MMP: A Novel Placement Algorithm for Combined Macro Block and Standard Cell Layout Design†

Hong Yu, Xianlong Hong, Yici Cai

Department Of Computer Science and Technology

Tsinghua University, Beijing 100084, P. R. China

Tel: +86-10-62785564

Fax: +86-10-62781489

e-mail: {yuhong, hong, caiyc}@tiger.cs.tsinghua.edu.cn

Abstract—In this paper, an efficient mixed-mode placement algorithm called MMP is presented for the high performance mixed block and standard cell designs. Our approach combines the well-known quadratic placement with bottom-up clustering, as well as the slicing partitioning strategy. This approach can account for macro blocks and standard cells simultaneously. Our method is both very efficient and effective, while it can be run very fast, too. We have tested our algorithm on a set of sample circuits from industry and consistently obtained excellent results.

Keywords: Mixed-Mode, Clustering, Quadratic Placement, Partitioning.

I. INTRODUCTION

As the first step in physical design process, the problem of module placement is crucial for automatic layout design in VLSI. A good placement is essential to a good layout design. In the physical implementation of deep-submicron ICs, placement solution quality is a major determinant of whether timing correctness and routing completion will be achieved. Generally, many factors are involved in producing a good placement. These include, for example, the total wire length, wire crossings, heat dissipation and total circuit area. Note that while total wire length does not represent all of these design goals exactly, it is reasonable to assume that overall shorter wires lead to smaller circuit area, less resistance, and fewer wire crossings. Therefore, in row-based placement, the first order objective has always been obvious: to place connected cells closer together so as to reduce total wirelength and lower bounds on signal delay.

Conventional placement algorithms always assume that the placement objects are similar in size, and the height of the largest cell is at most 3 times of the smallest cell. However, with the cell count on a chip continues to increase and the systems become more and more complicated, it is usually impractical to design very large circuits only with row-based design styles, such as gate-array or standard-cell. In real designs, a large portion of a semi-custom chip consists of macro blocks like ROMs, RAMs, PLAs and datapaths which are pre-destined and much larger than the basic cells. In this case, conventional placement algorithm can no longer apply.

The aim of our work is to be able to automatically generate area efficient layouts for netlist containing a mixture of a large number (10,000 or more) of small gates (standard cells) and a few (10 to 20) large macro blocks. The task of placement is to locate the positions of both standard cells and large macro blocks so as to meet the design requirements.

Due to the difficulty and complexity, very few papers about this kind of problems have been reported, and up to now there has appeared no effective algorithm yet. However, with the need for high performance circuit layout continues to grow, some helpful research and attempts have been carried on.

In 1985, L. Sha [1] adopted the technique of nonlinear programming to solve the problem of macro block placement. Its main idea is to optimize the total wirelength by means of mathematical programming, while taking the condition of non-overlap as subjects. A. Alon [2] and C. S. Ying [3] took the strategy of penalty function in order to remove the overlaps among the blocks. The main weak of this kind of method is that there are too many subjected conditions, and the computing complexity is very high. Moreover, since the penalty function is non-convex, the optimizing is very apt to be stuck at local optima.

Another effective approach referring to this problem is the strategy of graph partitioning based on min-cut [4] [5]. This method divides the circuit repeatedly in order to minimize the number of nets cut by the partitions. However, when there exist many standard cells, how to locate the positions of macro blocks and small cells simultaneously within a short time is a very hard work.

In addition, to deal with this problem, some researchers take use of the scheme of stochastic search such as simulated annealing. This method is based on an iterative statistical search algorithm accepting intermediate configurations that increase the global cost, in order to escape from local optima [6]. A practical implementation of

† This project is supported by 973 National Key Project. No. G1998030413.

simulated annealing is the Timberwolf package [7] [8]. Commonly to say, its results are good, but the algorithm is very time consuming because simulated annealing algorithm is a kind of systematic exhaustive search.

