A Robust Detailed Placement for Mixed-Size IC Designs

Jason Cong and Min Xie
Computer Science Department, University of California, Los Angeles
California 90095 USA
e-mail: {cong,xie}@cs.ucla.edu

Abstract— The rapid increase in IC design complexity and wide-spread use of intellectual-property (IP) blocks have made the so-called mixed-size placement a very important topic in recent years. Although several algorithms have been proposed for mixed-sized placements, most of them primarily focus on the global placement aspect. In this paper we propose a three-step approach, named XDP, for mixed-size detailed placement. First, a combination of constraint graph and linear programming is used to legalize macros. Then, an enhanced greedy method is used to legalize the standard cells. Finally, a sliding-window-based cell swapping is applied to further reduce wirelength. The impact of individual techniques is analyzed and quantified. Experiments show that when applied to the set of global placement results generated by APlace [1], XDP can produce wirelength comparable to the native detailed placement of APlace, and 3% shorter wirelength compared to Fengshui 5.0 [2]. When applied to the set of global placements generated by mPL6 [3], XDP is the only detailed placement that successfully produces legal placement for all the examples, while APlace and Fengshui fail for $4/9$ and $1/3$ of the examples. For cases where legal placements can be compared, the wirelength produced by XDP is shorter by 3% on average compared to APlace and Fengshui. Furthermore, XDP displays a higher robustness than the other tools by covering a broader spectrum of examples by different global placement tools.

I. Introduction

Placement is a critical step in VLSI circuit design because it determines the interconnect more than any other step in physical design. The rapid increase in IC design complexity, and the wide-spread use of intellectual-property (IP) blocks have made the so-called mixed-size placement a very important topic in recent years.

Formally, mixed-size placement solves the following problem: Given a rectangular region R and a netlist N, place standard cells and macros within the region without overlap. The optimization objective can be the minimization of total half-perimeter wirelength, routed wirelength, performance, power, etc.

A number of algorithms have been proposed for mixed-size placement, and they can be divided into two classes. The first class of algorithms removes the overlap between placeable objects during global placement, leaving detailed placement with only the task of further wirelength reduction. In this class is a two-pass approach that combines a recursive min-cut-based placer, Capo, and a fixed-outline floorplanner, Parquet [4]. Macros are first shredded into pieces and placed by the standard cell placer. The locations of macros are subsequently derived by reassembling the component pieces, and residual overlap is removed through the floorplanner. The second pass places standard cells with all macros fixed. A top-down "correct-by-construction" approach was proposed in [5] that may invoke Parquet many times in intermediate levels. Another algorithm in this class is mPG-ms [6], which uses simulated annealing to gradually legalize macros and fix them in the intermediate levels of the multilevel optimization. Dragon2005 [7] is a two-pass simulated annealing-based placer. Standard cells and macros are placed together in the first pass. In the second pass, the macros are held fixed, and the standard cells are placed again. To further reduce wirelength, it shifts cells when swapping cells from different rows during detailed placement. The most successful algorithm in this class is the recently published PolarBear [8], which combines recursive min-cut with an extra legalization step for every placement subproblem generated by the partitioning. With white space at 5%, PolarBear produces placement with wirelength 10% shorter than Capo 9.3, while Fengshui 5.1 often fails to find legal solutions.

The second class of algorithms may leave overlaps between macros and cells after global placement. Most analytical placers, including Kraftwerk [9], BonnPlace [10, 11, 12], Aplace [1], FDP [13], mPL5 [14], UPlace [15], and some min-cut-based placers, such as Fengshui [16, 2], belong to this category. In this case, the detailed placement is expected to remove the overlap, as well as reduce the wirelength. BonnPlace uses a quadratic programming-based approach coupled with quadri-section [10]. To legalize macros, a bottom-up branch and bound search with linear programming (LP) is proposed. Standard cells are evened out between placement regions with a min-cost-max-flow formulation. Further wirelength reduction is achieved by solving a LP formulation on each row. Fengshui [16, 2] uses a greedy scheme that considers simultaneously perturbation of macros and wirelength minimization for legalization. Windows spanning multiple rows for cell permutation are used for wirelength reduction. Domino [17] iteratively improves wirelength by shredding cells into uniform pieces and solving a min-cost-max-flow formulation. UPlace [15] applies zone refinement for both legalization and wirelength reduction purposes. The objective it considers combines wirelength and zone height.

