12-1-2003

On Improving Recursive Bipartitioning-Based Placement

Chen Li

Cheng-Kok Koh

ON IMPROVING RECURSIVE BIPARTITIONING-BASED PLACEMENT

CHENCHEN LI
CHENG-KOK KOH

TR-ECE 03-14
DECEMBER 2003

PURDUE UNIVERSITY

SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, IN 47907-2035

On Improving Recursive Bipartitioning-Based Placement*

Chen Li and Cheng-Kok Koh
School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University West Lafayette, IN 47907-1285
{li35,chengkok}@ecn.purdue.edu

December 1, 2003

*This research is partially supported by SRC project 947.1.

Contents

List of Figures

List of Tables

Abstract

We present an improved partitioning-based placement tool in terms of the half-perimeter wirelength. In contrast to placement tools that exploit white spaces to reduce the routing congestion, our placement tool exploits white spaces to reduce the total wirelength. In particular, we generate white spaces during the legalization step and treat white spaces as pseudo cells during the detailed placement step. On IBM benchmarks, our results are better than those of Feng Shui [1] and Capo [4] by 1.8% and 11.1%, respectively, on the average. Our results are comparable to those of Dragon [17]. On the synthetic benchmarks PEKO, our placement tool outperforms all other placement tools except mPL [6]. On another suite of synthetic benchmarks PEKU with global nets, our results are the best among all the existing public academic placement tools.

1 Introduction

Automatic placement, a critical step in the physical design flow of integrated circuits, has been well studied in the past. The emergence of high-quality multi-level hypergraph partitioning algorithms [3, 12] has spawned several good partitioning-based placement tools, such as Capo [4], Feng Shui [18] and Dragon [16]. Capo incorporates the min-cut partitioner MLPart [3] and end-case minimum-wirelength placers [5]. Feng Shui uses an iterative deletion method [13] to specify the terminal propagation information for the partitioner hMetis [12] to get better results. Dragon [16] combines the quadrisection subroutine from hMetis [12] and a simulated annealing iterative improvement approach. The results of these placement tools indicate that the solution quality of the partitioning-based method is comparable to that of other placement techniques, such as force-directed method [11] and analytical method [14].

In this paper, we investigate a few issues that are commonly encountered in the partitioning-based placement flow. In the partitioning step, the sequence of cut directions has been improved in [19] to reduce the wirelength; by simply comparing the ratio of region height to width with a fixed aspect ratio, a near-optimal cut direction can be determined. However, it is not clear how this fixed aspect ratio can be pre-determined. We propose a *dynamic aspect ratio scheme* to solve this problem. Most partitioning-based placement tools have an assumption that a horizontal cut line has to coincide with a horizontal row boundary. In [1], "fractional cut" was proposed to allow a horizontal cut line to lie within a row. To make "fractional cut" more effective, we propose a *pre-partitioning technique* and improve the *terminal propagation method*.

In the legalization step and detailed placement step, most placement tools either do not allow white spaces to be generated in the legalized placement [16, 1], or only exploit white spaces to reduce the routing congestion [17]. In these tools, white spaces have seldom been exploited to reduce the wirelength. We introduce *white spaces* in our greedy legalization step and treat white spaces as pseudo cells in our detailed placement step. Allowing white spaces to be generated and to be permuted with cells is one of the main contributions of our work.

Our tool produces placements with the total wirelengths shorter than those of Feng Shui and Capo and similar to those of Dragon on IBM benchmarks. On synthetic PEKU benchmarks, our tool produces total wirelengths shorter than other placement tools. On PEKO benchmarks, our placement tool outperforms all other tools except mPL.

