國立交通大學

資訊科學與工程研究所

碩 士 論 文

以電路重新配置改進全域繞線器的可繞度和線長

Improving Routability and Wire-Length of Global Routing with Circuit Replacement

研 究 生：呂建宏

指導教授：李毅郎　教授

中 華 民 國　九 十 七　年 八 月

以電路重新配置改進全域繞線器的可繞度和線長

Improving Routability and Wire-Length of Global Routing with Circuit Replacement

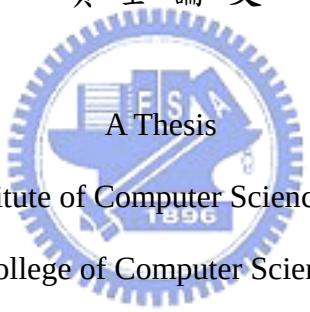
研 究 生：呂建宏　　Student：Chien-Hung Lu

指導教授：李毅郎　　Advisor：Yih-Lang Li

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文



A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

August 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年八月

以電路重新配置改進全域繞線器的可繞度和線長

研究生 : 呂建宏　　指導教授: 李毅郎 博士

國立交通大學 資訊科學與工程研究所

摘要

在實體設計中，配置器扮演的相當重要的角色。它決定了電路線長的最低界線，且配置結果會對全域繞線器的可繞度和線長有很大的影響。為了拉近配置器和繞線器的關係，我們結合配置器和繞線器來幫助我們配置器可以更準確的估計繞線器的線長。這個作品提出兩個階段來改善繞線器的可繞度和減少繞線器的線長，分別為:減少線長和減少擁擠。在減少線長的階段中，我們提出了三個方法：減少線長的移動，加強化的區域重新安排，和以雙族匹配方法重新佈置元件。在減少擁擠的階段中，我們提出了兩個方法:元件排列和避免擁擠的移動來。實驗結果說明了我們用 IBMv2 的測試檔改善了兩個配置器:Dragon 和 mPL-R 的線長分別改善了 4.2 百分比和 2.3 百分比。另外，我們改善了兩個配置器預先繞線的滿載，分別改善了 6.4 百分比和 6.6 百分比。

Improving Routability and Wire-Length of Global Routing with Circuit Replacement

Student: Chien-Hung Lu **Advisor: Dr. Yih-Lang Li**

Institute of Computer Science and Engineering

National Chiao Tung University

Abstract

Placement plays an important role in physical design. It determines the lower bound of circuit wirelength and its result significantly influences circuit routability. In order to close the gap between placement and routing, we integrate global routing and placement to help our placer perform more accurate estimation of routed wirelength. This work presents two stages to improve routability and wirelength of global routing including wirelength minimization stage and congestion reduction stage. In wirelength minimization stage, we present threes methods, namely wirelength-reduced cell shifting, enhanced local re-ordering, and cell rearrangement by bipartite matching. In congestion reduction stage, we present cell sorting based congestion reduction and congestion-avoided cell shifting. Experimental results demonstrate that our placer improves total wirelength by 4.2% and 2.3% as compared to Dragon and mPL-R on IBMv2 benchmarks. Furthermore, our placer improves overflow of pre-routing by 6.4% and 6.6% as compared to Dragon and mPL-R.

Acknowledgement

I am deeply grateful to my advisor, Dr. Yih-Lang Li for his continuous guidance, support, and ardent discussion throughout this research. His useful suggestions help me to complete the thesis. Also I express my sincere appreciation to all classmates in my laboratory for their encouragement and help. Especially I want to thank my senior classmate Ke-Ren Dai for his assistance and direction in this work.

This thesis is dedicated to my parents and my families for their patience, love, encouragement and long expectation.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

As semiconductor manufacturing technology advances to the nanometer scale and VLSI design complexity continues to increase, interconnection delay dominates the circuit delay. Placement has become a critical stage in VLSI design flow because placement determines the lower bound of total wirelength. Therefore, a good placement result improves the circuit performance. In recent years, the wirelength is not the only objective in placement. Since the position of every cell is fixed in the placement and will not be altered in the routing stage, if cells are placed too closely for wirelength minimization, the global router may produce many overflows and the detailed router may yield routing violations because of the limited routing resource. Thus, routability must also be considered in the placement stage and wirelength minimization with increased routability is the objective of modern placement.

Half-perimeter wirelength (HPWL) is used to estimating the wire-length of a net in the placement stage. However, if the number of pins is greater than three, the accuracy of HPWL decreases significantly. ROOSTER [1] illustrates that Steiner-tree wirelength (StWL) is more accurate than HPWL as the estimation metric of routed wirelength (rWL) in placement.

Traditionally, placement and routing are performed independently. Recent researches have substantially progressed in the performance and speed of global routing. IPR [2] considered global routing during placement to know total wirelength and congestion information to improve total wirelength and routability in placement stage. In recent years, some methods were presented for congestion reduction. In [3], the white spaces allocated into congestion region to reduce congestion called White

Space Allocation (WSA). RUDY [4] presented a fast and accurate routing demand estimation. Its routing demand estimate is based on rectangular uniform wire density. Jiang et al. [5] presented a net overlapping removal technique to reduce congestion in global placement. In our placer, we use routing information to know real congestion region from router and solve it.

This work presented three techniques for routed wirelength minimization, i.e. reduced-wirelength cell shifting, enhanced local re-ordering, and cell rearrangement by bipartite matching. Reduced-wirelength cell shifting shifts the cell to its “optimal region”. Enhanced local re-ordering solves local mistake of locally horizontal wirelength. Finally, cell rearrangement by bipartite matching finds the better position of several cells using bipartite matching in large range. This study also proposed cell sorting based congestion reduction and congestion-avoided cell shifting to reduce routing congestion.

The rest of paper is organized as follows: Section 2 presents preliminary. Section 3 gives an overview of our work. Section 4 presents the wirelength minimization technique. Section 5 describes the proposed congestion reduction technique. Section 6 summarizes the experimental results. Conclusions are finally drawn in Section 7.

