RC24001 (W0607-031) July 12, 2006
Electrical Engineering

IBM Research Report

FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control

Natarajan Viswanathan
IBM Research Division
Austin Research Laboratory
11501 Burnet Road
Austin, TX 78758

Min Pan, Chris Chu
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011-3060

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control

Natarajan Viswanathan, Min Pan and Chris Chu
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011-3060, USA
email: {nataraj, panmin, cnchu}@iastate.edu

Abstract— In this paper, we present *FastPlace 3.0* – an efficient and scalable multilevel quadratic placement algorithm for large-scale mixed-size designs. The main contributions of our work are: (1) A multilevel global placement framework, by incorporating a two-level clustering scheme within the flat analytical placer *FastPlace* [26, 27]. (2) An efficient and improved Iterative Local Refinement technique that can handle placement blockages as well as placement congestion constraints. (3) A placement congestion aware standard-cell legalization technique in the presence of placement blockages.

On the ISPD-2005 placement benchmarks [19], our algorithm is $14.9X$ and $4.4X$ faster than state-of-the-art academic placers *APlace2.0* and *mPL6* respectively. In terms of wirelength, we are on average, 3% better than *APlace 2.0* and 4% higher as compared to *mPL6*. We also achieve competitive results compared to a number of academic placers on the placement congestion constrained ISPD-2006 placement benchmarks [20].

I. Introduction

As semiconductor technology advances into the nanometer regime, there has been a tremendous increase in the size of integrated circuits. Designs today often contain over a million objects and this number is steadily increasing. Hence, it is necessary to have efficient and scalable placement algorithms that can handle this ever increasing placement problem size.

Existing placement algorithms employ various approaches including *simulated annealing* [23, 24], *paritioning* [1, 2, 7, 28] and *analytical placement* [4, 9–11, 16, 17, 21, 26, 27]

Analytical placement algorithms based on the quadratic objective funtion (also called quadratic placers) are very popular as they are quite efficient and also give good quality of results. They typically employ a flat placement methodology [9–11, 17, 26, 27] so as to maintain a global view of the placement problem.

But, with circuit sizes steadily increasing towards tens of millions of placeable objects, a flat placement methodology may not be effective in handling the large placement problem size in a reasonable amount of runtime. Hence, for efficient and scalable placement algorithm design, a hierarchical approach is beneficial. To this effect many modern placers follow a hierarchical or multilevel approach for placement [3, 4, 13, 15, 21, 25]

Also, an important placement constraint that needs to be handled by current placers is that of placement congestion. Designers often run placement algorithms with specific *placement_target_density* values. To determine the placement density, a pre-defined grid is imposed over the placement region. Usually, the grid is square with the height being a multiple of the standard-cell row height. This gridding divides the placement region into bins. The *density* of a bin is then defined as the ratio of the the total area of movable objects within the bin to the total available free space within the bin. The *placement_target_density* basically specifies the maximum possible occupation for any bin in the placement region. Satisfying the *placement_target_density* constraint means that the *density* of all the bins in the placement region should be less than or equal to the *placement_target_density* value.

The purpose of the *placement_target_density* is to allow for more room within a bin for the subsequent routing step. It also allows for more space within a bin for buffer insertion, gate-sizing and other timing optimization transformations that might be performed after the placement step.

In this paper we address the two issues of scalability and placement congestion. We present an efficient multilevel quadratic placement algorithm with placement congestion control for large-scale standard-cell and mixed-size designs. The main contributions of our work are:

- Incorporating a multilevel framework within the global placement stage of the flat quadratic placer *FastPlace* [26]. This is done by employing two levels of clustering: an intial netlist based fine-grain clustering followed by a netlist and physical based coarse-grain clustering. Where the physical information for the second level is obtained from an initial placement of the fine-grain clusters.

- An improved Iterative Local Refinement Technique to reduce the wirelength based on the half-perimeter measure. This technique is very effective in simultaneously reducing the wirelength while spreading the cells around the placement region. It can also effectively deal with placement blockages and placement congestion constraints.

- A density-based standard-cell legalization technique. This technique operates on the segments created in the placement region due to the presence of placment blockages. It satisfies segment capacities as well as placement congestion constraints and legalizes the standard-cells within segments.