2 Overview of our placement flow

Similar to most existing partitioning-based placement tools, our tool performs placement in three steps: the partitioning step, the legalization step and the detailed placement step. In the partitioning step, the circuit netlist is divided recursively by the partitioning algorithm hMetis, and then sub-netlists are assigned to certain chip regions until every partition contains exactly one cell. We adopt "fractional cut" [1] in our partitioning step, and also enhance it by dynamic aspect ratios and using pre-partitioning for an improved terminal propagation method. In the legalization step, all cells are aligned to rows and overlaps between cells are removed by a greedy algorithm. Simultaneously, white

spaces are generated between cells. In the detailed placement step, the wirelength are further reduced by searching locally the minimum wirelength of a moving window containing a small number of cells. That is accomplished by permuting real cells together with white space pseudo cells inside the window. These pseudo cells can be compacted, expanded, deleted and generated to make a permutation legal and the wirelength shorter.

3 Improved partitioning step

Currently, most partitioning-based placement tools use straight cut lines to divide the chip region. There are various unintentional constraints that restrict the effectiveness of this "divide-and-conquer" approach. One is the cut direction. Some placement tools determine the cut direction by comparing the ratio of width and height of the current region with a fixed aspect ratio [19]. We relax this constraint by using "dynamic aspect ratio". Another is the cut location. They restricted the horizontal cut line to be located on a row boundary. Similar to [1], we relax this constraint by using "fractional cut". We further propose a pre-partitioning technique and improve the terminal propagation method to make "fractional cut" more effective.

3.1 Dynamic aspect ratio

In [19], a fixed and empirical aspect ratio is used to determine the cut directions throughout the entire partitioning process. We propose a simple dynamic aspect ratio scheme to make every final partition shape match the cell shape as much as possible. Recall that every final partition contains exactly one cell. We define the aspect ratio of a partition as the width over the height of this partition and the average aspect ratio of cells contained in a partition as the average width of these cells over the height of a cell. If the aspect ratio of a partition is larger than the average aspect ratio of cells contained in this partition, we cut this partition vertically; otherwise, we cut it horizontally.

3.2 Pre-partitioning and terminal propagation

By using fractional cut, we allow partitioning algorithm to produce more unbalanced partitions in order to reduce the cut size. We use hMetis as our bipartitioning subroutine and allow each partitioned area to range from 40% to 60% of the total area .

Terminal propagation was first proposed in [10]. It is an effective method to reduce the wirelength. But partitioning order may make some terminals indefinite. This can be illustrated by the example in Fig 1. After the first level of bisection, locations of cells are updated to centers of their respective partitions. Now if we want to cut region A into two regions horizontally at the next level, some cells in region B should be considered as terminals of region A. However, should these terminals be propagated to the upper region or the lower region of A?

We solve this problem partially by using a "pre-partitioning" technique. We pre-partition all regions at the same level to make cells have more specific locations. We first pre-partition regions A and B into regions C', D'(not shown

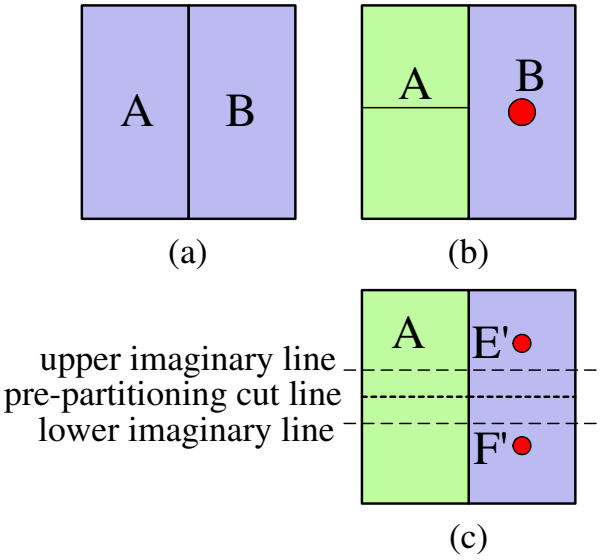

Figure 1: Improved terminal propagation

in this figure) and E', F', respectively. We then partition region A again based on terminals propagated from regions E' and F'. Note that if region B is cut vertically during pre-partitioning, those terminals are still indefinite.

