Multi-level Placement for Large-Scale Mixed-Size IC Designs [*]

Chin-Chih Chang,[†] Jason Cong and Xin Yuan

Department of Computer Science, University of California

Los Angeles, CA 90095 USA

e-mail: {cchang, cong, yuanxin}@cs.ucla.edu

Abstract— In this paper we study the large-scale mixed-size placement problem where there is a significant size variation between big and small placeable objects (the ratio can be as large as 10,000). We develop a multi-level optimization algorithm, MPG-MS, for this problem which can efficiently handle both large-scale designs and large size variations. Compared with the recently published work [1] on large-scale mixed macro and standard cell placement benchmarks for wirelength minimization, our method can achieve 13% wirelength reduction on average with comparable runtime.

I. INTRODUCTION

Circuit placement is an important step in VLSI physical design as it defines the interconnects which determine the overall performance of the final layout in deep submicron designs. This problem has been extensively studied in the past several decades. Historically, due to its complexity, the placement problem has been classified into two separate problems. One is called the *gate/cell placement problem*, where the designs consist of a large number of small placeable objects with similar sizes, as in the standard cell designs. The other is the so-called *module/block placement (floorplanning) problem*, which consists of a small number of large blocks (typically around a hundred), with possibly flexible aspect ratios. For the gate/cells placement problem, the optimization objectives include wirelength, delay, and routability, etc. Heuristic methods, such as min-cut-based algorithms and quadratic placement algorithms, were developed to handle the cell placement problem with emphasis on handling the high complexity. For the floorplanning problem, handling non-overlapping constraints and soft modules has been the major challenge. Several abstract floorplan representations, such as sequence-pair [2] and TCG [3], were developed to represent the overlap-free legal solutions which include an optimal solution. Various searching algorithms, such as simulated annealing, can be used to search for the optimal solution based on these representations.

With the advance of IC technology, especially with the reuse of IP blocks for multi-million-gate ASICs and SOC designs,
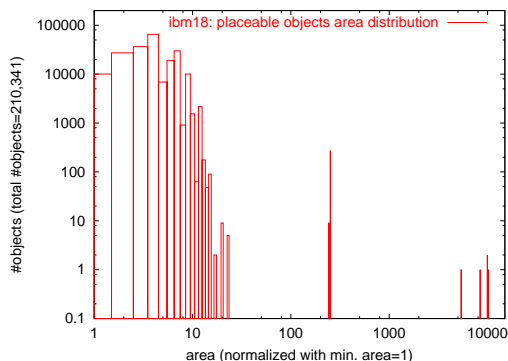

Fig. 1. Area distribution of placeable objects in circuit $ibm18$

most of modern IC designs consist of a very large number of standard cells mixed with many big macros, such as ROMs, RAMs and IP blocks. For example, Figure 1 shows the area distribution of the placeable objects of circuit *ibm18* [1] released from [1]. It shows that the number of placeable objects is very large (easily over 100,000) and the number of big objects can be as many as several hundreds. Moreover, it shows that the size ratio between big and small objects can be as large as 10,000. [2]

The traditional standard-cell placement techniques usually assume that the placeable objects have identical or similar sizes, while the floorplanning techniques usually can not handle the large-scale placement problem. Therefore, none of them is capable of solving the large-scale mixed-size placement problem alone. A common approach to this problem is to use a hierarchical design flow [5, 6, 7], where the standard cells are first partitioned into blocks using either the *logical hierarchy* or min-cut-based partitioning algorithms. Floorplanning is then performed on the partitioned blocks, together with macros for area and wirelength minimization. Finally, the cells in each block are placed separately. Though this method can reduce the problem size to the extent where the floorplanning technique can be applied, the quality of the final placement may not be good. As pointed out in [8], pre-partitioning standard cells to form rectangular blocks may prevent such a hierarchical method from finding an optimal or near-optimal solution in

---

[1]It was derived from ISPD-98 (IBM) circuit benchmarks [4] for mixed macro and standard cell placement.