The rest of this paper is organized as follows: Section II gives an overview of the multilevel global placement framework and an outline of our algorithm. Section III describes the two-level clustering scheme used during global placement. Section IV describes the improved Iterative Local Refinement technique and its use in placement congestion control. Section

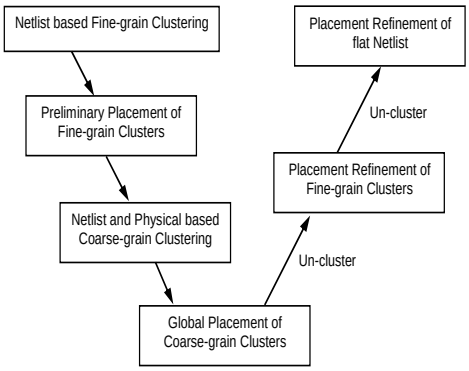

Fig. 1. Multilevel Global Placement Framework.

V describes the density aware legalization and detailed placement techniques. Experimental results are provided in Section VI followed by the conclusions in Section VII.

II. Overview of the Algorithm

In this section, we give an overview of the multilevel global placement framework and the outline of our placement algorithm.

Our multilevel framework is summarized in Figure 1 and follows the classical hierarchical flow that has been used in many existing placement algorithms [3, 4, 6, 13, 15, 21].

In Step 1 of the multilevel flow, we create fine-grain clusters of about 2-3 objects per cluster based on the connectivity information of the original flat netlist. In Step 2 we perform an initial placement of the fine-grain clusters. This step is extremely fast and its purpose is to have some placement information for the subsequent physical clustering step. In Step 3 we create coarse-grain clusters by performing a second level of clustering. This step not only considers the connectivity information between the objects (based on the clustered netlist previously obtained) but also uses the physical locations of the clusters obtained from the inital placement of Step 2. This creates a good-quality clustering solution for the subsequent global placement step. In Step 4 we perform global placement on the coarse-grain clustered netlist until the clusters are evenly distributed over the placement region. Since the number of placeable objects at this level are significantly less as compared to the original flat netlist, this step is extremely fast and significantly contributes to the overall efficiency of the placement algorithm. After the placement of the coarse-grain clusters, we perform a series of un-clustering and placement refinements in Steps 5 and 6, finally yielding a global placement solution of the original flat netlist.

The entire flow of our placement algorithm is summarized in Figure 2. The key components of our placement flow are: global placement using a multilevel framework, legalization of macro blocks using the Iterative Clustering Algorithm [27] followed by a density based standard-cell legalization scheme and an efficient detailed placement algorithm [22]. The individual components of the flow are described in more detail in the subsequent sections.

Stage 1: Global Placement
- **Level 1: Initial Placement**
  1. Construct fine-grain clusters using netlist based clustering
  2. Solve initial quadratic program
  3. Repeat
     a. Perform *regular Iterative Local Refinement* on fine-grain clusters
  4. Until the placement is roughly even
- **Level 2: Coarse Global Placement**
  5. Construct coarse-grain clusters using netlist and physical based clustering
  6. Repeat
     a. Solve the convex quadratic program
     b. Perform *density-based Iterative Local Refinement* on coarse-grain clusters
     c. Perform *regular Iterative Local Refinement* on coarse-grain clusters
     d. Perform *cell-shifting* on coarse-grain clusters and add spreading forces
  7. Until the placement is quite even
- **Level 3: Refinement of fine-grain clusters**
  8. Un-cluster coarse-grain clusters
  9. Perform *density-based Iterative Local Refinement* on fine-grain clusters
  10. Perform *regular Iterative Local Refinement* on fine-grain clusters
- **Level 4: Refinement of flat netlist**
  11. Un-cluster fine-grain clusters
  12. Perform *density-based Iterative Local Refinement* on flat netlist
  13. Perform *regular Iterative Local Refinement* on flat netlist

Stage 2: Legalization
1. Legalize and fix macro blocks using *Iterative Clustering Algorithm*
2. Move standard-cells among segments to satisfy segment capacities
3. Legalize standard-cells within segments

Stage 3: Detailed Placement

Fig. 2. Outline of Our Placement Flow.

III. Clustering for Placement

Circuit clustering is an attractive method to reduce the placement problem size for large- scale VLSI designs. If clustering is performed in a careful manner, it can also yield better wirelength along with faster runtime as compared to flat placement approaches.