As we allow unbalanced bipartitioning, the exact location of the cut line is determined by the area ratio of two sub-netlists. We draw two imaginary lines at 40% and 60% of the region A, between which the cut line may be located. Only cells in region B that are connected to region A and are located above the upper imaginary line or below the lower imaginary line are treated as terminals to the upper region or the lower region of A respectively. The remaining cells are considered as floating terminals and are not propagated to the sub-regions of region A.

4 Legalization and detailed placement

Some placement tools [16, 1] do not allow white spaces to be generated in the legalized placements. For example, Dragon2000 [16] packs all cells to the left in legalized placements. In the detailed placement step, when swapping two cells in different rows, locations of other cells in these two rows might need to be adjusted to remove the possible overlap and white space, which restricts the effectiveness of the local search. Our tool allows spaces in legalized placements. In contrast to some placement tools that exploit white spaces to reduce the routing congestion, [17] for example, our tool exploits white spaces to reduce the wirelength by permuting white spaces with cells in the detailed placement step.

Algorithm: Legalization

Input placement after partitioning step
Output legalized placement

sort cells in the increasing order of their x-coordinates.
for each row r **do**
  $xstart[r]$=leftmost x-coordinate of a row
end for
for each cell n in the sorted cell list **do**
  for each row r **do**
    ry =y-coordinate of this row
    if $(n.x - n.w/2 < xstart[r])$ /*overlap*/
      $cost[r] = f(n, xstart[r] + n.w/2, ry)$
    else /*no overlap*/
      $cost[r] = f(n, n.x, ry)$
    end if
  end for
  $r_{tgt} = \arg\min\{cost[r]\}$
  ry =y-coordinate of row r_{tgt}
  move cell n to $(xstart[r_{tgt}] + n.w/2, ry)$ or (x, ry)
  update $xstart[r_{tgt}]$
end for

Figure 2: Our legalization algorithm

4.1 Legalization step

We use a simple greedy algorithm outlined in Fig 2 in our legalization step, which may generate white spaces between cells. We sort all cells in the increasing order of x-coordinates of their centers. Starting from the leftmost cell, we determine one cell at a time the row that the cell belongs to by minimizing the "relocation cost". "Relocation cost" is defined as the cost incurred in moving a cell from the original location to a new location. It can be either the increase in wirelength due to this move or the distance between the original location and new location. We use the former cost function in this work. When a cell is assigned to a row, it is not necessary for this cell to be located right next to another cell. The cell keeps its x-coordinate, if that does not result in an overlap. Otherwise, we place it right next to the last legalized cell in this row. This approach naturally generates spaces between cells.

In Fig 2, $n.x$ and $n.w$ denote the x-coordinate and width of cell n, respectively. $cost[r]$ denotes the relocation cost for moving this cell to row r. The function $f(n,x,y)$ denotes the relocation cost of moving cell n from its original location to position (x,y).

4.2 Detailed placement step

In the detailed placement step, every white space in the chip area is first divided into pseudo cells, each with a width that is equal to the minimum width of all cells. We then choose a window that contains six cells, including real cells

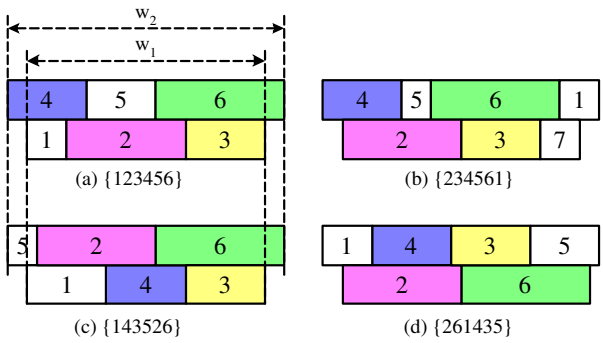

Figure 3: Detailed placement of cells in a window with spaces. Cell 1 and 5 are spaces and others are real cells.

and pseudo cells, and swap cells in this window to locally optimize the wirelength. This window covers adjacent cells in either one row or two rows. We slide the window over the entire chip from left to right, and from bottom to top, overlapping with the previous window by three cells. If a window covers cells only in one row, every permutation of these cells does not generate an overlap. We now present how white space pseudo cells can be permuted with real cells in a two-row window.