[2]We categorize placeable objects into big and small objects based on the assumption that the size difference between large and small objects should be greater than 20 or 30.

terms of wirelength and delay minimization.

Therefore, a new methodology was proposed in [8], which first flattens the *logical hierarchy* to the extent that we are certain that the circuit elements in each module of the flattened hierarchy should physically stay together. Then a *physical hierarchy* is generated, which defines the global, semi-global, and local interconnects (based on their levels in the physical hierarchy). The physical hierarchy generation process determines the rough locations of the placeable objects in the flattened hierarchy which can include standard cells, small functional blocks (such as 32-bit adder, shifter, etc.), hard IP blocks and soft IP blocks. The core of the physical hierarchy generation is to solve the large-scale mixed-size placement problem for the flattened designs.

In this paper, we propose a multi-level placement algorithm, called MPG-MS, for large-scale mixed-size IC designs. The inputs include a set of placeable objects, netlist, I/O pads, the target chip width and height, and delay information for each placeable object. Pin locations are provided for objects with a fixed dimension. For soft objects with flexible aspect ratios under a fixed area, pin locations are assumed to be at the centers of the objects. [3] The outputs include a valid placement solution of all the placeable objects, the orientations of big objects, and aspect ratios of soft objects. The optimization objectives can be wirelength minimization, delay minimization, routability optimization, or a combination of them. We adopt the multi-level optimization method to handle this problem. Mixed-size placeable objects are simultaneously placed, the locations of larger objects are gradually fixed, and overlaps between larger objects are gradually removed while the locations of smaller objects are further refined during the multi-level optimization process. In this paper, we focus on wirelength minimization, however we think that the proposed method can be applied to other objectives as well.

The remainder of this paper is organized as follows. Section II reviews the previous work. Section III describes our placement algorithm, MPG-MS. The experimental results are shown in Section IV, followed by the conclusions and future work in Section V.

II. PREVIOUS WORK

Early approaches to the mixed-size placement problem use iterative improvement methods, i.e., simulated annealing-based placement techniques [9, 10], to simultaneously place the big and small objects. Although they give good results for small to medium-size designs, such methods have difficulty in scaling to very large designs due to their high complexity.

In [11], a quadratic placer was extended and combined with a two-level clustering scheme to handle the mixed macro and standard cell placement problem. However, the testcases in [11] were not large enough to demonstrate the effectiveness of this method for large-scale designs.

---

[3]Pin assignment can be performed during the placement phase. In this paper, we will not consider pin assignment during the placement.

Recently, a placement-floorplanning-placement flow [1] was presented to place designs with macros and a large number of small standard cells. This flow is similar to the hierarchical design flow as both of them use floorplanning techniques to generate an overlap-free floorplan followed by standard cell placement. Rather than using pure partitioning algorithms to generate blocks for standard cells, this flow proposes to use an initial placement result to guide block generation for standard cells. As we pointed out in Section I, such a hierarchical approach may lead to sub-optimal solutions, which is confirmed by the results of our method.

III. MULTI-LEVEL PLACEMENT FOR LARGE-SCALE MIXED-SIZE DESIGNS: MPG-MS

The major challenge in placing big and small objects together is how to handle the interaction between placing big and small objects. Without a good initial placement for *big objects*, the final placement may not be good, as the placement of *small objects* will largely depend on the locations of the *big objects*. On the other hand, placing *big objects* without considering *small objects* will not yield a favorable result, as the interconnections between *small objects* can not be ignored and they will play a somewhat important role in determining the quality of the final layout. Therefore we shall place them simultaneously. However, moving a *big object* can greatly affect wirelength, delay, and other objectives, and it is harder to remove the overlap between *big objects* than for *small objects*.