Most macro legalization schemes used in the second class suffer from two limitations. First, they may not produce a legal placement in the end. Second, they may cause a large perturbation to the global placement during legalization, resulting in longer wirelength. Fig. 1 and Fig. 2 show an example of applying Fengshui's legalization scheme on a global placement generated by an analytical placer, mPL6 [3]. The legalized wirelength increases by more than 10% compared to global placement wirelength.

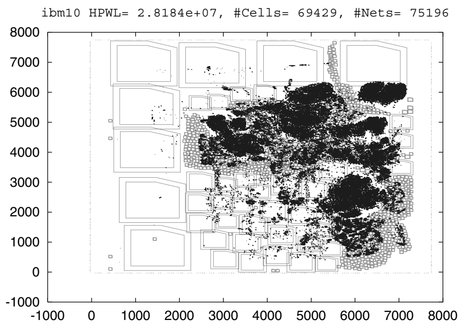

Fig. 1. An example global placement generated by mPL6 [3].

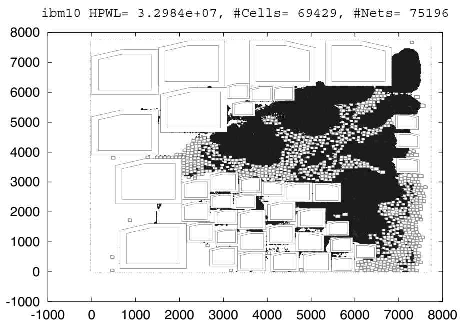

Fig. 2. Legalization by applying Fengshui's greedy method. The wirelength increases by more than 10%.

In this paper we present XDP, a three-step approach for mixed-size detailed placement. First, a combination of constraint graph and linear programming is used to legalize macros. Then, an enhanced greedy method is used to legalize the standard cells. Finally, a sliding-window-based cell swapping is applied to further reduce wirelength. The impact of individual techniques is analyzed and quantified. Experiments show that when applied to the set of global placement results generated by APlace [1], XDP can produce wirelength comparable to the native detailed placement of APlace, and 3% shorter wirelength when compared to Fengshui 5.0 [2].[1] When applied to the set of global placements generated by mPL6 [3], XDP is the only detailed placement that successfully produces legal placement for all the examples, while APlace and Fengshui fail for $4/9$ and $1/3$ of the examples. For cases where legal placements can be compared, the wirelength produced by XDP is shorter by 3% on average. Furthermore, XDP displays a higher robustness than the other tools by covering a broader spectrum of examples generated by different global placement tools.

The remainder of this paper is organized as follows: Section II presents each step of our detailed placement algorithm: Section II.A describes the macro legalization step, Section II.B describes the cell legalization step, and Section II.C presents the wirelength reduction step. Section III presents experiment results, and Section IV provides conclusions and future work.

[1]A number of other placement tools have mixed-size capability. Among them, only APlace and Fengshui exports their detailed placement capability.

II. Mixed-Size Detailed Placement

II.A. Macro Legalization

The first step of our algorithm removes the overlap between macros in the global placement, which can be formulated as the following problem:

Given a set of rectangular blocks, $M = \{m_1, m_2 ... m_n\}$, pack the blocks within a rectangular region R without overlap. The objective is to minimize the perturbation, i.e., total movement of the blocks from their original locations.

Before a detailed description, we introduce some notations here.

Let m_i be the ith macro. Its center coordinate in global placement is (x_i, y_i). Its width and height is w_i and h_i respectively. The coordinate of m_i after macro legalization is denoted as (x'_i, y'_i).

Let the lower left corner of the placement region R be $(0, 0)$, the top right corner be (W, H).

Let G_h be a directed acyclic graph (DAG). For each macro m_i, v_{h_i} is the corresponding node in G_h. G_h has a source node v_{h_s} and a sink node v_{h_t}.

To represent the constraint that m_i should be on the left of m_j, a directed edge from v_{h_i} to v_{h_j} will be inserted into G_h. The edge weight is set to be $\frac{w_i+w_j}{2}$. Our graph definition is similar to those widely used in floorplaning [18].