Chapter 2
Preliminaries

2.1 Overview of our Global Router

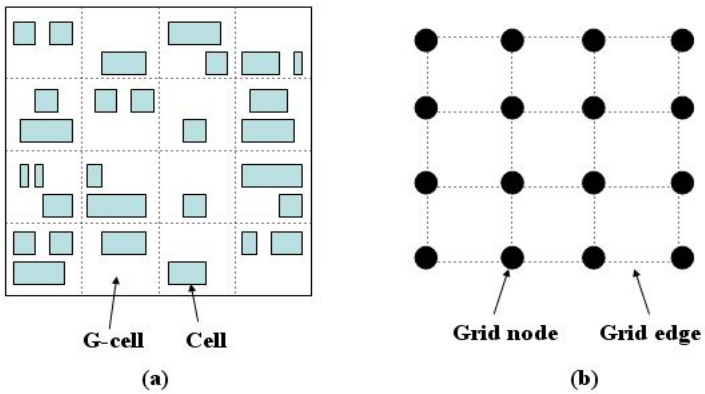


Figure 1. G-Cells and their Corresponding Grid Graph.

A routing region can be partitioned into an array of global cells or bins (G-cells) and modeled by a grid graph G(V, E), where every node of the graph denotes a global cell, and every edge (global edge) is termed the adjacency of the related global cells of its two end nodes. The capacity of an edge indicates the number of routing tracks that can be accommodated across the abutting boundary. Figure 1 (b) shows a one-layer grid graph of the circuit containing 4x4 G-cells in Figure 1 (a). The metrics of routability can be measured based on the overflow of all global edges and wire length of every net. Few overflow global edges and short wire lengths imply high routability. The increasing number of overflow global edges raises the difficulty of routing, reducing the wire length lowers the congestion of some global edges.

Figure 2 gives a brief overview of our global router. Firstly we will decompose each multiple pin net into two-pin net by minimal spanning tree (MST). Then we will complete most two-pin net routing in pre-routing stage. Finally, rip-up and reroute stage will try to resolve overflow as much as possible.

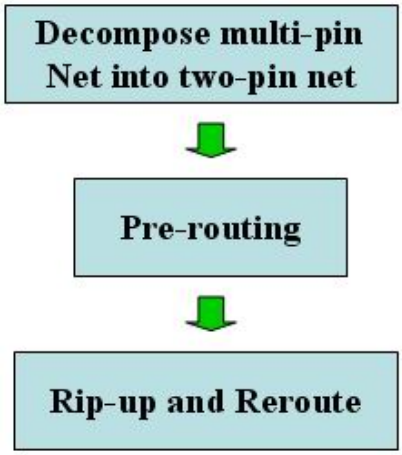


Fig. 2. An Overview of our Global Router

2.2 Optimal region

Goto [6] first presented the idea using median to find optimum placement of modules in electrical circuit layout. *FastDP* [7] presented the idea of "optimal region" using the similar concept. The concept of optimal region is that if a cell is placed in its optimal region while the other cells are fixed, the wirelength is optimal.

For any cell i, we search all nets (denoted as N_i) connecting to it. Then we find boundary of bounding boxes that are formed with cells except cell i in each net. For each net n in N_i, we find boundaries of its bounding box $x_l[n]$, $x_r[n]$, $y_l[n]$, $y_u[n]$ denoted as left, right, lower, upper boundaries respectively. Then we sort the x series ($x_l[n_1]$, $x_r[n_1]$, $x_l[n_2]$, $x_r[n_2]$, …) and y series ($y_l[n_1]$, $y_u[n_1]$, $y_l[n_2]$, $y_u[n_2]$, …) and find the medians of x series and y series, where the x_{opt} and y_{opt} are the medians of x series and y series. So the optimal position is (x_{opt}, y_{opt}). If number of elements in the x and y series are even, the optimal position would be an optimal region. Otherwise it would be a line or a point.

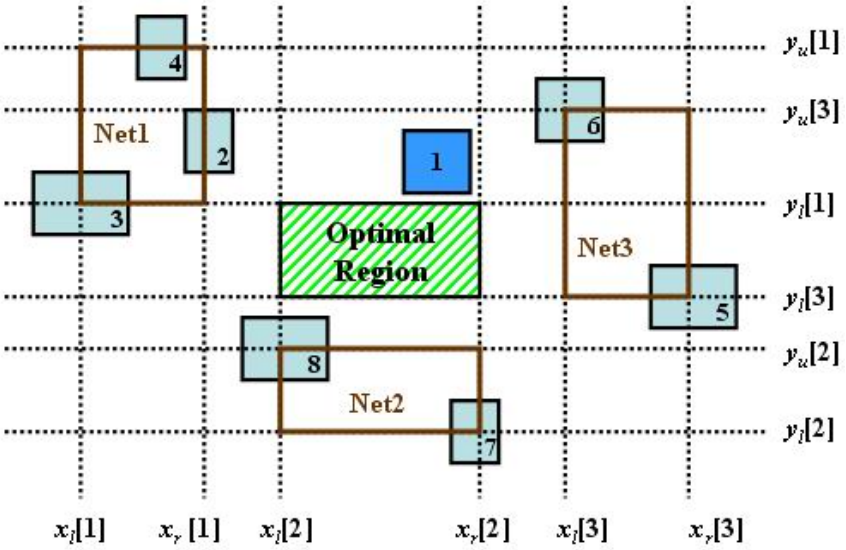


Fig. 3. Optimal Region

Figure 3 shows an example of optimal region. There are three nets that connect to cell 1: Net1 contains cells 1, 2, 3, and 4; Net2 contains cells 1, 7, and 8; Net3 contains cells 1, 5, and 6. We record positions of cells excepts cell 1 to form x series ($x_l[1]$, $x_r[1]$, $x_l[2]$, $x_r[2]$, $x_l[3]$, $x_r[3]$) and y series ($y_l[1]$, $y_u[1]$, $y_l[2]$, $y_u[2]$, $y_l[3]$, $y_u[3]$). Finally, we sort x and y series and get the optimal region. The shadowed region with slanted lines is optimal region of cell 1.