We think this problem can be nicely solved using the multi-level optimization method which has been successfully applied to several VLSI CAD areas, such as partitioning, cell placement and routing. The multi-level optimization method is very good at efficiently handling high complexity design problems. It consists of a coarsening phase and a refinement phase. In the multi-level optimization approach for the placement problem, placeable objects are clustered in the coarsening phase and gradually declustered and refined in the refinement phase by performing placement. The coarsening phase helps to reduce not only the problem size, but also the size variation between placeable objects at each level so that placement techniques can be more efficiently applied in the refinement phase. At each level in the refinement phase, the placer looks at a different level of abstraction of the flat design. Such abstraction provides enough detailed information of *small objects* for the placer to place *big objects* and *small objects* simultaneously. When a good initial placement for *big objects* is obtained, we shall fix their locations so that the placement of the *small objects* can be further optimized. When we fix the locations of the *big objects*, we shall generate an overlap-free placement for them based on the initial placement, keeping their locations as close to the initial placement as possible in order to have a consistent placement solution. We call this process *big objects legalization*. Figure 2 illustrates the proposed multi-level mixed-size placement flow.

Our algorithm, MPG-MS, follows the simulated annealing-based multi-level optimization framework MPG proposed
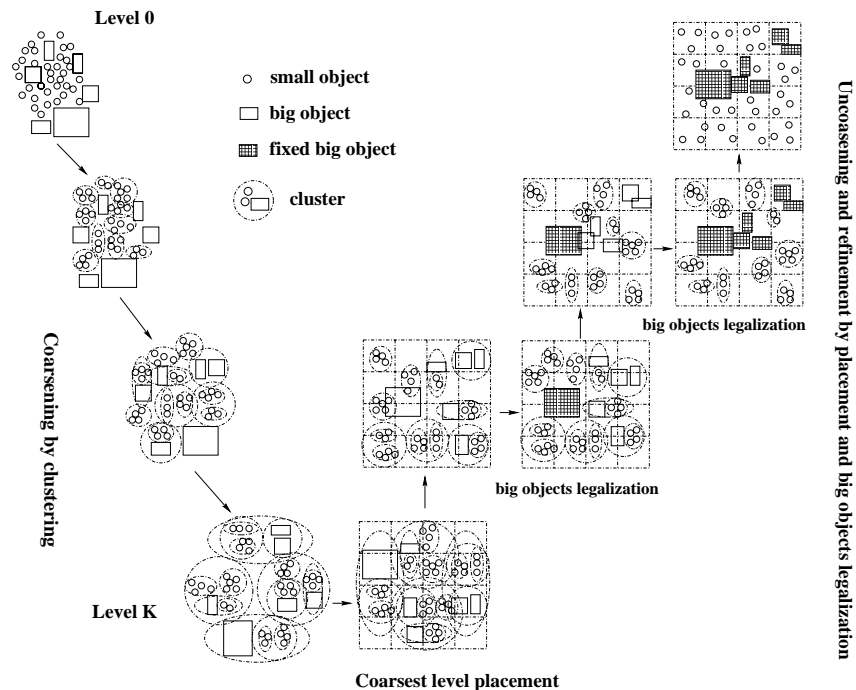
Level 0

○ small object

☐ big object

▦ fixed big object

⬭ cluster

Coarsening by clustering

Level K

Coarsest level placement

big objects legalization

big objects legalization

Uncoarsening and refinement by placement and big objects legalization

Fig. 2. Multi-level mixed-size placement flow

in [12], which will be reviewed in the next subsection.

A. Review MPG

In MPG, the coarsening is performed by recursively clustering the placeable objects using the *FirstChoice* (FC) clustering algorithm [13] to build a hierarchy of netlist and placement instances from level L_0 to L_1, ..., L_n. Level L_0 represents the input netlist and placement instance. Level L_n represents the coarsest level where the number of clusters is no less than a user-specified number, say 500. The refinement is performed by placing the clusters at each level to a bin structure using SA techniques. There are two key techniques that enable MPG to handle large-scale designs: a hierarchical area density control mechanism and the simulated annealing technique for multi-level optimization. Due to page limitations, we will not explain them in detail. [4] In general, other placement techniques can also be used at each level in the refinement phase.