For each node in G_h, we calculate two values, $L(v_{h_i})$ and $R(v_{h_i})$, using Equation 1.

$$\begin{aligned} L(v_{h_s}) &= 0 \\ L(v_{h_j}) &= \max(L(v_{h_i}) + weight(e_{ij}))\ \forall e_{ij} \in G_h \\ R(v_{h_t}) &= \max(L(v_{h_t}), W) \\ R(v_{h_i}) &= \min(R(v_{h_j}) - weight(e_{ij}))\ \forall e_{ij} \in G_h \end{aligned} \tag{1}$$

For each edge e_{ij} in G_h, we calculate $slack(e_{ij})$ using Equation 2.

$$slack(e_{ij}) = R(v_{h_j}) - L(v_{h_i}) - weight(e_{ij})\ \forall e_{ij} \in G_h \tag{2}$$

It can be seen that the definitions are analogous to those defined for timing analysis. For each node, we also calculate value $disp(v_{h_i})$ using Equation 3. This is to model the potential displacement for each macro.

$$disp(v_{h_i}) = \begin{cases} L(v_{h_i}) - x_i\ if L(v_{h_i}) \geq x_i \\ x_i - R(v_{h_i})\ if R(v_{h_i}) \leq x_i \\ 0\ otherwise \end{cases} \tag{3}$$

In the end the total displacement of a constraint graph is defined using Equation 4.

$$disp\,(G_h) = \sum_{v \in G_h} disp(v) \tag{4}$$

Similarly, we can define G_v and the corresponding values for the vertical direction.

II.A.1. Initial Constraint Graph Generation

Given a global placement, we examine each pair of macros m_i and m_j, and create a constraint edge between them. The edge can be either horizontal or vertical, depending on the relative locations of m_i and m_j. Fig. 3 gives three relative locations that we consider. The type of the constraint edges is such that the macros are given the most flexibility in the constraint graphs. The edge weights are assigned accordingly.

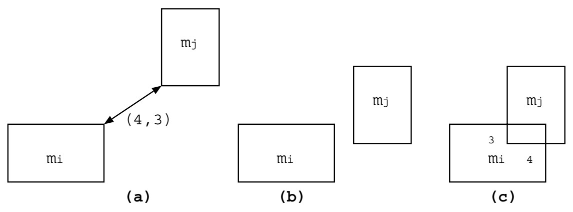

Fig. 3. Three types of relative macro locations which are used to determine the constraint type between each pair of macros. Constraint edge weight is assigned accordingly.

II.A.2. Constraint Graph Adjustment

After the constraint graph construction, we traverse each graph and calculate the longest path. [2] If the longest path exceeds the chip dimension, some of the edges need to be adjusted to reduce the longest path. By adjustment we mean change an edge's direction from horizontal to vertical while keeping its head and tail,[3] or vice versa. In the following discussion, we assume the longest path in G_h exceeds the chip width, while the path in G_v is within the chip height.[4]

Formally, we need to solve the following subproblem:

Given G_h with $L(v_{h_t})$ greater than r, select a subset of constraint edges in G_h to move the G_v, so that the $L(v_{h_t})$ after adjustment is reduced, subject to the constraint that $L(v_{v_t})$ after adjustment should not be greater than H. The objective is to minimize $disp(G_v)$ after the adjustment.

This problem needs to be addressed since identifying the right set of edges for adjustment may not be trivial under certain circumstances. Fig. 4 presents a global placement of macros with dimensions. The dimension of the placement region is 25 × 10. Fig. 5(a) presents the G_h corresponding to Fig. 4. Edges in the critical path are highlighted with weights. Since macro 2 and 3 have the same width, we have two converging paths with the same length. Fig. 5(b) gives the corresponding G_v. A straightforward method that examines one edge at a time will not pick e_{12}, e_{13}, e_{24} or e_{34} for adjustment, since the final longest path will not change. This leaves us with only the choice of e_{45} or e_{57}. However, adjusting either of them will make the longest path on the Y direction exceed the chip height.