Chapter 3
Overview of Our Placer

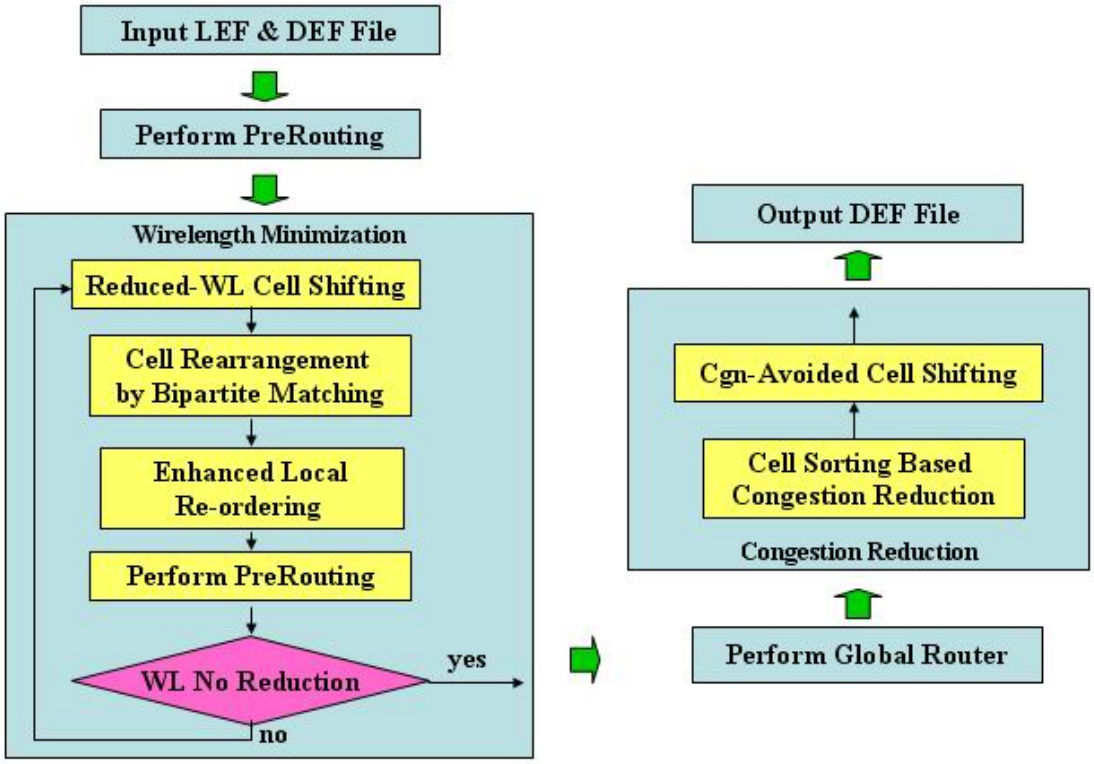


Fig. 4. Design flow of our placer

Figure 4 presents the flow of the proposed placer. In the beginning, our placer reads LEF and DEF files to get information of cells and net. In Prerouting stage, each net is decomposed into a set of two-pin nets using minimal spanning tree (MST) in global router. In Global Router stage, we route every nets and get congestion information using global router. In wirelength minimization stage, we repeat three methods: reduced-wirelength cell shifting, cell rearrangement by bipartite matching, and enhanced local re-ordering until routed wirelength has no significant improvement. After wirelength reduction stage, we perform global router again to get the new congestion of each grid edge. Finally, cell sorting based congestion reduction and congestion-avoided cell shifting are performed for reducing congestion.

Chapter 4
Wirelength Minimization

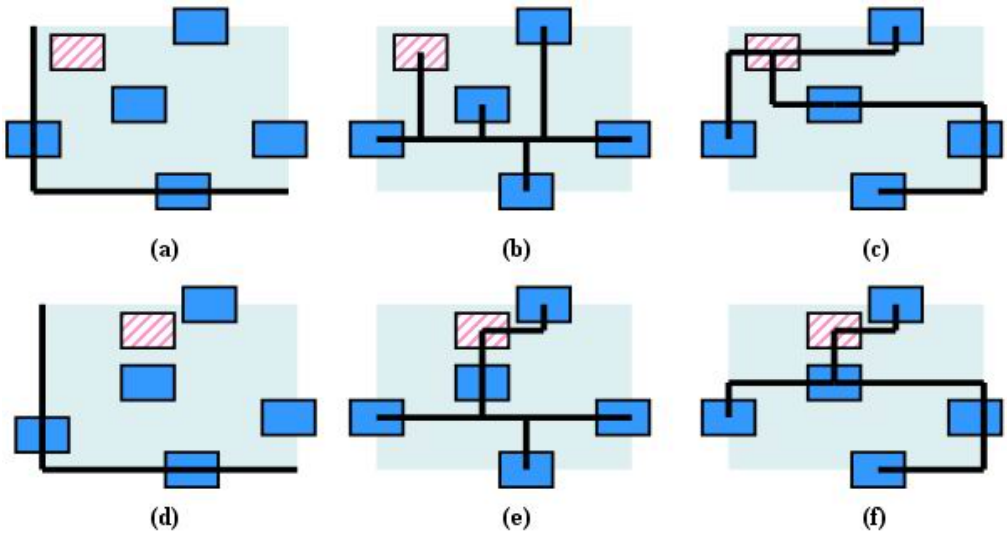


Fig. 5. Difference of different estimations. (a) HPWL, (b) Steiner WL, and (c) RMST of original circuit for a six-pin net. (d) HPWL, (e) Steiner WL, and (f) RMST of new circuit for a six-pint net.

Wirelength minimization is a main object of the placement stage. Traditionally, half-perimeter wirelength (HPWL) is chosen to be the estimation of routed wirelength. In [1], it presented that rectilinear Steiner minimal tree (RSMT) is more accurate thar HPWL for estimating routed wirelength (rWL). Since global router becomes efficient and effective in recent years, we can integrate routing and placement to get real routed information. In order to reduce rWL, we use following methods to achieve this goal. In the beginning, our global router employ minimal spanning tree (MST) to decompose multi-pin nets into 2-pin nets. Then we will try to reduce the wirelength of two-pin nets by our placer.

Figure 5 (a), (b), and (c) show the different wirelength using different estimates: HPWL, RSMT, and rectilinear MST (RMST). Then we shift the cell with slated line to right and observe change of wirelength with different estimates. The result shows in

Figure 5 (d), (e), and (f): HPWL doesn't be changed and both wirelength of RSMT and RMST is reduced. We can conclude that our method for rWL minimization is more accurate than HWPL. In this work, we use three techniques to reduce the wirelength of two-pin nets: reduced-wirelength cell shifting, enhanced local re-ordering, and cell rearrangement by bipartite matching.

4.1 Reduced-Wirelength Cell Shifting

The fundamental concept of reduced-wirelength cell shifting is to shift cell to its "optimal region". We use two-pin nets relations that get from global router and reduce wirelength of two-pin nets in order to reduce rWL.