Fig 3 shows a two-row window containing six cells labeled 1 through 6, where cell 1 and cell 5 are pseudo cells. We define the widths available for the first row and the second row as w_1 and w_2, respectively. For every permutation of these six cells $\{\pi_1\pi_2\pi_3\pi_4\pi_5\pi_6\}$, we partition it into two sub-sequences $\{\pi_1 \dots \pi_i\}$ and $\{\pi_{i+1} \dots \pi_6\}$. Note that there are only seven ways to partition this permutation into two sub-sequences. If the total widths of *real* cells in two sub-sequences are not larger than w_1 and w_2 respectively, this partitioning is legal; otherwise, this partitioning is illegal. An illegal partitioning is discarded. In each permutation and partitioning, pseudo cells can be compacted, expanded, deleted and generated in different ways to occupy the available widths. We exhaustively search for a legal partitioning of a permutation such that the wirelength is minimized.

Fig 3(a) shows the original six cells and widths available for two rows. Permutation $\{123456\}$ in Fig 3(a) and its partitioning $\{123\}$ and $\{456\}$ correspond to the original placement. Permutation $\{234561\}$ in Fig 3(b) has only one legal partitioning, $\{23\}$ and $\{4561\}$. Under this legal partitioning, cells are placed as in Fig 3(b), where a pseudo cell labeled 7 is generated and pseudo cells 1 and 5 are compacted. Fig 3(c) shows another permutation and its legal partitioning $\{143\}$ and $\{526\}$. Note that this permutation has another legal partitioning $\{1435\}$ and $\{26\}$. Fig 3(d) shows a permutation whose partitionings are all illegal.

Note that cell locations may not conform to site location constraints after all these permutations. To legalize the placement, at the end of the detailed placement step, we snap all cells to placement sites.

5 Experimental results

IBM-place 2.0 easy benchmarks are converted from ISPD98 [2] partitioning benchmarks by mapping cells to commercial standard cell library by the authors of [17]. PEKO and PEKU benchmarks are constructed by the authors of [7] and [9] to study the optimality, scalability and stability of current placement tools.

We evaluate our placement tool using these benchmarks. Our placement results are verified to be legal by placement utilities from UMichigan [15] and center-to-center half-perimeter wirelengths are measured. In this work, as we focus on the quality of the partitioning-based method, we have not optimized the run-time. However, our placement tool still has shorter run-time than Dragon. We show our best results of five runs. Some results of other placement tools are average of five runs. Note that newer versions of some placement tools may improve their results. Also note that some tools, such as Capo and Dragon, are congestion-driven. Therefore, comparison with these tools may not be fair.

5.1 Impact of white spaces

To evaluate the impact of white spaces in our placement flow, we compare our legalization and detailed placement steps with those of Feng Shui on IBM-easy benchmarks. Both Feng Shui and our placement tool read in two sets of partitioning results, one is the results of the partitioner of Feng Shui, the other is the results of our partitioner.

Table 1 shows that our combination of legalization and detailed placement is about 1.3% better than that of Feng Shui 2.0. It also shows that our legalization step and detailed placement step always maintain or reduce the wirelength of partitioning results. Noting that our legalization algorithm is simple, we conclude that considering white spaces during legalization and detailed placement contributes to these improvements. Although white spaces increase the wirelength of our legalized placement, they allow more legal partitionings of permutations to be generated in the detailed placement step, which helps the detailed placement step to find a smaller wirelength of cells contained in a window. This is the reason why white space can reduce the wirelength in our placement flow.

We measure the amount of the space in the legalized placement and the final placement of benchmark ibm01. The total white spaces, which exclude the spaces located to left or right of each row, occupy only 1.1% of chip area.