To solve this problem, we extract a subgraph of G_h, consisting of edges with zero slack. This graph is similar to that used for timing optimization in logic synthesis [19, 20, 21, 22]. We name this subgraph the zero-slack network of G_h. According to this network, another DAG, G_c, will be constructed. Each edge and node in the zero-slack network have a corresponding counterpart in G_c. For an edge e_{ij} in the network, if adjusting causes the longest path in the Y direction to exceed the chip height, the corresponding edge capacity in G_c will be set to $+\infty$. Otherwise, the capacity is set using Equation 5.

$$\max(y_i - R(v_j) + \tfrac{h_i+h_j}{2}, 0) + \max(L(v_i) + \tfrac{h_i+h_j}{2} - y_j, 0) \tag{5}$$

The first component is the potential perturbation on m_i's x coordinate because of the constraint edge under consideration. The second component is the potential perturbation on m_j's x coordinate because of the constraint edge adjustment. To reduce the complexity, we use $L(v_i)$ and $R(v_j)$ before the adjustment, rather than those values after the adjustment. All edges incident on v_{h_s} or v_{h_t} will be assigned a capacity $+\infty$. It can be seen that the definition of edge capacity is set to encourage choosing edges with potentially large slack on the orthogonal direction. A min-cut is then calculated on G_c. For each edge in the cut, the corresponding edge in G_h will be adjusted. Compared to [23], instead of permuting the sequence pair and evaluating the impact of the constraint graphs, we operate directly on the graphs, giving us more flexibility and finer granularity in the operations. Furthermore, our basic operations are more targeted to meeting the packing constraints.

Fig. 5(c) gives the G_c for G_h with edge capacity assigned. The solution for this instance is the min-cut formed by e_{12} and e_{13}. Adjusting this increases the longest path on the Y direction to 9, but is still within the chip height. Fig. 6 gives the final constraint graphs after the adjustment.

The adjustment process iterates until the longest paths in both graphs are shorter than the chip dimension, indicating we have found a set of non-overlapping constraints that can be satisfied. Empirically, it terminates after a few iterations. [5]

II.A.3. Macro Coordinate Determination

The constraint graphs and the subsequent adjustment are essentially used to find a set of non-overlapping constraints that can be satisfied. Our next stage is to determine the exact locations of the macros so that the total perturbation on macros is minimized. This can be formulated as the following linear programming problem:

$$\begin{array}{lll}
\min \sum_{i=1}^{n} (w_{x_i} \times dx_i + w_{y_i} \times dy_i) & & \\
s.t. -dx_i \le x'_i - x_i \le dx_i & & \\
-dy_i \le y'_i - y_i \le dy_i & & \\
x'_j - x'_i \ge \frac{w_i+w_j}{2} & \text{if } \exists e_{ij} \in G_h & \\
y'_j - y'_i \ge \frac{y_i+y_j}{2} & \text{if } \exists e_{ij} \in G_v & \\
\frac{w_i}{2} \le x'_i \le W - \frac{w_i}{2} & & \\
\frac{h_i}{2} \le y'_i \le H - \frac{h_i}{2} & &
\end{array} \tag{6}$$

Here, the dx_i and dy_i are used to quantify the perturbation of m_i. w_{x_i} and w_{y_i} are positive weights that can be set to either one or the number of connections on each macro. The next two inequalities are derived from the edges in G_h and G_v. The last two constraints force the macros to stay with the chip

[2]In case the graph thus constructed has cycles, we first derive a sequence-pair representation of the macros, and construct the constraint graphs according to the representation.

[3]We also investigated the alternative of swapping the head and tail of the constraint edge, depending on the global placement. Overall, we do not observe significant improvement in the final quality.

[4]In case the longest path in G_v exceeds the chip height already, we temporarily lift the chip height to be the same as the longest path in G_v.

[5]It is possible that the iterations may not find a feasible solution. In reality, we have not observed any instance of failure on the example we tested, even with only 2 to 3% of white space.

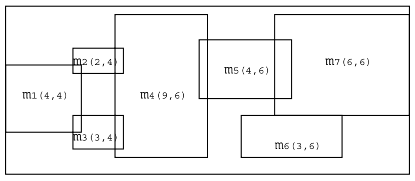

Fig. 4. An example of macros with overlap.