In our multilevel framework we use clustering in a *persistent* context as defined by [21]. As in, we use clustering at the beginning of placement to pre-process the flat netlist so as to reduce the placement problem size. We then perform global placement on the coarse-grain clustered netlist. Once the clusters have been evenly spread out over the placement region, we execute a series of unclustering and "refinement" steps to further optimize the wirelength and improve the placement congestion.

In our multilevel framework, we follow a two-level clustering scheme as shown in Figure 1. In the first level of clustering we create fine-grain clusters of about 2-3 objects per cluster. This clustering is solely based on the connectivity information between the objects in the original flat netlist. Since this clustering is performed at the beginning of placement, we restrict it to fine-grain clustering to minimize any loss in placement quality due to incorrect clustering. In fact, it was demonstrated in [12] that fine-grain clustering can improve placement efficiency with negligible loss in placement quality.

We then perform a fast, initial placement of the fine-grain clusters. The purpose of this step is to get some placement information for the next clustering level. Since each cluster in the first level has only around 2-3 objects, the initial placement of the clusters closely resembles an initial placement of the original flat netlist. We then create coarse-grain clusters by performing a second level of clustering. In this level, we consider both, the connectivity information between the clusters and their physical locations as obtained from the initial placement. We believe that generating coarse-grain clusters

Algorithm Clustering

Phase 1: Construct Initial Priority-queue (PQ)

For each object *j*

1. Find closest object *k* and clustering score *s(j, k)*
2. Insert triple *(j, k, s)* into PQ with *s* as the key

Phase 2: Form Clusters

while (*number_of_objects* > *target_number_of_objects*)

1. Pick top triple *(j, k, s)* from PQ
2. if *j* is marked invalid
3. Re-calculate closest object *k'* and clustering score *s'(j, k')*
4. Insert triple *(j, k', s')* into PQ
5. else
6. if *fine-grain clustering*
7. if (*a(j)* + *a(k)* < *max_cluster_size*) cluster *j* and *k* into new object *j'*
8. if *netlist + physical clustering*
9. Calculate *d(j, k)* the distance between *j* and *k*
10. if (*d(j, k)* < *distance_threshold* and *a(j)* + *a(k)* < *max_cluster_size*) cluster *j* and *k* into new object *j'*
11. Update netlist based on the clustering
12. For object *j'* find closest object *k'* and clustering score *s'(j', k')*
13. Insert triple *(j', k', s')* into PQ with *s'* as the key
14. Mark neighbours of *j'* as invalid

Fig. 3. Best-Choice Clustering Algorithm with Placement Information.

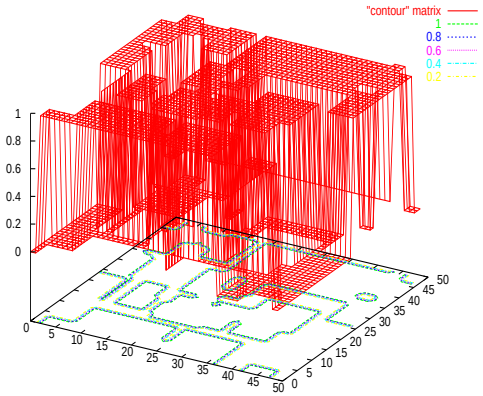

Fig. 4. Initial Contour Map Depicting Placement Blockages.

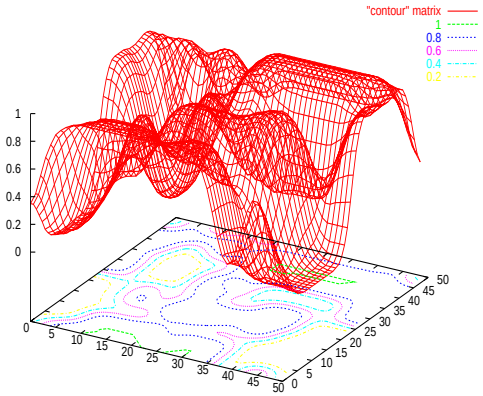

Fig. 5. Contour Map after Smoothing Transform.

based on actual placement information, is better than generating them by a solely netlist based approach; and such an approach would further minimize any loss in (or even improve) the final placement wirelength.

The key difference between our clustering scheme and the ones followed in [3, 5, 15, 21] is that we use actual placement information while forming coarse-grain clusters, whereas the other approaches generate coarse-grain clusters solely based on netlist information. Our approach closely resembles that of [13]. The difference being that [13] uses two-levels of netlist based clustering followed by physical clustering, whereas we only use one level of fine-grain netlist based clustering.