In this paper, an efficient and effective placement algorithm called MMP is presented for the high performance combined macro block and standard cell designs. MMP is the acronym of Mixed-Mode Placement. Our approach incorporates the well-known quadratic placement with bottom-up clustering, as well as slicing partitioning strategy. This method can account for both macro blocks and standard cells simultaneously, and can find the global optima within a very short time [9].

Organization of this paper is as follows: in Section II, the outline of MMP is depicted briefly; then in Section III, the algorithm is described and discussed in detail; subsequently, the algorithm of final placement is provided; in Section V, we show the experimental results, and finally we conclude this paper in Section VI.

II. Outline of MMP

We adopt quadratic wirelength as our objective function, simply because it tends to result in fewer very long nets than linear objective function, which implies that the quadratic function tends to place components more sparsely, resulting in better timing property and fewer components overlapping each other.

At beginning, circuit information is read in format of DEF/LEF, then the information is converted into our inner data format. Next, the original netlist has been reconstructed to be a mixed mode netlist. After that, a bottom-up clustering algorithm will be integrated into the quadratic placement problem by using clustering as a preprocessing step in the algorithm. First, clustering is performed on the circuit to obtain a condensed circuit in which each cluster of components has been collapsed to form a single component. The Q-Place problem is then solved on the condensed circuit instead of the original circuit. Since the number of components in the condensed circuit is usually much smaller than that of the original circuit, the time and space required by the Q-Place algorithm is reduced significantly. Moreover, our study shows that to solve the Q-place problem on the clustered circuit may usually lead to better result than that of direct solving, since strongly connected components in each cluster are not separated during the solving process [10]. Next, the global optimization and partitioning are to be implemented alternately; In each global optimization step, a mathematical programming problem is derived and solved. The solution is a global placement of both blocks and clusters. After that, we recover the clusters into the cells in the primary circuit and refine the solution we have obtained. In the end, final placement is implemented and the solution of placement is obtained.

III. Main algorithm description

A. Bottom-Up Clustering (CASH)

First, a clustering algorithm, named CASH [10] is performed on the reconstructed circuit to obtain a condensed circuit in which each cluster composed of a set of cells is seen to be as a single component. In clustering procedure, only standard cells be processed and any macro blocks or pads be not handled. After that, a Q-Place problem is solved on the clustered circuit instead of the original circuit.

Conceptually, we form a cluster C according to the inter-connectivity degree of the vertices, i.e., C is formed through greedily merging the most attractive vertex which has the biggest value of price P_j . Here, we define $p_j = \frac{innerConnect_j}{outerConnect_j}$, where $innerConnect_j$ is the total connections of cell j with other cells inside cluster C, and $outerConnect_j$ is the total connections of cell j with the cells outside cluster C. In the course of clustering, when we choose c_i as the seed of cluster C, we only search the vertices v_j, which connect to c_i instead of searching all the other vertices which have not been clustered. Since only a very small fraction of nets have more than ten terminals, our strategy can be carried out very fast.

In cluster C, when $| C | < C_{size}$, and there is no “free cell” vertices which connect to the vertices inside C, we stop adding other cells into C and begin to form another new cluster C’, simply because it is of no meaning to add any cell into C if it does not connect to the cells inside C.

If the total number of clusters is too large, the effect of reducing the scale of the primary problem by means of clustering will not be very significant and it may take a long time to run, while too small will lead to the ill-mapping between the "large cluster" with the "point" model. From our experiments, when the number of total clusters is kept between 4000-5000 and the size of each cluster is about the same, the algorithm can perform very perfectly. For more detail, please refer to [10]

B. Global Placement and Slicing Partition

The main loop of this phase is formed by an iteration of global optimization and partitioning steps. In each step of global optimization, a mathematical programming problem is formulated and solved. In the following partitioning step, clusters and blocks are assigned into sub-regions according to their positions. Two linear constraints are generated for each sub-region. Simply to know that the partitioning operation generates a slicing tree, in which each node represents a region. The loop is repeated until any sub-region contains either a single macro block or only standard cells. Then we can get a global placement for each cluster and block.