Algorithm : Reduced-Wirelength Cell Shifting

```
Variables: priority queue of cells
Initialize priority queue using distance between cell and its optimal region as cost
1  while (priority queue not empty)
2      Dequeue a cell i
3      while (cell i is marked shifted)
4          Unmark cell i
5          Enqueue i using distance between cell and its optimal region as cost
6          Dequeue a cell i
7      end
8      if (cost of cell i < 0.1)
9          break
10     end
11     Find i's optimal region and cells in its optimal region
12     Calculate total free sites in each row of optimal region
13     Find grid number of extended optimal region
14     for (each cells j in optimal region of cell i)
15         While (optimal region not to extend)
16             if (TFS_j >= width_i ||
                 width_j <= width_i && TFS_j + width_j <= width_i)
17                 Calculate and record HPWL of two-pin nets of cells that be shifted
18                 break
19             end
20             else
21                 Extend optimal region
22             end
23         end
24     end
25     Select maximal gain of HPWL between original and new circuit
26     if (maximal gain > 0)
27         Update positions and two-pin nets of cells that be shifted
28         Enqueue i using distance between cells and its optimal region as cost
29         Mark cell that be shifted except i
30     end
31     else
32         Enqueue i with (original cost * 0.5)
33     end
34 end
```

Fig. 6. Reduced-Wirelength Shifting Algorithm

Our reduced-wirelength cell shifting algorithm is shown in Figure 6. In this

algorithm, we use priority queue to decide the order of cells. In the beginning, we initialize the priority queue using distance between cell and its optimal region as cost and unmark all cells. For each cell *i* that pop from priority queue, it will be checked if be shifted before iterations until selecting an unmarked cell (line 3-7). If cost of cell *i* is less than 0.1, the while loop will be stopped and finish reduced-wirelength cell shifting algorithm (line 8-10).

For a cell *i*, even if we find its optimal region, there may have many choices to move to free sites or swap with the other cell in its optimal region. When we find optimal region of cell *i*, we would try all possible free sites and all cells that can be make space for cell *i* in the optimal region. Then we calculate gain that is difference between original and new HPWL of two-pin nets of cells which position has been changed. We select maximal gain, and use its result to shift cells. First we define some variables: TFS_*j* represents total free sites in cell *j* placed row of cell *i* shifting range, width_*i* represents width of cell *i*, and width_*j* is width of cell *j*. For the first case, if TFS_*j* is larger than width of cell *i* "width_*i*", cell *i* can be shifted in this free sites. Otherwise, if "width_*j*" is less than "width_*i*" and TFS_*j* plus width_*j* is less than width_*i*. For the cells that conform among two cases, we will calculate HPWL of two-pin nets and record gain of HPWL of two-pin nets (line 16-19). The results of all possible cells that can be shifted are predicted and we select the solution that has maximal gain for HPWL as final result. Then we use final result to shift cells and update information of cells and two-pin nets if maximal gain is larger than zero. Then, shifted cells except cell *i* will be marked and cell *i* will be put into priority queue using new distance between it and its optimal region (line 26-30). If maximal gain is less than zero, the placement will not be change and cell *i* will be put into priority queue by decreasing its priority (line 32).

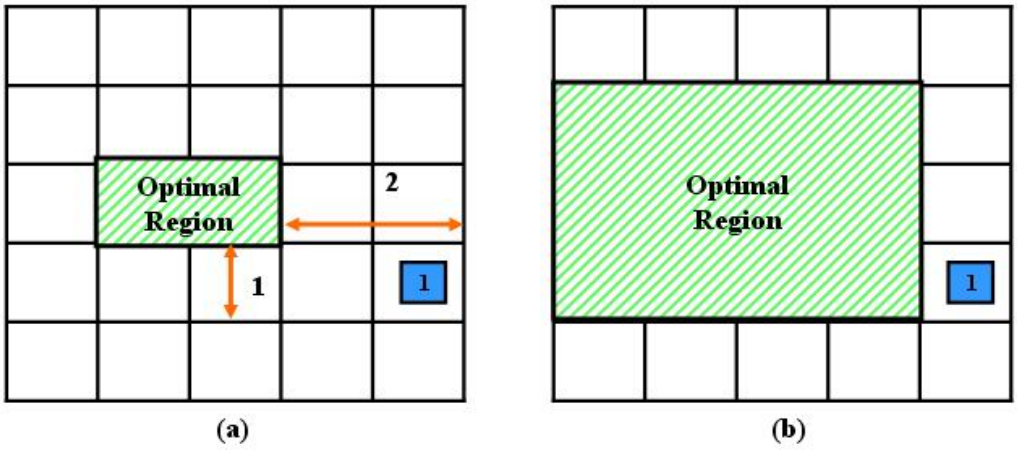


Fig. 7. Example of Extending Optimal Region. (a) Number of Extending Grids; (b) Result of Extending Optimal Region.

Even if we can not shift the cell into its optimal region, we still want the cell to be placed near its optimal region as much as possible. To achieve this goal, we will do optimal region extending. In the beginning, we firstly decide the number of grids when extending optimal region. The number of grids is less difference between cell's bin and its optimal region in vertical and horizontal. So, if the cell isn't shifted, its optimal region would be extended in four directions until exceeding its bin (line 15-23). Figure 7 shows an example of extending optimal region. Figure 7(a) displays the difference between cell's bin and its optimal region in vertical and horizontal is 1 and 2 respectively. The result of extending optimal region is shown in Figure 7(b).

4.2 Enhanced Local Re-ordering

FastDP [7] presented a local re-ordering technique that refines local horizontal wirelength in detail placement. In local re-ordering, it selected three consecutive cells formed a segment in each row from left to right and permuted these cells. Then, the best wirelength result of all orders was selected and distributed cells evenly in the segment.

```
Algorithm: Enhanced Local Re-ordering
1  for y = 1 to n
2      Perform ReorderingInRow(y, L)
3  end
4  for y = 1 to n
5      Perform ReorderingInRow(y, R)
6  end
7 Function: ReorderingInRow(y, dir)
8  num_reorde = 3
9  if dir == L
10     for x = 1 to m - (num_reorder - 1)
11         Select num_reorder cells from x to x + (num_reorder - 1)
12         Record wirelength of all permutation
13         Select the best result best_WL
14         if (best_WL < original_WL)
15             Update positions and two-pin nets of cells that be shifted
16         end
17     end
18 end
19 if dir == R
20     for x = m to m - (num_reorder - 1)
21         Select num_reorder cells from x to x + (num_reorder - 1)
22         Record wirelength of all permutation
23         Select the best result best_WL
24         if (best_WL < original_WL)
25             Update positions and two-pin nets of cells that be shifted
26         end
27     end
28 end
```

Fig. 8. Enhanced Local Re-ordering Algorithm

In this work, we have enhanced the local re-ordering technique to further improve its quality. In local re-ordering, it only performed local re-ordering from left to right in each row. The cell might successively shift to right if the distance of its best position is larger width of segment so it might shift to right with several iterations. However, if the best position of a cell is in the left side and distance between the best position and the cell is larger than width of segment, it will not be shifted to the best position. In order to handle the problem, we will not only do local re-ordering from left to right, but also do local re-ordering from right to left. The pseudo code of enhanced local re-ordering is shown in Figure 8. The algorithm begin in line 1-6 with doing cell reordering for each row, and it includes main function “ReorderInRow” that performs the local re-ordering form left to right and from right to left in each row. In our implementation, we select three consecutive cells within segment for each iteration. First, we perform “ReorderInRow” (line 9-18) from left to right in each row.