B. Our Flow: MPG-MS

We first place the big and small objects to a bin structure[5] using the multi-level optimization method. After a coarse placement result is generated, (which is also an overlap-free placement for *big objects*), a detailed placement is performed to remove the overlap between the *small objects*. The following

---

[4]Please refer to [12] on how these two techniques efficiently place a large number of objects with different sizes.

[5]The bin structure can be specified by users or automatically set according to the design size. In general, the placement bin structure should be fine enough so that the wirelength estimated in the coarse placement stage is close to the wirelength estimated in the detailed placement stage.

section mainly focuses on how to handle *big objects* in the SA-based multi-level placement framework.

B.1 Coarsening Phase

Big objects will not be clustered at the beginning of the coarsening phase, but will be gradually clustered at coarser levels, as we need to fix the locations for *big objects* before we reach level L_0 in the refinement phase. In order to do that, we first classify the *big objects* into several groups according to their sizes. *Big objects* with similar sizes can be clustered from the same level. In that manner, we can gradually cluster *big objects* with other clusters, or *small objects*. Not allowing *big objects* to be clustered at all will deteriorate the quality of the clustering result measured by the connectivity and thus affect the efficiency of refinement.

B.2 Refinement Phase

After we decluster the clusters in level L_{i+1}, *big objects* will appear at level L_i if they are clustered in level L_{i+1}. *Big objects* that are legalized are called *fixed big objects*. *Big objects* that need to be legalized at current level are called *floating big objects*. The hierarchical area density control mechanism combined with SA-based moves can efficiently place objects with different sizes. In addition to moving placeable objects from bin to bin, changing the orientations of hard objects and the aspect ratios of soft objects are included in SA moves. After the SA process we get a placement for the current level which may not be an overlap-free placement for *big objects*. If the placement of *big objects* is valid, i.e., overlap-free, we do nothing and move to the next level of refinement; other-

wise, we need to generate an overlap-free placement for *big objects* which is as close to the original placement as possible. Given an invalid placement of *big objects*, the problem is how to move them to get an overlap-free placement under the chip dimension constraints, while trying to minimize the placement change (movement). This is a non-trivial problem, as the rectangle packing problem under the chip dimension constraints is NP-complete [14], let alone the goal of minimizing the movement. We call it the *big objects legalization problem*. Therefore, we propose a heuristic flow to handle it.

First, given the initial placement, we check whether it is possible to get a valid placement for *big objects* under the target chip dimension constraints. This is called *feasibility checking* (Section B.3). The placement is called *feasible* if it passes the *feasibility checking*. Depending on the result of *feasibility checking*, we then use one of the following two schemes to legalize the *big objects*: the complete legalization scheme if the initial placement is *feasible*, or the partial legalization scheme if it is not. The complete legalization scheme (Section B.4) generates an overlap-free placement based on a feasible initial placement. The partial legalization scheme (Section B.5) tries to remove partial overlap and to fix part of the *big objects*. According to the locations of newly fixed *big objects*, we again need to perform SA-based placement at the current level to place the remaining clusters if the complete legalization scheme is used, or to place *floating big objects* together with clusters if the partial legalization scheme is used, hoping that the SA process can remove the overlap and bring another better initial placement for the *floating big objects*. *Fixed big objects* can not be moved in the SA process. The wirelength-driven SA process, combined with the hierarchy area density control mechanism can push the overlapping movable clusters or *floating big objects* away from *fixed big objects* while trying to minimize the total wirelength. After SA placement is completed, we do the legalization again if any *floating big objects* exist. If, after several iterations, all the *big objects* still can not be legalized, we give up at this level and proceed to refinement at the next level. If it still fails at level L_0, we report failure. Figure 3 illustrates how the *big objects legalization* and the SA process are integrated.

B.3 Feasibility Checking

In order to preserve the relative locations between the *big objects*, sequence-pair (SP) representation [2] is used to capture the relationship between the locations of the *big objects* in the invalid placement.