For both levels of clustering, we use the *Best-Choice* clustering algorithm described in [21]. In Figure 3 we summarize the modified version of the *Best-Choice* clustering algorithm using Lazy-Update speed-up technique to consider our two-level clustering scheme.

IV. Congestion Aware Iterative Local Refinement

The Iterative Local Refinement (ILR) technique is a key component of our placement flow. This technique is highly effective in minimizing the wirelength of the placement while simultaneously distributing the cells over the placement region. This section describes the improved ILR technique and its application in placement congestion control.

We separate the ILR technique into two components. A density-based ILR *d-ILR* and the regular ILR *r-ILR*. The core algorithm used to move the cells, within both the components is the same and hence we only describe it in the context of the *r-ILR*. In the subsequent subsections, we first give an overview of the ILR technique of [26], followed by the enhancements made to the technique. We then describe the top level flow for ILR based placement congestion control.

A. Description of the Technique

During ILR the placement region is binned and the utilization of all the bins is determined. After this step, the respective *source* bins of all the cells is determined. For every cell present in a bin, 8 scores are computed that correspond to moving the cell to its nearest 8 neighboring bins. For calculating the score, it is assumed that a cell is moving from its current position in a *source* bin to the same relative position in the *target* bin. The score for each move is a weighted sum of two components: The first component is the half-perimeter wirelength reduction for the move and the second is a function of the utilization of the source and target bins.

For a cell i currently in bin m, if we define:

- α: Weight for the wirelength component.
- β: Weight for the utilization component.
- $wl_i(m)$: Half-perimeter wirelength when i is in bin m
- $wl_i(n)$: Half-perimeter wirelength when i is in bin n
- $U(m)$: Utilization function for bin m
- $U(n)$: Utilization function for bin n

Then the score for the move from bin m to bin n is given by:

$$s_i(m,n) = \alpha(wl_i(m) - wl_i(n)) + \beta(U(m) - U(n))$$

If all 8 scores for the cell are negative, it is not moved. Otherwise, it is moved to the bin with the highest score for the move. During one iteration of the Refinement, the above steps are followed for all the cells in the placement region. This iteration is then repeated until there is no significant improvement in the wirelength. For the first top-level iteration of ILR, the

width and height of the bins are set to 5 times that of the bin used during the Cell Shifting step. The width and height of these bins are then gradually brought down to the values used during Cell Shifting over subsequent iterations of the global placement.

B. Enhancements to the ILR Technique

A major drawback of the above described ILR technique is that every bin in the placement region, irrespective of whether it is sparse or dense, will have the same weight for the utilization component. This does not accurately reflect the placement distribution. A sparse bin should have a lesser utilization weight so that more cells can be moved into it. Correspondingly, the utilization weight for a dense bin should be higher so as to enable movement of cells out of this bin. In the enhanced version of the ILR each bin has its associated utilization weight that is constantly updated based on the placement distribution.

Another extension to the ILR is in handling placement blockages. Most ASIC circuits contain many placement blockages in the form of fixed macros. Quadratic placers often place a lot of movable objects on top of the fixed macros. These objects have to be moved out of the fixed macros in an effective manner with minimal increase in the wirelength. To handle fixed macros during placement, we construct a contour map of the placement region. Based on the fixed macros, each bin in the contour map has a value of either 1 in case it overlaps with a fixed macro or 0 otherwise. The initial contour map for one of the placement benchmarks is shown in Figure 4. We then run a smoothing transform on the entire contour map. This transform smooths the sharp edges in the original contour map creating a smoothed version as shown in Figure 5. This smoothing is basically done so that cells can easily move over and cross a fixed macro if required or slide down the slope so that it can be moved out of the macro.