We in turn select three consecutive cells every time and do permutation to know all possible order. Then we distribute these cells within segment and calculate HPWL of two-pin nets of these cells for each order. Finally, we select the best wirelength “best_WL” and compare with the original wirelength “original_WL”. If “best_WL” is less than “original_WL”, we update the position and two-pin nets of cells that be shifted. Second, we perform “ReorderInRow” (line 19-28) from right to left in each row. It is similar the above method, the difference of both is just that we select consecutive three cells form right to left in each row.

4.3 Cell Rearrangement by Bipartite Matching

In each iteration of reduced-wirelength cell shifting, only one cell can be shifted for wirelength minimization. Then, the objective of enhanced local re-ordering is to improve the horizontal wirelength locally in each row. In our work, cell rearrangement by bipartite matching is presented to solve several cells rearranging in the larger-scale range.

The wirelength minimization problem can be modeled as a bipartite graph G=(V, E) with vertex classes X and Y, where each vertex x_i represents a cell, vertex y_j represents a position, and the weight on $e_{i,j}$ represents the HPWL of two-pin nets of cell x_i is placed position y_j. Figure 9 shows the flow of cell rearrangement by bipartite matching. First, we must select a segment in each row form left to right and rearrange cells in the segment. For each segment, only independent cells that have no two-pin nets connected with other cells in the segment would be considered. Because if there is no such constraint, the distance of two-pin nets between cells in the segment would be hard to decide. The placed position of cells is width of the segment evenly divided the size of independent cells. Then the independent cells are decided by bipartite matching for minimizing HPWL of two-pin nets of cells.

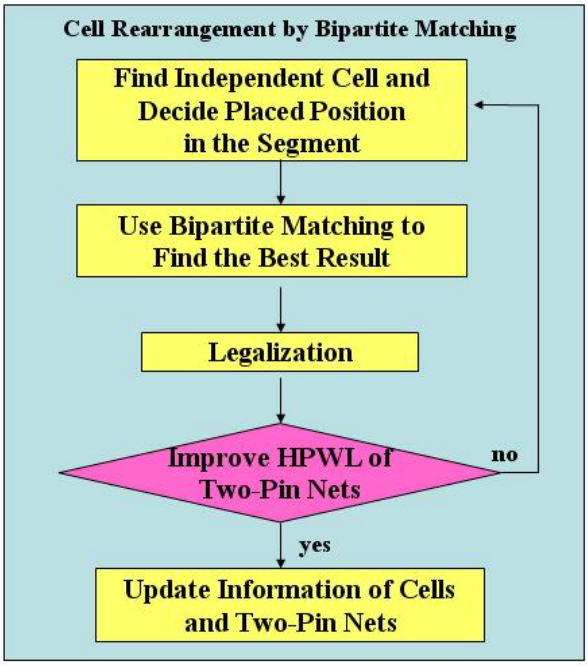


Fig. 9. Design flow of Cell Rearrangement by Bipartite Matching

However, position of cells after bipartite matching may produce overlap, so we must legalize cells in the segment. In legalization stage, we use Abacus [8] to legalize cells. After legalization, if total HPWL of two-pin nets of circuit is improved, information of cells and two-pin nets are updated. If total HPWL is not improved, the circuit will not be changed.

Chapter 5
Congestion Reduction

In previous chapter, we have presented wirelength reduction technique. However, if we only consider wirelength minimization, it may produce lower routability result. In this chapter, we will present two congestion reduction techniques to try to produce routable circuit: cell sorting based congestion reduction and congestion-avoided cell shifting. In cell sorting based congestion reduction, we use two-pin nets difference as score to sort cell position to solve horizontal congestion. In congestion-avoided cell shifting, we will shift the cells in the congested bins to further reduce congestion.

5.1 Cell Sorting Based Congestion Reduction

In this section, we present cell sorting based congestion reduction algorithm that can fast reduce horizontal congestion to improve routability. In our cell sorting based congestion reduction algorithm, we will select cells in each congestion bins as segment, and we will compute difference of two-pin nets (diff_TPN) for each cell in the segment. The definition of diff_TPN is the number of two-pin nets that connect to its right side subtracts the number of connection to the left side. In order to define right two-pin nets (right_TPN) and left two-pin nets (left_TPN), we must set left and right boundary of the segment first. The left boundary is position of the leftest cell and right boundary is position of the most right cell. If the position of the connecting cell of the two-pin net exceeds right boundary, then this two-pin net is a right_TPN. If the position of the connecting cell of the two-pin net exceeds left boundary, we call such two-pin net a left_TPN. Then we can compute diff_TPN using number of the right two-pin nets subtracts number of the left two-pin nets. Finally, we sort diff_TPN in the increasing order and decide order of all cells in segment by sorting result. Then we

separate each neighbor cells using average free sites in the segment.

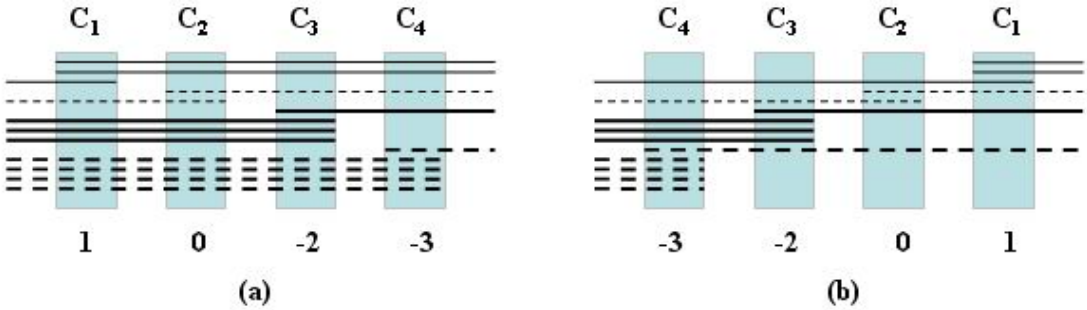


Fig. 10. Illustration of Cell Sorting Based Congestion Reduction method. (a) Original circuit before Cell Sorting; (b) New circuit after Cell Sorting.