5.2 Results on IBM/PEKO/PEKU benchmarks

Wirelength results of our placement tool, Feng Shui 2.0, Capo 8.6 and Dragon 2.23 on IBM benchmarks are shown in Table 2. Our placement tool outperforms Dragon on four of IBM benchmarks. They tie on the average. Our results are better than those of Feng Shui by 1.8% on the average. Our results are consistently better than those of another partitioning-based placement tool Capo by 11.1%.

Table 3 shows wirelength results of our placement tool, Feng Shui 2.0, capo 8.6, Dragon 2.20 and mPL 2.0 [6] on PEKO benchmarks. Table 4 and Table 5 show wirelength results of our placement tool, Feng Shui 2.0, Capo 8.5,

Table 1: Comparison of legalization and detailed placement steps between our placement tool and Feng Shui 2.0. Both tools read in partitioning results of our partitioner and Feng Shui 2.0 partitioner. All values are scaled by 10^8.

bench-marks	WL of our partitioner	WL after legalization and detailed placement steps		WL impr%	WL of Feng Shui 2.0 partitioner	WL after legalization and detailed placement steps		WL impr%
		Feng Shui 2.0	our placement			Feng Shui 2.0	our placement	
ibm01	0.508	0.513	0.504	1.4	0.513	0.515	0.505	1.9
ibm02	1.45	1.45	1.43	1.4	1.49	1.49	1.47	1.3
ibm07	3.28	3.29	3.25	1.2	3.29	3.29	3.25	1.2
ibm08	3.52	3.53	3.48	1.4	3.59	3.58	3.53	1.2
ibm09	3.00	3.01	2.96	1.7	2.99	2.99	2.94	1.4
ibm10	5.76	5.77	5.70	1.2	5.59	5.60	5.54	1.1
ibm11	4.42	4.43	4.37	1.4	4.65	4.64	4.59	1.2
ibm12	7.78	7.79	7.72	0.9	7.76	7.78	7.72	0.8
Average				1.3				1.3

Table 2: Wirelength results on IBM 2.0 easy benchmarks. Results of other tools are from [1]. All values are scaled by 10^8.

benchmarks	our placement WL	Feng Shui 2.0		Capo 8.6		Dragon 2.23	
		WL	impr%	WL	impr%	WL	impr%
ibm01	0.50	0.52	3.8	0.57	12.3	0.51	2.0
ibm02	1.43	1.50	4.7	1.60	10.6	1.44	0.7
ibm07	3.25	3.30	1.5	3.71	12.4	3.31	1.8
ibm08	3.48	3.60	3.3	3.89	10.5	3.39	-2.7
ibm09	2.96	3.02	2.0	3.31	10.6	2.96	0
ibm10	5.70	5.66	-3.6	6.34	10.1	5.61	-1.6
ibm11	4.37	4.48	2.5	4.89	10.6	4.43	1.4
ibm12	7.72	7.74	0.3	8.76	11.9	7.60	-1.6
Average			1.8		11.1		0.0

Dragon 2.20, mPG 1.0 [8] and mPL 1.2 on PEKU benchmarks. In these three tables, QR (quality ratio) is defined as the wirelength of a placement result over its optimal wirelength for PEKO or its upper bounds for PEKU.

For PEKO benchmarks, our results are 8.5% better than those of Feng Shui, 11.7% better than those of Capo, 25.2% better than those of Dragon, and 9.4% worse than those of mPL. For PEKU benchmarks, our placement tool obtains the best wirelength results among all the state-of-the-art placement tools. Unlike mPL, our placement tool consistently obtains good wirelengths with different percents of non-local nets.