First we generate an SP from the invalid placement of the *big objects* by modeling each *big object* as a point (without dimension) located at the center of the object in the coordinate system. We then rotate the coordinate system clockwise 45 degrees. After we sort the *big objects* according to their x and y coordinates in the rotated system in non-decrease order, Γ_+ is set to the order list according to the x coordinates and Γ_- is set to the order list according to the y coordinates. An example is shown in Figure 4.
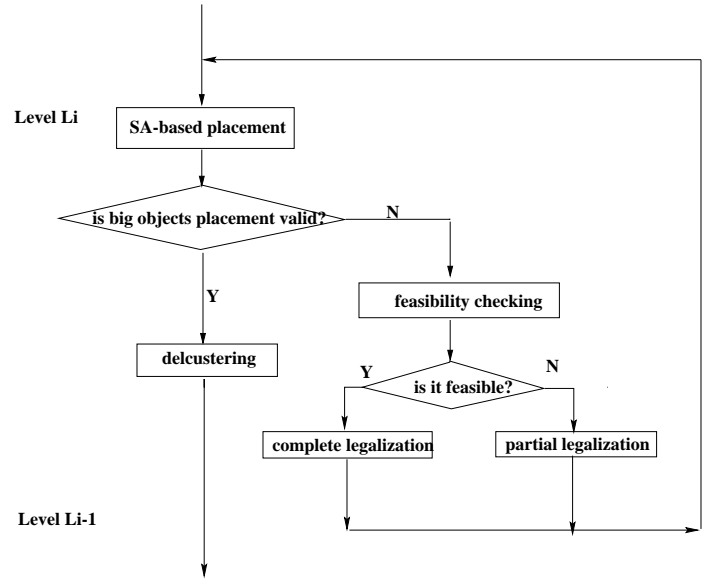

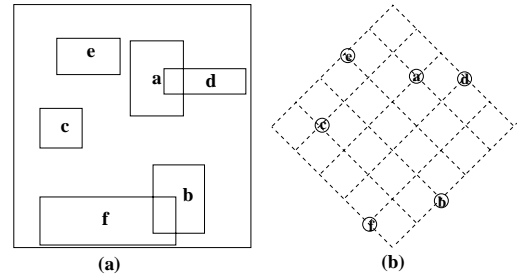Fig. 3. Interaction between *big objects legalization* and SA-based placement

Fig. 4. An example of deriving an SP from a given placement. (a)A given placement of *big objects*. (b)In the rotated coordinate system, each object is modeled as a point located at the center of the object. The SP $(\Gamma_+, \Gamma_-) = (ecadfb, fcbead)$.

According to the derived SP, we can build the horizontal-constraint graph G_H and vertical-constraint graph G_V. The vertex-weight $w(v)$ of vertex v in G_H (G_V) is set to the width (height) of the corresponding object. No weight is associated with the edges. If the longest path in G_H (G_V) is not longer than the width (height) of the chip W (H), this SP is considered to be feasible and the complete legalization scheme is used to generate an overlap-free placement; otherwise it is infeasible and the partial legalization scheme is used. [6]

B.4 Complete Legalization Scheme

After we determine that an SP is feasible, we are sure that at least the packing solution can guarantee an overlap-free placement. However, we also need to move *big objects* as little as possible from their original locations in order to minimize

---

[6]Note that the SP derived by our algorithm may not be the best SP that can lead to a feasible placement. We can swap adjacent elements in SP to reduce the longest path while trying to maintain the relative relationship between objects' locations. A couple of heuristic methods were introduced in [15] trying to transform an infeasible SP to a feasible SP. We can use them. However such heuristics still can not guarantee success.