Figure 10 shows an example of cell sorting based congestion reduction. Figure 10(a) is the original circuit, and the new circuit after cell sorting based congestion reduction is shown in Figure 10(b). In Figure 10(a), the two-pin nets of C_1, C_2, C_3 and C_4 are thin real lines, thin dashed lines, bold real lines and bold dashed lines respectively. The cell C_1 has three two-pin nets include two right_TPNs and one left_TPN. The diff_TPN of C_1, C_2, C_3 and C_4 are 1, 0, -2 and -3. Then we sort this number of sequence and get order of cell be C_4, C_3, C_2 and C_1. The result shows in Figure 10(b) and we can observe that the congestion of new circuit is less than the congestion of original circuit.

Cell sorting based congestion reduction only considers horizontal wires and solves horizontal congestion. However, vertical congestion may be increased by sorting. In order to avoid such situation, we will check the number of pins in the global bins after sort. Figure 11 displays the flow of cell sorting based congestion reduction. Firstly, we will select a congested segment do cell sorting based congestion reduction. Then we will sort the cell by diff_TPN. If number of pins exceeds maximal pin number in each bin, the sorting result will be rejected. Finally, the information of cells and two-pin nets would be updated.

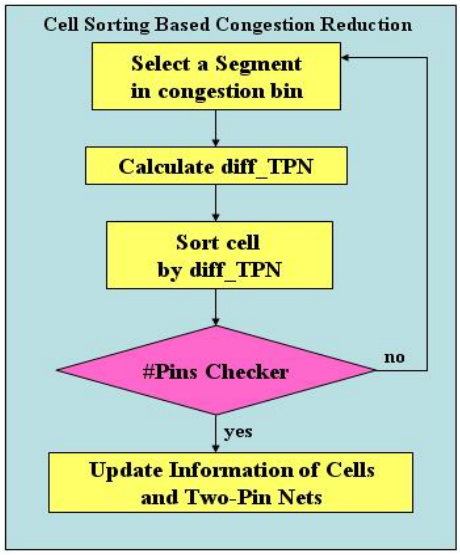


Fig. 11. Design flow of Cell Sorting Based Congestion Reduction

5.2 Congestion-Avoided Cell Shifting

The goal of congestion-avoided cell shifting is to reduce congestion by shifting the cells in the congestion region to non-congestion region. This method has two important issues: how to select cells and where to be placed. In section 5.2.1, we will describe simulated evolution-based cell selection. In section 5.2.2, cell shift with avoiding congestion algorithm will be introduced.

5.2.1 Simulated Evolution-based Cell Selection

SILK [9] is a simulated evolution-based router, it presented a rip-up and re-route technique for detailed routing. The simulated evolution technique scores each creature in the current population, and determines the survival rate of each creature in the next generation. SILK scores each net in generation using routing violation and path quality and normalized the scores to 0.1-0.9. Then SILK generates a random number between 0.0 and 1.0. The normalized score of nets are greater than their random number are be marked. After comparing all nets, the marked nets are put into queue. The order of nets in the queue decreases from higher normalized score to lower normalize soccer. Thus, SILK avoids falling local optimal, it let nets with lower cost have chance to be ripped and rerouted.

In our work, we use congestion, wirelength, and number of pins to generate cells in congestion region as follow:

$$score = \alpha \cdot [\alpha' \frac{\sum_{\forall n_i} cgn(n_i)}{\#two-pin\ nets} + \beta' \frac{\sum_{\forall cgn(n_i)>avg(cgn)} [cgn(n_i) - avg(cgn)]^2}{\#cgn(n_i)}] + \beta \cdot [\frac{total\ WL}{\#two-pin\ nets}] + \gamma \cdot \#pins \quad (1)$$

,where α , β , γ , α' , and β' are user-defined constant, $\sum_{\forall n_i} cgn(n_i)$ is total congestion of two-pin nets of cell *i*, then congestion of the two-pin net is average of congestion of bins that the two-pin net go through, $\#two-pin\ nets$ is total number of two-pin nets, $\sum_{\forall cgn(n_i)>avg(cgn)} [cgn(n_i) - avg(cgn)]^2$ is square of difference between $cgn(n_i)$ and $avg(cgn)$, $avg(cgn)$ is average of congestion of all two-pin nets of cell *i*, $total\ WL$ is HPWL of all two-pin nets, and $\#pins$ is number of pins of cell *i*. In Equation (1), the first part is average congestion and variation. The variation helps our placer to detect which cell may have larger congestion of two-pin nets when they have the same average congestion.

Figure 12 displays the pseudo code of simulated evolution-based cell selection. In the beginning, we calculate score of cells by Equation (1), all score are normalized to 0.1-0.9 and generator generates the random between 0.0 and 1.0. If the cells with normalized score that are larger than their random number will push queue. Finally, we order cells in queue from higher normalized score to lower normalized score. The cell with higher normalized score will be shifted first. In next section, we describe the function: CongstionAvoidShifting(cell *i*).

```
Algorithm : Simulated Evolution-based Cell Selection
Input: A set of cells in congestion bins C
for each cell i
    Calculate score_i by equation (1)
end
for each cell i
    Normalize score_i into norm_score_i between 0.1 and 0.9
end
for each cell i in C
    Generate a random number r between 0.0 and 1.0
    if (norm_score_i > r)
        mark cell i as re-shifting net and push i into
queue
    end
end
Sort cells in queue by norm_socre
for each cell I in queue in ordering
    Perform CongestionAvoidedShfiting(cell i)
end
```

Fig. 12. Simulated Evolution-based Cell Selection Algorithm

5.2.2 Cell Shifting with Avoiding Congestion

The objective of congestion-avoided cell shifting is to reduce congestion by shifting the cells in the congestion region to non-congestion region. However, where should the cells be replaced remains a problem. In this section, we will show the cost function we use to decide where cell be shifted first. Then, we will describe the function CongestionAvoidedShifting(cell i).