Based on the above enhancements, for cell i in bin m, if:

- α: Weight for the wirelength component.
- $\beta(m)$: Weight of the utilization component for bin m.
- $\beta(n)$: Weight of the utilization component for bin n.
- γ: Weight for the contour component.
- $wl_i(m)$: Half-perimeter wirelength when i is in bin m
- $wl_i(n)$: Half-perimeter wirelength when i is in bin n
- $U(m)$: Utilization function for bin m
- $U(n)$: Utilization function for bin n
- $C(m)$: Contour height of bin m
- $C(n)$: Contour height of bin n

Then, the score for the move from bin m to bin n is given by:

$$s_i(m,n) =$$

$$\alpha(wl_i(m)-wl_i(n))+(\beta_m U(m)-\beta_n U(n))+\gamma(C(m)-C(n))$$

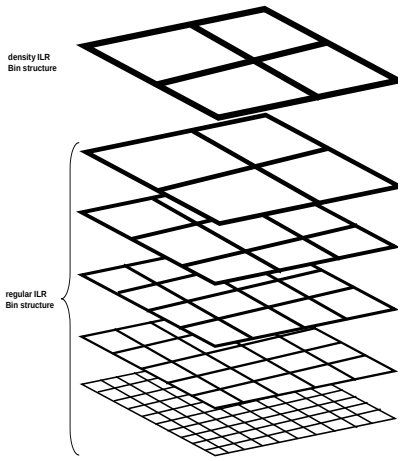

Fig. 6. Bin Structure for Iterative Local Refinement.

C. ILR for Placement Congestion Control

For placement congestion control, the ILR is divided into 2 components. The *d-ILR* uses the global pre-defined grid structure used for placement *density* computation. It then calculates the utilization and contour height for these bins. Cells are then moved from *source* to *target* bins of the global bin structure. based on the score function given in the previous subsection.

Once the *d-ILR* is performed, we then run the *r-ILR* as before in which the bin sizes are initial set to a large value and then decreased over subsequent placement iterations. The interaction between the *d-ILR* and the *r-ILR* can be seen in Figure 6 which shows the decrease in the size of the bins from the *d-ILR* stage to the end of the *r-ILR* stage.

V. Legalization and Detailed Placement

The aim of the legalization step is to resolve module overlaps present in the global placement solution and yield a legal non-overlapping placement. Our legalization stage is divided into two steps. In the first step, we ignore all the standard-cells and resolve overlaps among the macro blocks. We then fix them in legal positions in the placement region and legalize the standard cells. These steps are described in more detail below.

A. Macro Block Legalization

During legalization, we do not want to move the macros by a significant amount from their global placement positions. Hence, the goal of the macro block legalization algorithm is to resolve overlaps among the macros by perturbing them by the minimum possible distance from their global placement positions. This is achieved by using the *Iterative Clustering Algorithm* [27] for macro block legalization.

Briefly, [27] formulates the macro block legalization problem as a Minimum Perturbation Floorplan Realization problem. It uses the sequence pair [18] for floorplan representation and considers both movable and fixed macros during the construction of the sequence pair. It then uses low-temperature simulated annealing to determine a good sequence pair such that the corresponding placement of the movable macros, obtained from the *Iterative Clustering Algorithm* will resolve

overlaps among the macro blocks with minimum perturbation from their global placement positions.

B. Density Based Standard Cell Legalization

After resolving overlaps among the macro blocks, we fix their positions for all subsequent steps of placement and treat them as placement blockages. Each row in the placement region is then fragmented into segments due to the placement blockages. The aim of the density based standard-cell legalizer is to satisfy the segment capacities as well as the placement congestion constraint and assign cells to legal locations within the segments. This is done as follows.

B.1 Selective Bin-based Standard Cell Movement

First, the placement region is binned with the height of each bin being equal to the standard-cell height and its width being equal to around $4\times$averageCellWidth. We call this bin structure as the Regular Bin Structure (RBS). The utilization of each bin is then determined. Simultaneously, the utilization of each segment in the placement region is also determined. The utilization of a segment is defined as the total width of all the cells within the segment. If the total width of all the cells within a segment is greater than the segment width, it is considered to be above capacity.

Based on the segment utilizations and placement blockages, we construct a *move map* of the entire placement region. For each bin in the RBS this map has a value of either 1 (for moving cells into or out of this bin) or 0 (for not moving cells into or out of this bin). The values are assigned as follows: For bins that completely overlap placement blockages we assign a value of 0 as we do not want cells to be moved on top of the blockage. Also, if the the utilization of a particular segment is greater than the *placement_target_density* then, a small region of bins in and around the current segment is assigned a value of 1 for move based legalization to be performed on these bins. This is depicted in Figure 7 where there are two segments that are above capacity (shown by the diagonal lines). Then, we turn on move based legalization for only a small region of bins around the segments (shown by the dotted regions).