the extra movement due to the overlap removal and thus make the *big objects* placement result as consistent with the initial placement of the current level as possible.[7] We use a heuristic method, called *longest path compaction*, for this problem. It adjusts the distance between non-overlapping *big objects* in order to push the overlapping *big objects* away. First we add weight to the edges in G_H (G_V) in the following way: for an edge $e_{i \rightarrow j}$ connecting object (vertex) i and j, if there is no overlap between them, the edge weight $w(e)$ is set to the horizontal (vertical) distance between object i and j in the initial placement; otherwise it is set to zero. We then compute the longest path in the modified G_H (G_V). If there exists a path $p = (v_1, \ldots, v_n)$ in G_H (G_V) whose length exceeds the width (height) of the chip W (H), we "compact" the path by reducing the positive edge weight. For example, in G_H, for the longest path p, where $length(p) = \sum_{v \in p} w(v) + \sum_{e \in p} w(e) > W$, we compute the value of scale factor k, where $k = (W - \sum_{v \in p} w(v)) / \sum_{e \in p} w(e)$. Because passing the feasibility check can guarantee that $W - \sum_{v \in p} w(v) \geq 0$, k is a number between 0 and 1, i.e., $0 \leq k < 1$. We then reduce the edge weight $w(e)$ to $k \cdot w(e)$, i.e., proportionally scale down the distance between the non-overlapping objects. After compacting one path, other paths' lengths may change. Therefore we need to re-compute the longest path and compact it until the length of the longest path does not exceed the chip dimension. As each time when we compact a path, the positive weights of edges in this path decrease, other paths' lengths will not increase and thus the process will converge after, at most, m iterations, where m is the number of paths whose lengths exceed the chip dimension before the process of *longest path compaction* starts. After the compaction process, we can get an overlap-free placement for the *big objects*.

B.5   Partial Legalization Scheme

When the SP is infeasible, we have to partially fix the locations for some non-overlapping *big objects* using heuristic methods. For a *fixed big object*, if it overlaps with other *floating big objects*, we first identify them and then push them aside to remove the overlap. For a *floating big object*, we identify a group of *floating big objects* which overlap with it and pack them to remove the overlap. We then fix the *big objects* that do not overlap with others and re-perform the SA-based placement at the current level. *Big objects* that have been fixed are not moved in the SA process. The 2nd-round SA process tends to move the overlapping objects away from the fixed objects. We then perform legalization again on the SA placement result. If after several iterations, all the *big objects* still can not be legalized, we give up at this level and proceed to refinement at the next level. If it still fails at level L_0, we report failure.

---

[7]In fact, the problem can be formulated into a nonlinear programming (NLP) problem where the constraints are the requirement that no two *big objects* overlap spatially and the objective is to minimize the sum of extra movement due to overlap removal. However, it is very expensive to solve such NLP.

TABLE I BENCHMARK CHARACTERISTICS

circuit	#cells	#MAs	#pads	#nets	$\sum A_m$	A_m^b	$A_m^b : A_m^s : A_c^s$
ibm01	12260	246	246	14111	67.13%	6.37%	8416:252:1
ibm02	19071	271	259	19584	76.89%	11.36%	30042:240:1
ibm03	22563	290	283	27401	70.75%	10.76%	33088:240:1
ibm04	26925	295	287	31970	59.82%	9.16%	26593:240:1
ibm05	28146	0	1201	28446	0.00%	0.00%	-
ibm06	32154	178	166	34826	72.90%	13.64%	36347:175:1
ibm07	45348	291	287	48117	52.56%	4.75%	17578:240:1
ibm08	50722	301	286	50513	67.35%	12.11%	50880:240:1
ibm09	52857	253	285	60902	52.42%	5.42%	29707:240:1
ibm10	67899	786	744	75196	81.37%	4.80%	71299:252:1
ibm11	69779	373	406	81454	49.76%	4.48%	29707:240:1
ibm12	69788	651	637	77240	73.00%	6.43%	74256:252:1
ibm13	83285	424	490	99666	47.64%	4.22%	33088:240:1
ibm14	146474	614	517	152772	26.72%	1.99%	17860:144:1
ibm15	160794	393	383	186608	43.34%	11.00%	125562:240:1
ibm16	182522	458	504	190048	48.71%	1.89%	31093:252:1
ibm17	183992	760	743	189581	23.78%	0.94%	12441:252:1
ibm18	210056	285	272	201920	11.96%	0.96%	10152:243:1