6 Conclusion

In this paper, we present an improved partitioning-based placement tool. While we follow the same placement flow as most existing partitioning-based placement tools, we improve the partitioning step by dynamic aspect ratios, pre-partitioning, and improved terminal propagation. We also propose a simple greedy legalization algorithm, which allows white spaces to be generated between cells. Our detailed placement step incorporates white spaces to reduce

Table 3: Wirelength results on PEKO benchmarks. Results of other tools are from [1]. All values are scaled by 10^6.

bench-marks	Optimal WL	our placement		Feng Shui 2.0		Capo 8.6		Dragon 2.20		mPL 2.0	
		WL	QR	WL	QR	WL	QR	WL	QR	WL	QR
Peko01	0.814	1.19	1.46	1.25	1.55	1.29	1.59	1.46	1.80	1.10	1.36
Peko02	1.26	1.91	1.52	2.09	1.66	2.03	1.61	2.43	1.93	1.76	1.40
Peko03	1.50	2.25	1.50	2.62	1.75	2.66	1.77	2.93	1.95	2.05	1.37
Peko04	1.75	2.65	1.51	2.82	1.61	3.12	1.78	3.87	2.21	2.31	1.32
Peko05	1.91	2.80	1.47	3.01	1.58	3.16	1.65	3.79	1.98	2.57	1.35
Peko06	2.06	3.12	1.51	3.44	1.67	3.57	1.73	4.35	2.11	2.78	1.35
Peko07	2.88	4.46	1.55	4.91	1.71	5.07	1.76	6.24	2.17	3.95	1.37
Peko08	3.14	4.85	1.54	5.25	1.67	5.57	1.77	6.79	2.16	4.99	1.59
Peko09	3.64	5.66	1.55	5.98	1.64	6.47	1.78	7.72	2.12	4.76	1.31
Peko10	4.73	7.24	1.53	7.87	1.66	8.00	1.69	8.49	1.79	6.59	1.39
Average			1.51		1.65		1.71		2.02		1.38

Table 4: Wirelength results on PEKU benchmarks. Other results except for Feng Shui are from [9]. All values are scaled by 10^6.

bench-marks	Upper bound WL	our placement		Feng Shui 2.0		Capo 8.5		Dragon 2.20		mPG 1.0		mPL 1.2	
		WL	QR	WL	QR	WL	QR	WL	QR	WL	QR	WL	QR
1%Peku01	1.25	1.71	1.37	1.71	1.37	1.99	1.59	2.09	1.67	2.13	1.71	2.41	1.93
1%Peku05	3.31	4.21	1.27	4.45	1.34	5.26	1.59	5.54	1.68	5.53	1.67	7.48	2.26
1%Peku10	10.6	13.4	1.26	14.1	1.33	16.4	1.55	14.5	1.37	18.0	1.70	27.8	2.63
1%Peku15	33.9	41.7	1.23	42.6	1.26	50.2	1.48	47.8	1.41	55.6	1.64	96.7	2.85
1%Peku18	41.9	50.9	1.21	52.7	1.26	59.6	1.42	60.2	1.44	68.2	1.63	121	2.88
Average			1.27		1.31		1.53		1.51		1.67		2.51

the wirelength locally. Our legalization step and detailed placement step always maintain or reduce the wirelength of the partitioning results. Our results are 1.8% better than those of Feng Shui 2.0 and 11.1% better than those of Capo 8.6 and comparable to those of Dragon 2.23 on IBM benchmarks. Our results on PEKO are only worse than mPL and better than the others and our results on PEKU are the best among all the public placement tools.

7 Acknowledgment

We thank the authors of [1] for providing Feng Shui 2.0.

References

[1] Ameya Agnihotri, Mehmet Can Yildiz, Ateen Khatkhate, Ajita Mathur, Satoshi Ono, and Patrick H. Madden. Fractional cut: Improved recursive bisection placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 307–310, 2003.

[2] C. J. Alpert. The ISPD98 circuit benchmark suite. In *Proc. Int. Symp. on Physical Design*, pages 80–85, 1998.