For each cell *i*, the cost function for congestion-avoid cell shifting is defined as follows:

$$cost = \alpha \cdot \frac{1}{distance(org\_pos, new\_pos)} + \beta \cdot [\alpha' \cdot diff\_left + \beta' \cdot diff\_right + \gamma' \cdot diff\_up + \omega' \cdot diff\_down] \quad (2)$$

,where α , β , α' , β' , γ' ,and ω' are user-defined constants, $distance(org\_pos, new\_pos)$ is the distance between original position and new position of cell *i*, $diff\_left$, $diff\_right$, $diff\_up$, and $diff\_down$ represent difference of two-pin nets in left-direction, right-direction, up-direction, and down-direction respectively.

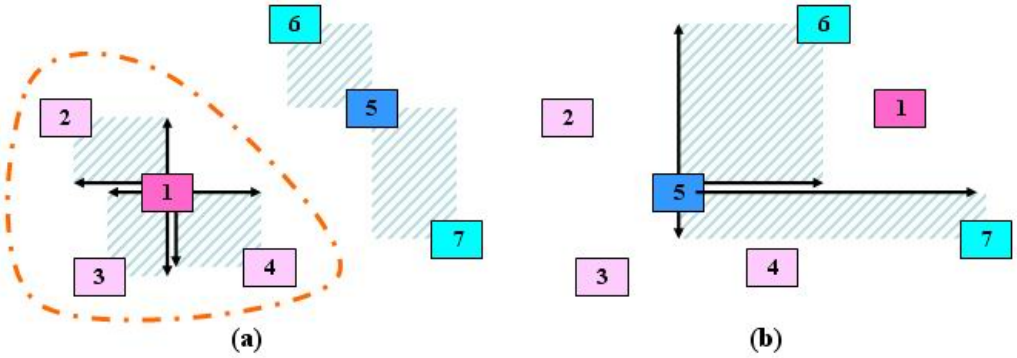


Fig. 13. Example of Difference of Two-Pin Nets

Figure 13 represents how to calculate $diff\_left$, $diff\_right$, $diff\_up$, and $diff\_down$. In Figure 13(a), cell 1 has two-pin nets relation with cell 2, cell 3, and cell4; cell 5 has two-pin nets relation with cell 6 and cell 7 respectively. The dashed line shows congestion region and the other region is non-congestion region. The shadowed region with slanted lines is bounding box of cells that have two-pin net relation. In each shadowed region with slated line, we use two arrows to represent possible direction when routing between cells with two-pin net relation. So, cell 1 has three bounding box and six arrows of bounding box from cell 1 to other cells. If we want to know change of difference of two-pint nets after swapping cell 1 and cell2. The cell 5 will swap with cell 1 and calculate difference of two-pin nets of cell 5 using new position that shows in Figure 13(b). Finally, $diff\_left$, $diff\_right$, $diff\_up$, and $diff\_down$ can be calculate through subtracting number of arrows of Figure 13(b) from number of arrows of Figure 13(a) in left-direction, right-direction, up-direction, and down-direction respectively. In this example, $diff\_left$, $diff\_right$, $diff\_up$, and $diff\_down$ equal 2, -1, 0, and 1 respectively. In this cost function, we can calculate between two cells or a cell and a space. If we calculate cost between a cell and a space, the difference of four directions are produced arrows of four directions. The cell with higher cost can be swapped with a space or a cell and reduce congestion with higher opportunity because this cell with less two-pin nets

connected with it.

```
Function: CongsetionAvoidedShifing(cell i)
 1  Find non-congested bins in where cell i is shifted
 2  Calculate total free sites in each row of non-congested bins
 3  for (each cells j in non-congested bin of cell i)
 4       if (TFS_j >= width_i ||
           width_j <= width_i && TFS_j + width_j <= width_i)
 5           Calculate and record gain of cell i by equation (2)
 6      end
 7  end
 8  Select maximal gain
 9  if (maximal gain > 0)
10      Update position and two-pin nets of cells that be shifted
11      Rip-up and reroute two-pin nets of shifted cells
12      Update information of bins and C if congested situation is changed
13  end
```

Fig. 14. CongestionAvoidedShifting Function

Figure 14 shows the pseudo code of the proposed CongestionAvoidedShifting Function. TFS_*j* represents total free sites in cell *j* placed row of cell *i* shifting range, width_*i* represents width of cell *i*, and width_*j* is width of cell *j*. If free sites are enough or cell *i* can swap with cell *j*, the gain is calculated (line 4-6). After calculating and recording all gain, we select maximal gain. If maximal gain is larger than zero, we update position and two-pin nets of shifted cells and rip-up and reroute two-pin nets of shifted cells. Finally, we update information of bins (line 9-13).

Chapter 6
Experimental Results

The proposed placer was implemented in C/C++ language on AMD Opteron 2.6GHZ with 16GM memory. IBMv2 benchmarks [9] were used in our experiment. The characteristics of these benchmarks, including number of cells, number of nets, core utilization, white space and routing layers, are shown on Table I..

Table I Characteristic of IBM Version 2 Benchmarks

name	cells	nets	rows	core(row) utiliztion	white space	routing layer
ibm01-easy	12028	11753	132	85.12%	14.88%	4
ibm01-hard	12028	11753	130	88%	12%	4
ibm02-easy	19062	18688	153	90.42%	9.58%	5
ibm02-hard	19062	18688	149	95.28%	4.72%	5
ibm07-easy	44811	44681	233	89.95%	10.05%	5
ibm07-hard	44811	44681	246	95.30%	4.70%	5
ibm08-easy	50672	48230	243	90.03%	9.97%	5
ibm08-hard	50672	48230	236	95.16%	4.84%	5
ibm09-easy	51382	50678	246	90.24%	9.76%	5
ibm09-hard	51382	50678	240	95.12%	4.88%	5
ibm10-easy	66762	64971	321	90.22%	9.78%	5
ibm10-hard	66762	64971	313	95.08%	4.92%	5
ibm11-easy	68046	67442	281	90.11%	9.89%	5
ibm11-hard	68046	67442	273	95.33%	4.67%	5
ibm12-easy	68753	68376	347	85.22%	14.78%	5
ibm12-hard	68753	68376	388	90.06%	9.94%	5

We compare our result with several state-of-the-art academic tools, including Dragon 3.0.1 [10] and mPL-R [3]. The results are given in Table II–III. In Table II, we show results including total run time and routed wirelength(rWL). For rWL, Our Placer is better than Dragon about 4.2% after replacing by Our Placer. We compare the same items with mPL-R in Table III. The rWL of Our Placer yields 2.3% improvement compared to mPL-R. For overflow of pre-routing, Our Placer is better

than Dragon 6.4% and better than mPL-R 6.6% after congestion reduction.