IV. EXPERIMENTAL RESULTS

We implemented MPG-MS in C++/STL and tested it on a Sun Blade 1000 workstation running at 750MHz frequency. The testcases, downloaded from [16] in the GSRC Bookshelf format, are large-scale, mixed macro and standard cell placement benchmarks, except for circuit $ibm05$. The macros are all hard blocks with fixed aspect ratios and pin locations. The locations of I/O pads are given and not moved during placement. The circuits' characteristics are listed in Table I which consists of the number of standard cells (#cells), the number of macros (#MAs), the number of I/O pads (#pads), the number of nets (#nets), the total macro area vs. the total area of standard cells and macros in percentage (tot. A_m), the area of the biggest macro vs. the total area of standard cells and macros in percentage (A_m^b), and the ratio between the area of the biggest macro, the smallest macro and the smallest cell ($A_m^b : A_m^s : A_c^s$). We compared our placement results of wirelength (WL) and runtime (CPU) with those reported in [1] [8] in Table II. We also shown the total number of levels in the refinement phase for each circuit in the column titled "#LVs", the legalization schemes used for *big objects legalization* in the form $C^i|P_j^i$, where C^i refers to performing the complete legalization scheme at level L_i, P_j^i stands for running j iterations of the partial legalization scheme followed by a complete legalization scheme at level L_i. Figure 5 shows the final placement generated by our method for circuit $ibm02$.

These results show that our method MPG-MS can consistently out-perform the flow proposed in [1] with an average wirelength reduction of 13%, which demonstrates the efficiency of our method in handling such large-scale mixed size placement problem.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we present a method to handle the large-scale, mixed-size placement problem for fixed die-size IC designs. It

---

[8]Their runtimes were measured on a 1GHz PC/Intel system running Linux.

circuit	[1] Flow		MPG-MS				
	WL	CPU (min)	WL	WL improve	CPU (min)	#LVs	legalization scheme
ibm01	3.96e6	18	3.01e6	24%	18	4	C^3, C^2
ibm02	8.37e6	31	7.42e6	11%	32	4	C^3, P_2^2
ibm03	1.22e7	42	1.12e7	8%	32	4	C^3, C^2
ibm04	1.35e7	47	1.05e7	22%	42	4	C^3, P_2^2
ibm05	1.15e7	8	1.09e7	5%	36	4	-
ibm06	1.03e7	56	9.21e6	10%	45	5	C^4, C^3, C^2
ibm07	1.58e7	58	1.37e7	13%	68	5	C^4, C^2
ibm08	2.12e7	94	1.64e7	22%	82	5	C^4, P_2^2
ibm09	1.96e7	66	1.86e7	5%	84	5	C^4, C^3, C^2
ibm10	6.07e7	229	4.36e7	28%	172	5	C^4, C^2
ibm11	2.85e7	106	2.65e7	7%	112	5	C^4, C^3, C^2
ibm12	5.17e7	675	4.43e7	14%	153	6	C^5, C^4, P_{10}^2
ibm13	3.94e7	151	3.77e7	4%	151	5	C^4, C^2
ibm14	5.62e7	286	4.35e7	23%	276	6	C^5, C^4, C^3, C^2
ibm15	7.05e7	237	6.55e7	7%	385	6	C^5, C^4, C^3, C^2
ibm16	n/a	n/a	7.24e7	-	436	6	C^5, C^4, C^3, C^2
ibm17	9.24e7	503	7.85e7	15%	606	6	C^5, C^4, C^3, C^2
ibm18	5.49e7	318	5.07e7	8%	437	6	C^5, C^2
avg.				13%			

Fig. 5. The final placement of circuit $ibm02$ (Blocks in light color are macros, blocks in dark color are standard-cells).

is based on a multi-level optimization approach. Mixed-size placeable objects are simultaneously placed to obtain a good initial placement for *big objects*, then the big objects are gradually fixed and any overlap between the *big objects* is gradually removed, while small object placement is further refined during the multi-level optimization process. By integrating *big objects* placement and *small objects* placement into a single flow with consistent objectives, we can better optimize the designs compared with the hierarchical design flow where floorplanning is performed for the partitioned blocks followed by the standard cell placement. Experimental results on large-scale mixed-size placement benchmarks show that our method can out-perform the hierarchical flow by 13% on average in terms of total wirelength. Though we show the result for wirelength minimization using our proposed method, we believe that other objectives can be optimized in a similar way. Therefore we plan to incorporate other objectives, such as delay optimization, into the optimization process when placing large-scale mixed-size IC designs.