B.1 Objective Formulation

The objective function is based on nets' quadratic wirelength model. For a net n, its length is computed according to the following equation:

$$L_n = \sum_{ik,\, jl \in n} [(x_i + \xi_{ik} - x_j - \xi_{jl})^2 + (y_i + \eta_{ik} - y_j - \eta_{jl})^2]$$

(ξ_{ik}, η_{ik}) is the coordinates of pin k in cluster C_i relatively to the left-down corner of cluster C_i; (ξ_{jl}, η_{jl}) is similar. Since the number of nodes connected to each net is different from one another, we assign a weight w_n to each net n. The objective is to minimize the weighted sum of the quadratic wire length of all nets:

$$L_{total} = \sum_{n \in N} w_n * L_n$$

B.2 Partitioning and Iteration

According to the characteristics of macro blocks, and to simplify the problem, we suppose that all the blocks are hard and can not be rotated or reversed.

At the *l*th partition level, the placement plane will be divided into 2^l regions and each region contains a subset of clusters and/or blocks. Let $\Re^l$ denote the index set of regions. If r is a region, we use τ_r to denote the set of clusters and/or blocks contained in r, and use (μ_r, ν_r) to denote the coordinate of the center of the region r. Since a cluster or a block can reside in only one region, we use $\Re_i$ to denote the region which the cluster (or block) i resides in. Thus, in order to place blocks and clusters in the region evenly, for each region *r*, we get two distribution constraints on the global placement:

$$\sum_{i \in \tau_r} x_i / |\tau_r| = \mu_r \qquad \sum_{i \in \tau_r} y_i / |\tau_r| = \nu_r$$

Let L(x, y) = ϕ(x) + ϕ(y). Since ϕ(x) and ϕ(y) are similar in formation, here we consider only ϕ(x). (ϕ(y) is in the same way). Combining the objective function and the distribution constraints, we get a constrained quadratic programming problem (*LQP*):

$$LQP: min\{\Phi(x) = 1/2 * X^T Q X + b_x^T X \mid A^{(l)} X = u^{(l)}\}$$

We use Lagrange Relaxation Method to solve the LQP just in the same way as in VEAP [9]. According to [9], we can get the global optima :

$$x = Q^{-1} b_x, \qquad y = Q^{-1} b_y .$$

where the vectors *x* and *y* denote the coordinates of the movable clusters and blocks to be placed. Q is the matrix corresponding to quadratic form of the wirelength function of nets, which is positive definite if all movable clusters and blocks are connected to some fixed blocks or I/O pads either directly or indirectly. This holds since all clusters and blocks will be connected to I/O pads, which are assumed fixed during the iterations. While Q is a sparse matrix, Q^{-1} is usually dense. Moreover, to compute Q^{-1} is very time-consuming. As a result, direct solvers are not practical. However, we change the formula as follows:

$$Q\,x = b_x , \qquad Q\,y = b_y .$$

Then, we may take use of an iterative approach to solve above equations. Obviously, a well-suited iterative solution method for this class of problems is the conjugate-gradient method (CG). To improve the condition number of matrix Q, we use preconditioned conjugate-gradient method (PCG) in practice, which may result in an efficient solution.

B.3 Slicing Partitions

To remove the overlaps among blocks, we use an efficient partitioning method with iteration, i.e. top-down slicing according to the relative positions of clusters and blocks. Once after partitioning, each current region is separated into two sub-regions. We force to place each block within a sub-region according to some heuristics. Then we use the Q-place algorithm in the similar way as that used in VEAP [9] to find positions for all blocks and clusters. The key feature is that we again solve linear sparse equations, repeating the process in a top-down hierarchy to obtain the final solution.