For moving the cells among the bins we use a technique that is quite similar to the Iterative Local Refinement. The difference being that the score for a move during the legalizer is a weighted sum of three components: The first being the half-perimeter wirelength reduction for the move. The second being a function of the utilization of the source and target bins. The third being a weighted difference of the *move map* values for the source and target bins. Since the legalization technique is mainly used to even out the placement and bring all the segments within capacity, a higher weight is assigned to the second and third components.

Once all the segments are brought to within capacity, we assign the cells to legal positions within each segment. The key advantages of the selective bin-based legalizer is that it does not disturb the global placement solution by a significant amount. Secondly, the bin-based legalizer also distributes the cells evenly within segments. This helps to satisfy placement congestion constraints.

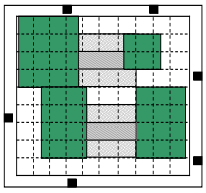

Fig. 7. Selective Bin-based Standard Cell Movement.

C. Detailed Placement

The aim of the detailed placement stage is to further reduced the wirelength of the placement. For detailed placement, we adopt a modified version of the *FastDP* [22] detailed placer that can handle placement congestion constraints.

VI. Experimental Results

Our algorithm was tested on the ISPD-2005 Placement Benchmarks [19] and the placement congestion constrained ISPD-2006 Placement Benchmarks [20]. These benchmark suites have been derived from industrial ASIC designs with circuit sizes ranging from 210K to 2.48M placeable objects. In addition, the ISPD-2006 benchmark suite is also placement congestion constrained, with a specific *placement_target_density* assigned to each circuit.

For runtime comparison we ran two state-of-the-art placers *APlace 2.0* [15,16] and *mPL6* [5] on the ISPD-2005 Placement Benchmarks. *APlace 2.0* is a faster version of the placer used in the ISPD 2005 Placement contest [14] and *mPL6* comprises of enhanced versions of the placers described in [4] and [8]. All experiments are run on a 2.5 GHZ AMD Opteron 252 machine with 8 GB RAM.

In Table I, we compare our placer with *APlace 2.0* and *mPL6* on the ISPD-2005 benchmark suite. It can be seen that we are on average, 3% better in terms of wirelength as compared to *APlace 2.0* and $14.9X$ faster. As compared to *mPL6* we have 4% higher wirelength and are $4.4X$ faster.

In Table II we compare our placement results with that of other placers reported during the ISPD 2005 placement contest. It should be noted that for the contest, all the placers were given the benchmarks in advance and there was no limit on the CPU time required to get the best possible results on the individual circuits. From Table II, the contest version of *APlace* is on average 5.6% better than our placer in terms of half-perimeter wirelength. In [15] the authors report that the entire benchmark set takes 113.2 hrs on a 1.6 GHZ machine. Since they do not give any other machine specification we are unable to make runtime comparisons with the contest version of *APlace*. It can also be seen that our results are better than the reported results of all the other placers.

In Table III we compare our placement results with that of other placers reported during the ISPD 2006 placement contest using the same scoring function as the contest. On average, we have 3% higher score than the best reported results using the ISPD-2006 contest scoring function. Looking at individual results, on 4 of the 8 benchmarks we are better than or comparable to the best reported results during the placement contest.

Table IV gives the runtime comparison of our placer with other placers in the ISPD 2006 placement contest. On average,

TABLE I
HALF-PERIMETER WIRELENGTH AND RUNTIME COMPARISON OF OUR PLACER WITH *mPL6* AND *APlace2.0* ON THE ISPD-2005 BENCHMARK SUITE.

Circuit	**Half-Perimeter Wirelength**			**Runtime**		
	Our	$\frac{mPL6}{our}$	$\frac{APlace2.0}{our}$	Our (sec)	$\frac{mPL6}{our}$	$\frac{APlace2.0}{our}$
adaptec2	94908408	0.97	1.01	517	4.35	17.72
adaptec4	204785632	0.95	1.02	1164	5.81	21.59
bigblue1	95458136	1.01	1.05	506	5.41	16.85
bigblue2	157384512	0.97	0.98	1309	5.96	15.31
bigblue3	387811008	0.89	1.06	4963	2.12	7.60
bigblue4	833668928	0.99	1.05	8823	2.73	10.44
		0.96	**1.03**		**4.40**	**14.92**

TABLE II
HALF-PERIMETER WIRELENGTH COMPARISON OF OUR PLACER WITH OTHER ACADEMIC PLACERS ON THE ISPD-2005 BENCHMARK SUITE.