REFERENCES

[1] S. N. Adya and I. L. Markov, "Consistent placement of macro-block using floorplanning and standard-cell placement," in *Proc. Int. Symp. on Physical Design*, pp. 12–17, 2002.

[2] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, 1996.

[3] J.-M. Lin and Y.-W. Chang, "TCG: a transitive closure graph-based representation for non-slicing floorplan," in *Proc. Design Automation Conf*, pp. 764–769, 2001.

[4] http://nexus6.cs.ucla.edu/~cheese/ispd98.html.

[5] J. Apte and G. Kedem, "Heuristic algorithms for combined standard cell and macro block layouts," in *Advanced Research in VLSI: Proc. of the Sixth MIT Conference*, pp. 367–385, 1990.

[6] M. Upton, K. Samii, and S. Sugiyama, "Integrated placement for mixed macro cell and standard cell designs," in *Proc. Design Automation Conf*, pp. 32–35, 1990.

[7] A. Shanbhag, S. Danda, and N. Sherwani, "Floorplanning for mixed macro block and standard cell designs," in *Proc. the forth Great Lakes Symp. on VLSI*, pp. 26–29, 1994.

[8] J. Cong, "An interconnect-centric design flow for nanometer technologies," *Proceedings of the IEEE*, vol. 89, pp. 505–527, April 2001.

[9] C. Sechen and A. Sangiovanni-Vincentelli, "The Timberwolf placement and routing package," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 2, pp. 510–522, 1985.

[10] W. J. Sun and C. Sechen, "Efficient and effective placement for very large circuits," in *Proc. Int. Conf. on Computer Aided Design*, pp. 336–339, 1990.

[11] H. Yu, X. Hong, and Y. Cai, "MMP: a novel placement algorithm for combined macro block and standard cell layout design," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 271–276, 2000.

[12] C.-C. Chang, J. Cong, D. Pan, and X. Yuan, "Physical hierarchy generation with routing congestion control," in *Proc. Int. Symp. on Physical Design*, pp. 36–41, 2002.

[13] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," in *Proc. Design Automation Conf*, pp. 343–348, 1998.

[14] B. S. Baker, E. G. Coffman, and R. L. Rivest, "Orthogonal packings in two dimensions," *SIAM J. Compute.*, vol. 9, no. 4, pp. 846–855, 1980.

[15] S. Nag and K. Chaudhary, "Post-placement residual-overlap removal with minimal movement," in *Proc. Design, Automation and Test in Europe Conference*, pp. 581–586, 1999.
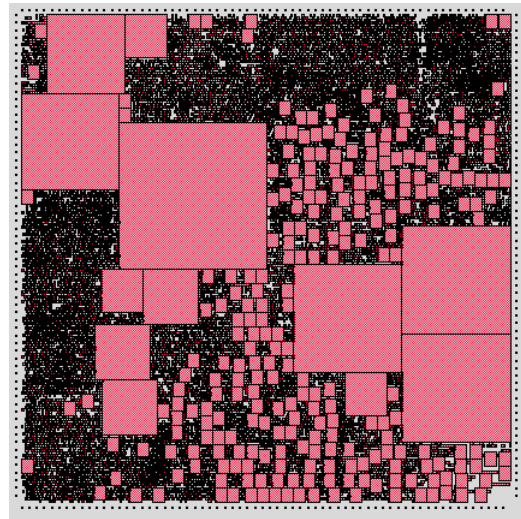
[16] http://vlsicad.eecs.umich.edu/BK/ISPD02bench.