As shown in Fig.1, we generalize a partition line to divide the blocks into two sets. We determine the location of the partition line as follows: at first, sort the blocks from left to right and add up the block areas until we get roughly half the total block area. Then we determine the V-PartitionLine according to the ratio of left region area A_l and right region area A_r: $\alpha_v = A_l / A_r$ ($\alpha_v > 1$, if $\alpha_v < 1$, then $\alpha_v = 1 / \alpha_v$). If the V-Partition line happens not to traverse any block, the V-Partition is done. If not, we may either move the positions of blocks which have been cut by the partition line or move the position of partition line so that no block has been cut by the partition line.

As shown in Fig.2(a), when we have made a V-Partition, the V-Partition line traverses block A and C. In

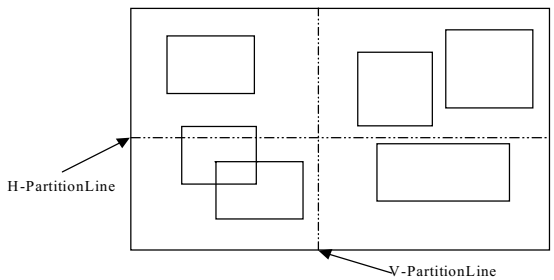
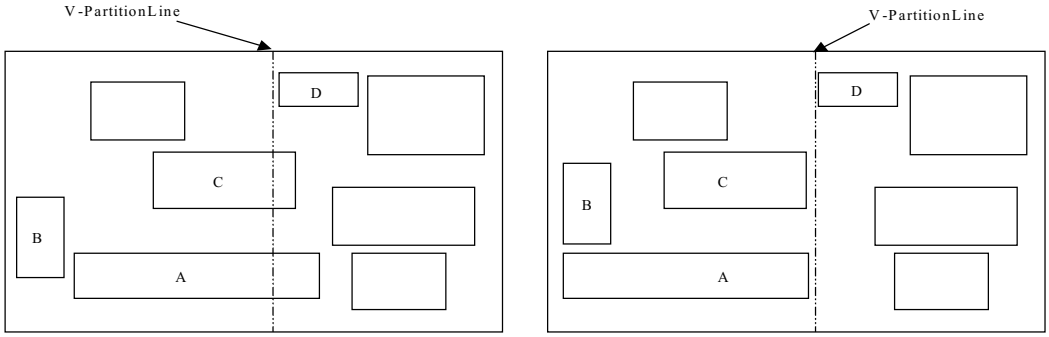


Fig. 1 Partitioning a region



(a) Before removal (b) After removal

Fig. 2 Adjusting the Positions of Macro Blocks

this case, we may first find the "critical block" in each sub-region. The "critical block" means the block which has the biggest dimension of width, (i.e. block A in the left sub-region of Fig. 2(a)). Similarly, in an H-Partition, the "critical block" means the block which has the biggest dimension of height. After that, we will first move the critical block to the left so that it can no longer be cut by the partition line. Next, we should move other blocks in this sub-region to their suitable positions. The result is shown as Fig. 2(b).

In most cases, the above scheme will work very well. However, in some cases, it may encounter some troubles. As shown in Fig. 3(a), if the dimension of critical block is too big in width, to adjust only the positions of DP blocks can not avoid traversing blocks by the partition line. In the case as shown in Fig. 3(a), we must adjust both the positions of blocks and the position of V-ParitionLine. The result of adjusting is shown as in Fig. 3(b). In this case, after we move the position of partition line, maybe we will have to move a few blocks which belong to a sub-region previously to the other sub-region so that the area ratio α_v can remain the same, i.e., block D in Fig. 3(b).

