991.