Table 5: Wirelength results on Peku01 with different percents of non-local nets. Other results except for Feng Shui are from [9]. All values are scaled by 10^6.

bench-marks	Upper bound WL	our placement		Feng Shui 2.0		Capo 8.5		Dragon 2.20		mPG 1.0		mPL 1.2	
		WL	QR	WL	QR	WL	QR	WL	QR	WL	QR	WL	QR
.25%Peku01	0.923	1.29	1.40	1.43	1.55	1.60	1.74	1.69	1.83	1.63	1.76	1.60	1.73
.50%Peku01	1.02	1.39	1.36	1.44	1.41	1.74	1.70	1.85	1.81	1.85	1.81	1.88	1.84
.75%Peku01	1.12	1.57	1.40	1.65	1.47	1.90	1.69	1.92	1.72	1.92	1.71	2.13	1.90
1%Peku01	1.25	1.71	1.37	1.71	1.37	1.99	1.59	2.09	1.67	2.13	1.71	2.41	1.93
2%Peku01	1.67	2.09	1.25	2.15	1.29	2.47	1.48	2.50	1.50	2.71	1.63	3.31	1.99
5%Peku01	3.02	3.47	1.15	3.45	1.14	3.97	1.31	3.87	1.28	4.28	1.42	5.28	1.75
10%Peku01	5.28	5.56	1.05	5.58	1.06	6.35	1.20	6.02	1.14	6.77	1.28	8.07	1.53
Average			1.28		1.32		1.53		1.56		1.62		1.81

[3] Charles J. Alpert, J.-H. Huang, and A. B. Kahng. Multilevel circuit partitioning. In *Proc. Design Automation Conf*, pages 530–533, 1997.

[4] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection alone produce routable placements? In *Proc. Design Automation Conf*, pages 477–482, 2000.

[5] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Optimal partitioners and end-case placers for standard-cell layout. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(11):1304–1313, 2000.

[6] T. Chan, J. Cong, T. Kong, and J. Shinnerl. Multilevel optimization for large-scale circuit placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 171–176, 2000.

[7] Chin-Chih Chang, J. Cong, and Min Xie. Optimality and scalability study of existing placement algorithms. In *Proc. Asia South Pacific Design Automation Conf.*, pages 621–627, 2003.

[8] Chin-Chih Chang, Jason Cong, Zhigang Pan, and Xin Yuan. Physical hierarchy generation with routing congestion control. In *Proc. Int. Symp. on Physical Design*, pages 36–41, 2002.

[9] Jason Cong, M. Romesis, and Min Xie. Optimality, scalability and stability study of partitioning and placement algorithms. In *Proc. Int. Symp. on Physical Design*, pages 88–94, 2003.

[10] Alfred E. Dunlop and Brian W. Kernighan. A procedure for placement of standard-cell VLSI circuit. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-4(1):92–98, 1985.

[11] Hans Eisenmann and Frank M. Johannes. Generic global placement and floorplanning. In *Proc. Design Automation Conf*, pages 269–274, 1998.

[12] George Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proc. Design Automation Conf*, pages 526–529, 1997.

[13] Patrick H. Madden. Partitioning by iterative deletion. In *Proc. Int. Symp. on Physical Design*, pages 83–89, 1999.

[14] G. Sigl, K. Doll, and F. M. Johannes. Analytical placement: a linear or a quadratic objective function? In *Proc. Design Automation Conf*, pages 427–432, 1991.

[15] UMichigan. http://vlsicad.eecs.umich.edu/BK/PlaceUtils/.

[16] M. Wang, X. Yang, and M. Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. Int. Conf. on Computer Aided Design*, pages 260–263, 2000.

[17] Xiaojian Yang, Bo-Kyung Choi, and M. Sarrafzadeh. Routability-driven white space allocation for fixed-die standard-cell placement. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(4):410–419, 2003.

[18] Mehmet Can Yildiz and Patrick H. Madden. Global objectives for standard cell placement. In *Proc. Great Lakes Symposium on VLSI*, pages 68–72, 2001.

[19] Mehmet Can Yildiz and Patrick H. Madden. Improved cut sequences for partitioning based placement. In *Proc. Design Automation Conf*, pages 776–779, 2001.