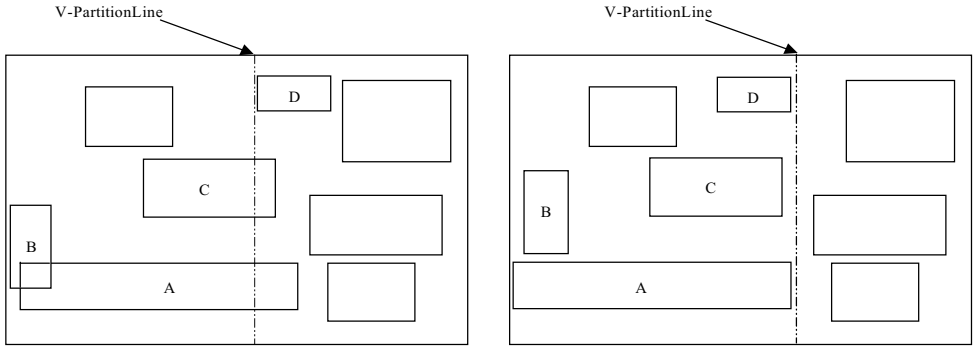
Since the total area of blocks usually covers less than 40% of the total area of the chip, with the area ratio α_v always remains about the same, the blocks in any region will be always looked like "sparse", which indicates that we can always make the partition if we remain the area of all the blocks in sub-region R_l and the area of blocks in sub-region R_r being proportional to α_v.

In the same way, we can determine the location of the H-PartitionLine, get a ratio of upper region area A_u and bottom region area A_b : $\alpha_h = A_u / A_b$. Then we check whether the V-Partition or the H-Partition is better (which ratio is nearer to 1). If $| \alpha_h - 1 | < | \alpha_v - 1 |$, then H-Partition is made. The previous H-PartitionLine is fixed, and also according to the principle of position sorting, we may assign clusters into the upper sub-region and bottom sub-region proportionally. If $| \alpha_h - 1 | \geq | \alpha_v - 1 |$, then V-Partition is made in the similar way.

To improve the objective of total wire length, in the following partitioning, we should repeat H-Partition and V-Partition alternatively. Of course, there exists such a special case that when H-Partition should be implemented, however, α_h is much larger than 1 while α_v is nearly equal to 1. In this case, we will implement V-Partition certainly.

B.4 Assigning Clusters

After the processing of blocks and determining of location of the partition line, we should allocate clusters to each sub-region proportionally according to the result of position sorting, so that the area ratio will remain nearly the same. To do this, we adopt the strategy of top-down assigning recursively.



(a) Before Adjusting (b) After Adjusting

Fig. 3 Adjusting the Positions of Partition Line

B.5 Routability Improvement

To improve the routability of the circuit, we take use of the method of min-cut. Whenever we finish a partition, then the min-cut strategy is used between the clusters in the two sub-regions. To remain the area ratio between the two sub-regions and to enhance the computing efficiency, we use a simple heuristic just like F-M Min-Cut method [5] in practice. First, we choose a cluster A in a sub-region, then we begin to search the clusters in another sub-region. If we have found a cluster B in another sub-region so that if we exchange the positions of cluster A and cluster B, the wire length can be reduced, we exchange them immediately.

To obtain a new global placement after partitioning, we introduce a simple heuristic: if a block is assigned to a region R, it can move only within region R. While the block can not move outside R, however, it can be assigned to a sub-region of R. In the mean time, clusters can always move across the whole chip freely.

We repeat the same procedure at the next level and continue until each region contains at most one block to be placed at the legal location in the region. By legal location, we mean the location that satisfies the legal site constraint. Since our partitioning scheme takes into account the available area, we will have no overlap.

B.6 Recover Clusters into Cells and Iterative-Improvement

During all the previous placement steps, we compute the positions of elements all in the form of clusters instead of cells. Therefore, it's time for us to recover the clusters into cells. Since the coordinates of the cells within a cluster is exactly the same, there is a large overlap of cells, which indicates that it must be very arbitrary when cells are assigned to slots. To eliminate or remove the overlap, we propose a simple heuristic to determine the coordinates of cell C_i. We suppose that the positions of all the other cells which connect to C_i are fixed, then we can get the optimal position of C_i. For C_i, the object function is :

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

Let:

$$\frac{\partial L_p}{\partial x_i} = 0, \qquad \frac{\partial L_p}{\partial y_i} = 0.$$

Then we have:

$$x_i = \left(\sum_{n \in N_i,\, i \neq j} w_n * x_j \right) \Big/ \sum_{n \in N_i,\, i \neq j} w_n * (n-1),$$

$$y_i = \left(\sum_{n \in N_i,\, i \neq j} w_n * y_j \right) \Big/ \sum_{n \in N_i,\, i \neq j} w_n * (n-1).$$

In this step, we apply this method to every cell recursively. The experimental results show that it is very effective in eliminating and removing overlap of cells.