Table II Comparison of Placement Results of Dragon and Our Placer

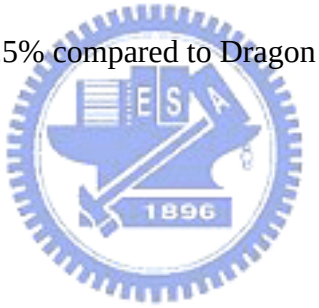
name	Dragon time(sec)	Dragon rWL(μm)	Our Placer rWL(μm)	Bef Cgn-reduction Overflow	Aft Cgn-reduction Overflow	Our Placer time(sec)
ibm01-easy	377.95	113791	**107258**	**1**	**1**	63.05
ibm01-hard	376.81	111071	**105884**	46	**31**	60.38
ibm02-easy	639.97	311777	**297428**	252	**251**	122.69
ibm02-hard	726.82	296855	**286877**	290	**228**	131.75
ibm07-easy	1028.09	651857	**626987**	**194**	204	273.51
ibm07-hard	1023.85	655502	**634786**	138	**137**	293.81
ibm08-easy	2542.12	748973	**716534**	135	**128**	337.94
ibm08-hard	2542.02	699039	**674864**	**117**	118	352.92
ibm09-easy	1945.93	616969	**591777**	**9**	19	313.29
ibm09-hard	1887.1	584243	**563714**	13	**8**	296.91
ibm10-easy	3479.25	1151624	**1103093**	93	**87**	510.65
ibm10-hard	4796.69	1112552	**1074329**	327	**320**	515.78
ibm11-easy	1978.26	883490	**846269**	72	**62**	398.57
ibm11-hard	3034.49	847257	**820545**	**103**	111	426.08
ibm12-easy	3360.41	1658950	**1575344**	657	**606**	648.03
ibm12-hard	3286.04	1539642	**1473632**	948	**879**	620.97
ratio	-	**1.042**	**1**	**1.064**	**1**	-

Table III Comparison of Placement Results of mPL-R and Our Placer

name	mPL-R time(sec)	mPL-R rWL(μm)	Our Placer rWL(μm)	Bef Cgn-reduction Overflow	Aft Cgn-reduction Overflow	Our Placer time(sec)
ibm01-easy	143.082	108039	**104651**	123	**118**	61.25
ibm01-hard	124.932	102843	**100153**	11	**7**	60.92
ibm02-easy	361.023	283954	**277950**	52	**48**	123.76
ibm02-hard	272.332	271112	**266986**	**82**	92	123.69
ibm07-easy	970.409	605564	**594187**	233	**218**	275.09
ibm07-hard	758.693	585232	**575386**	427	**370**	281.46
ibm08-easy	962.772	690856	**673528**	99	**92**	326.4
ibm08-hard	1060.74	666116	**652088**	**33**	**33**	336.43
ibm09-easy	828.371	558430	**545001**	**13**	**13**	304.27
ibm09-hard	843.824	536410	**525653**	65	**64**	331.05
ibm10-easy	1575.58	1056650	**1030586**	46	**39**	508.04
ibm10-hard	1345.67	1020905	**1000867**	**47**	56	512.81
ibm11-easy	1548.12	825040	**807187**	36	**27**	426.68
ibm11-hard	1177.92	777948	**763510**	61	**55**	423.75
ibm12-easy	2267.51	1496605	**1455516**	**492**	494	620.31
ibm12-hard	1932.74	1450010	**1416069**	871	**798**	638.58
ratio	-	**1.023**	**1**	**1.066**	**1**	-

Chapter 7
Conclusions

In this paper, we propose a placer that improves routed wirelength and routability. Our work presents three methods including reduced-wirelength cell shifting, enhanced local re-ordering, and cell rearrangement by bipartite matching to minimize routed wirelength and two methods including cell sorting based congestion reduction and congestion-avoided cell shifting to reduce congestion of global routing in placement stage. Experimental results shows our placer can improve routed wirelength on IBMv2 benchmark with other placer placement results. The routed wirelength (rWL) of our placer yield 4.3% and 2.5% compared to Dragon and mPL-R respectively.



Chapter 8
Bibliography

[1] J. A. Roy, J. F. Lu, and I. L. Markov. “Seeing the Forest and the Trees: Steiner Wirelength Optimization in placement,” *IEEE Trans. on Computer-Aided Design*, vol. 26 no. 4, pp. 632-644, April 2007.

[2] Min Pan and Chris Chu. “IPR: an integrated placement and routing algorithm,” In *Proc. Design Automation Conf.*, pages 59 – 62, 2007.

[3] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden, “Routability-driven placement and white space allocation,” In *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design*, 2004, pp. 394–401.

[4] P. Spindler and F. M. Johannes. “Fast and accurate routing demand estimation for efficient routability-driven placement,” In *Proc. of DATE*, pages 1226–1231, Nice, France, 2007.

[5] Zhe-Wei Jiang, Bor-Yiing Su, and Yao-Wen Chang . “Routability-driven analytical placement by net overlapping removal for large-scale mixed-size designs,” In *Proc. Design Automation Conf.*, pages 167-172, 2008.

[6] S. Goto. “An Efficient Algorithm for the Two-Dimensional Placement Problem in Electrical Circuit Layout,” In *IEEE Transitions on Circuits and Systems*, Vol. CAS-28, No. 1, pp. 12-18, 1981.

[7] M. Pan, N. Viswanathan and Chris Chu. “An Efficient and Effective Detailed Placement Algorithm,” In *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design*, pp. 48-55, 2005.

[8] Peter Spindler, Ulf Schlichtmann, and Frank M. Johannes. “Abacus: fast legalization of standard cell circuits with minimal movement,” In *Proc. Intl. Symp. On*

Physical Desgin, pages 47-53, 2008.

[9] Y.-L. Lin, Y.-C. Hsu, and F.-S. Tsai. “SILK: A simulated evolution router,” *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems*, pages 1108-1114, Oct. 1989.

[10] X. Yang, B.-K. Choi, and M. Sarrafzadeh. “Routability-driven white space allocation for fixed-die standard-cell placement,” In *International Symposium on Physical Design*, pages 42-47, 2002.