Placer	**Circuit**						**Average**
	adaptec2	adaptec4	bigblue1	bigblue2	bigblue3	bigblue4	
APlace	0.92	0.92	0.99	0.91	0.92	1.00	0.944
Our Placer	1.00	1.00	1.00	1.00	1.00	1.00	1.000
mFAR	0.96	0.93	1.02	1.07	0.98	1.05	1.004
Dragon	1.00	0.98	1.07	1.01	0.98	1.08	1.022
mPL	1.02	0.98	1.03	1.10	0.95	1.08	1.029
FastPlace	1.14	1.00	1.06	1.08	1.18	1.07	1.088
Capo	1.05	1.03	1.13	1.09	0.99	1.32	1.103
NTUplace	1.06	1.01	1.12	1.21	1.06	1.38	1.140
Fengshui	1.30	1.65	1.20	1.81	1.21	1.25	1.403
Kraftwerk	1.66	1.72	1.57	2.05	1.69	1.68	1.728

TABLE III
OUR PLACER COMPARED TO OTHER ACADEMIC PLACERS ON THE ISPD-2006 BENCHMARK SUITE USING THE ISPD-2006 PLACEMENT CONTEST SCORING FUNCTION.

Placer	**Circuit**								**Average**
	adaptec5	newblue1	newblue2	newblue3	newblue4	newblue5	newblue6	newblue7	
Kraftwerk	1.01	1.19	1.00	1.00	1.01	1.04	1.00	1.00	1.03
mPL6	1.00	1.06	1.07	1.17	1.00	1.02	1.00	1.00	1.04
NTUplace2	1.02	1.00	1.07	1.16	1.03	1.00	1.04	1.07	1.05
Our Placer	1.12	1.17	0.98	1.15	0.99	1.13	1.00	0.96	1.06
mFAR	1.09	1.23	1.09	1.16	1.09	1.13	1.03	1.04	1.11
APlace3	1.26	1.20	1.05	1.13	1.35	1.21	1.06	1.05	1.16
Dragon	1.08	1.21	1.29	1.90	1.05	1.13	1.03	1.23	1.24
FastPlace	1.82	1.22	1.02	1.37	1.35	1.76	1.04	1.05	1.33
DPlace	1.26	1.55	1.77	1.36	1.14	1.35	1.23	1.25	1.36
Capo	1.16	1.57	1.64	1.44	1.22	1.28	1.32	1.46	1.39

TABLE IV
RUNTIME RESULTS OF OUR PLACER COMPARED TO OTHER ACADEMIC PLACERS ON THE ISPD-2006 BENCHMARK SUITE.

Placer	**Circuit**								**Average**
	adaptec5	newblue1	newblue2	newblue3	newblue4	newblue5	newblue6	newblue7	
Our Placer	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Kraftwerk	1.72	1.94	0.97	0.41	3.04	2.46	1.68	1.64	1.73
mPL6	4.33	3.86	5.89	4.35	6.37	4.09	3.77	6.22	4.86
NTUplace2	5.49	3.70	4.28	2.98	8.18	6.78	4.34	4.70	5.06
mFAR	3.60	4.35	2.80	1.33	6.97	3.79	3.81	4.27	3.86
APlace3	10.61	7.37	5.35	5.60	16.42	10.87	9.13	12.02	9.67
Dragon	1.18	1.69	1.58	0.52	1.63	1.17	1.21	2.17	1.39
FastPlace	2.12	0.88	1.00	1.09	1.52	2.06	1.30	1.45	1.43
DPlace	1.51	1.76	6.18	0.46	1.80	1.51	1.26	2.08	2.07
Capo	5.09	4.39	5.45	2.72	7.59	6.91	5.79	12.04	6.25

the runtime of our placer is the least among all the placers.

VII. Conclusions

In this paper we describe an efficient and scalable quadratic placer for large-scale standard cell and mixed-size circuits. It is based on a multilevel global placement framework and incorporates an improved Iterative Local Refinement Technique that can handle placement blockages as well as placement congestion constraints. We also describe an efficient density based standard-cell legalization scheme.

The current implementation produces competitive results compared to other state-of-the-art academic placers on various benchmark circuits but at a much lesser runtime. Such an ultra-fast placer is very much needed in present day iterative physical synthesis flows to achieve timing closure without a significant runtime overhead.