IV. Final Placement

After global optimization and slicing partitions, a solution of global placement has been obtained. In this solution, although there are no overlaps between one macro block with another block, there exist some overlaps among standard cells. Furthermore, most of the positions of standard cells are illegal. Therefore, there need a final placement. Through final placement, we may move the positions of some standard cells to legal positions and eliminate overlaps. In standard cell designs, the cells are of approximately the same height but sometimes of fairly differing widths. For the mixed mode placement, since the standard cells are much smaller in size and much larger in number than the blocks, they are very ideal for use as a "filling" to cover up the empty spaces in the chip. The goal is not only to obtain narrow channels with equally distributed low wiring density, but also the rows with equal length. In MMP, the final placement for standard cells proceeds similarly to the method used in VEAP [9], which adopts a linear assignment algorithm and tends to achieve minimal wirelength and even row length. For more detail, please refer to [9].

V. Experimental Results

We have implemented our new mixed mode placement algorithm MMP on SUN ULTRA-SPARC workstations in C language. As what have been mentioned in section I, there are not many researches and discussions around the problem of mixed mode placement. For the papers reported up to now, the tested circuits are all hand-made circuits by the authors [1-2, 11-14]. For these test cases, there are usually several blocks, tens of cells and not more than one hundred nets. The scales of these sample circuits are too small to have much practicability and reality. To investigate the efficiency of MMP, we tested it with a set of real layout circuit designs which are from the industry. The main characteristics of the tested circuits are listed in Table I. The experimental results are summarized in Table II. Fig. 4 is the layouts of circuit 3 and circuit 5 produced by MMP. From the experimental results, it is shown that our approach MMP is very efficient and effective to solve the problem of mixed mode placement of very large circuit layout designs. No matter what about the stability, the practicability or the efficiency of the algorithm, the tested results of our approach are very encouraging and satisfied.

TABLE I

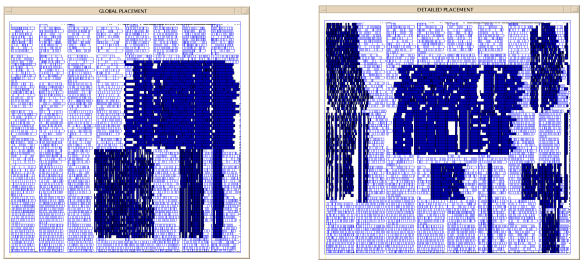
CHARACTERISTICS OF THE SAMPLE CIRCUITS

Sample Circuits	Cells	Blocks	Nets	A_B/A_T (%)
Circuit1	9253	0	10049	0
Circuit2	7094	2	10049	37
Circuit3	6330	4	10049	42
Circuit4	5996	6	10049	80
Circuit5	5895	9	10049	59
Circuit6	29384	9	31909	74.09
Circuit7	13405	7	31909	90.236

TABLE II

EXPERIMENTAL RESULTS OF THE SAMPLE CIRCUITS

Sample Circuits	Cpu Time	Wire length(um)
Circuit1	468.76	1493729.33
Circuit2	159.68	1491908.29
Circuit3	229.14	1959793.02
Circuit4	237.74	1631388.21
Circuit5	166.28	1423540.86
Circuit6	1611.56	23298654.25
Circuit7	1637.42	13097920.24



(a) Layout of Circuit3 (b) Layout of Circuit5

Fig 4. Layouts of Circuit3 and Circuit5

VI. CONCLUSION

In this paper, we have proposed a new fast mixed mode placement algorithm MMP for very large integrated circuit layout design in which quadratic placement algorithm is combined with bottom-up clustering strategy and slicing partitions. The algorithm MMP is effective and efficient in dealing with the problem of mixed mode circuit layout design which contains both macro blocks and standard cells. Our experiments on a set of real circuits which are from industry have demonstrated that our approach MMP is of not only high stability and strong practicability, but also excellent quality and fast speed. MMP is most fit for solving the problem of mixed mode placement of very large scale integrated circuits.

REFERENCES

[1] L. Sha & T. Blank, "ATLAS: A technique for layout using analytic shapes", Proc. International Conf. On Computer Aided Design, pp. 602-607, 1985.

[2] A. Alon & U. Ascher, "Model and solution strategy for placement of rectangular blocks in the Euclidead plane", IEEE Trans. CAD Vol. 7, pp. 378-386, 1988.

[3] C. S. Ying & J. S. L. Wong, "An analytic approach to floorplanning for hierarchical building block layout", IEEE Trans. Computer Aided Design, Vol. 8, pp. 403-412, 1989.

[4] B.W.Kernighan,S.Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", Bell System Technical Journal, Vol.49 , P.291-307, Feb. 1970.

[5] C.M.Fiduccia and R.M.Mattheyses,"A linear-time heuristic for improving network partitions", ACM/IEEE Proc.19th DAC , pp. 175-181, 1982.

[6] S. Kirkpatric, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing", Science, vol. 220, pp. 671-680, May 1983.

[7] C. Sechen and A. Sangiovanni-Vincentelli, "The timberwolf placement and routing package", IEEE Trans. Computer-Aided Design, vol. 20, no.2, pp. 510-522, April 1985.

[8] W. J. Sun and C. Sechen, "Efficient and Effective Placement for Very Large Circuits", Proc. Int. Conf. On CAD, 1990, pp. 336-339.

[9] Tianming. Kong , Xianlong. Hong and Changge Qiao, " VEAP: A Global Optimization Based Placement Algorithm for Standard Cell Design", ASP-DAC'97, Japan, 1997,1.

[10] Xianlong Hong, Hong Yu, Changge. Qiao, Yici Cai, "CASH: A novel quadratic placement algorithm for very large standard cell layout design based on clustering", Proceedings of the 5th International Conference on Solid-State and Integrated Circuit Technology, 1998, pp. 496-501.

[11] L. Sha and Robert W. Dutton, "An analytical algorithm for placement of arbitrarily sized rectangular blocks", 22nd ACM / IEEE Design Automation Conference, pp. 603-608.

[12] A. Herrigel and W. Fichtner, "An Analytic Optimization Technique for Placement Of Macro-Cells", 26th ACM / IEEE Design Automation Conference, pp. 376-381.

[13] Arun Shanbhag, SrinivasaRao Danda and Naveed Sherwani, "Floorplanning for Mixed Macro Block and Standard Cell Designs", Proc. The Conference of Great-Lake, 1994.

[14] Jitendra Apte, Gershon Kedem, " Heuristic Algorithms for Combined Standard Cell and macro Block Layouts", Proc. Of the Sixth MITConf. On Advanced Research in VLSI, 1990, pp. 367-385.

[15] T. Lengauer, "Combinatorial Algorithms for Integrated Circuit Layout", John Wiley & Sons, Chichester, 1990.

[16] Y.C. Wei and C.K. Cheng. "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning". In Proc. IEEE Intl. Conf. On Computer-Aided Design, pages 298-301,1989.

[17] E.L. Lawler, " Combinatorial Optimization: Networks and Matroids ", New York : Holt, Rinehert and Winston, 1976.

[18] J. Cong and M. Smith, " A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design", 30th ACM/IEEE Design Automation Conference, 1993, pp. 755-760.

[19] L. Hagen and A. Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering", Proc. IEEE Int. Conf. on Computer-Aided Design, 1991, pp. 10-13.