References

[1] A. R. Agnihotri, S. Ono, C. Li, M. C. Yildiz, A. Khatkhate C.-K. Koh, and P. H. Madden. Mixed block placement via fractional cut recursive bisection. *TCAD*, 24(5):748–761, May 2005.

[2] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection produce routable placements. In *Proc. DAC*, pages 477–482, 2000.

[3] T. Chan, J. Cong, T. Kong, and J. Shinnerl. Multilevel optimization for large-scale circuit placement. In *Proc. ICCAD*, pages 171–176, 2000.

[4] T. Chan, J. Cong, and K. Sze. Multilevel generalized force-directed method for circuit placement. In *Proc. ISPD*, pages 185–192, 2005.

[5] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie. mPL6: Enhanced multilevel mixed-size placement. In *Proc. ISPD*, pages 212–214, 2006.

[6] C. C. Chang, J. Cong, and X. Yuan. Multi-level placement for large-scale mixed-size IC designs. In *Proc. ASPDAC.*, pages 325–330, 2003.

[7] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang, and Y.-W. Chang. NTUplace: A ratio partitioning based placement algorithm for large-scale mixed-size designs. In *Proc. ISPD*, pages 236–238, 2005.

[8] J. Cong and M. Xie. A robust detailed placement for mixed-size ic designs. In *Proc. ASPDAC*, pages 188–194, 2006.

[9] H. Eisenmann and F. Johannes. Generic global placement and floorplanning. In *Proc. DAC*, pages 269–274, 1998.

[10] H. Etawil, S. Arebi, and A. Vannelli. Attractor-repeller approach for global placement. In *Proc. ICCAD*, pages 20–24, 1999.

[11] B. Hu and M. Marek-Sadowska. FAR: Fixed-points addition and relaxation based placement. In *Proc. ISPD*, pages 161–166, 2002.

[12] B. Hu and M. Marek-Sadowska. Fine granularity clustering for large scale placement problems. In *Proc. ISPD*, pages 67–74, 2003.

[13] B. Hu and M. Marek-Sadowska. Multilevel fixed-point-addition-based VLSI placement. *TCAD*, 24(8):1188–1203, August 2005.

[14] A. B. Kahng, S. Reda, and Q. Wang. APlace: A general analytic placement framework. In *Proc. ISPD*, pages 233–235, 2005.

[15] A. B. Kahng, S. Reda, and Q. Wang. Architecture and details of a high quality, large-scale analytical placer. In *Proc. ICCAD*, pages 890–897, 2005.

[16] A. B. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. *TCAD*, 24(5):734–747, May 2005.

[17] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *TCAD*, 10(3):356–365, March 1991.

[18] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI module placement based on rectangle-packing by the sequence pair. *TCAD*, 15(12):1518–1524, December 1996.

[19] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. The ISPD2005 placement contest and benchmark suite. In *Proc. ISPD*, pages 216–220, 2005.

[20] G.-J. Nam. ISPD 2006 placement contest: Benchmark suite and results. In *Proc. ISPD*, pages 167–167, 2006.

[21] G.-J. Nam, S. Reda, C. J. Alpert, P. G. Villarrubia, and A. B. Kahng. A fast hierarchical quadratic placement algorithm. *TCAD*, 25(4):678–691, April 2006.

[22] M. Pan, N. Viswanathan, and C. Chu. An efficient and effective detailed placement algorithm. In *Proc. ICCAD*, pages 48–55, 2005.

[23] C. Sechen and A. L. Sangiovanni-Vincentelli. TimberWolf 3.2: A new standard cell placement and global routing package. In *Proc. DAC*, pages 432–439, 1986.

[24] W.-J. Sun and C. Sechen. Efficient and effective placement for very large circuits. *TCAD*, 14(5):349–359, 1995.

[25] T. Taghavi, X. Yang, B.-K. Choi, M. Wang, and M. Sarrafzadeh. Dragon2005: Large-scale mixed-size placement tool. In *Proc. ISPD*, pages 245–247, 2005.

[26] N. Viswanathan and C. C.-N. Chu. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. *TCAD*, 24(5):722–733, May 2005.

[27] N. Viswanathan, M. Pan, and C. Chu. Fastplace 2.0: An efficient analytical placer for mixed-mode designs. In *Proc. ASPDAC*, pages 195–200, 2006.

[28] M. Wang, X. Yang, and M. Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. ICCAD*, pages 260–263, 2000.