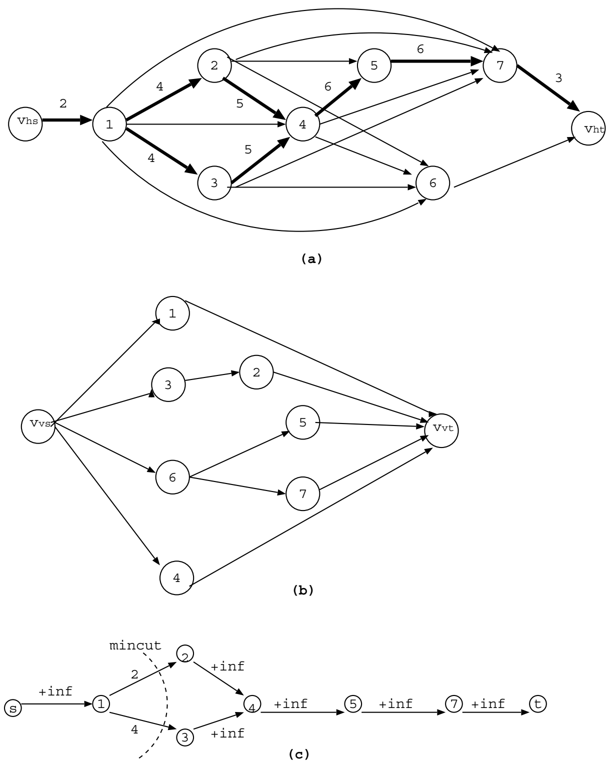

Fig. 5. (a) Constraint graph G_h. (b) Constraint graph G_v. (c) Corresponding G_c with edge capacity assigned. The min-cut identifies the set of edges which will be transformed from horizontal to vertical.

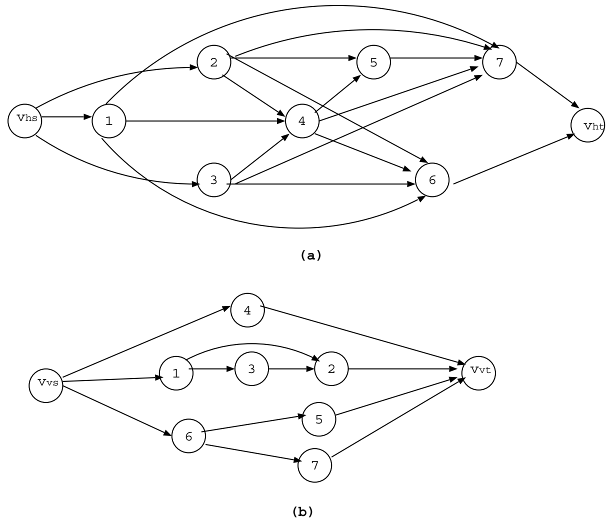

Fig. 6. (a) Constraint graph G_h after adjustment. (b) Constraint graph G_v after adjustment.

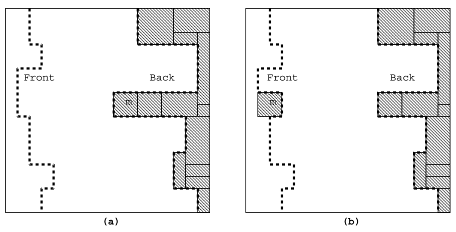

Fig. 7. (a) Front-end designates the leftmost site that can be occupied without overlap with already legalized objects. Back-end designates the rightmost site that can be occupied without overlapping with macros that have not been legalized yet. Back-end contour is initialized as the contour of macros if they are packed to the right boundary. (b) In addition to updating the front-end contour, the back-end contour of rows crossed by a macro will be updated after the macro is legalized.

region. Although the formulation is similar to that in [10, 24], we do not go through the bottom up branch and bound process, as proposed in [10]. Our constraint-graph-based method helps to prune the search space by following the relative order in the global placement. Furthermore, we only solve the LP after a legal packing of macros is guaranteed, while a LP may be tried for every possible combination of non-overlapping constraints [10] in the worst case. The objective can also be enhanced to consider wirelength by the formulation of Mongrel [25], as in [26]. To solve the LP, we used a public domain interior-point LP solver, BPMPD [27].

II.B. Cell Legalization

Following macro legalization, the second step removes the overlap between standard cells. This step is to solve the following problem:

Given a placement where overlap only exists between cells, or cells and macros, remove the overlap between all objects and obtain a legal placement. The objective is still minimization of wirelength.

A greedy heuristic has been proposed for this purpose in [16], as an extension of [28] for mixed-size placement. A front-end contour designating the leftmost empty site on each row is maintained. Movable objects are traversed in ascending order of the x coordinate. The location of each object is determined by considering the combination of incident wirelength and displacement penalty. The front-end contour is updated after each object is placed. Although it gives a satisfactory result, this method can not guarantee that all the macros can fit within the chip boundary when the legalization finishes. To mitigate this drawback, the global placement of Fengshui takes a conservative approach, packing the macros and cells very tightly to increase the chance of success during legalization [16]. Another alternative by APlace is to iteratively “squeeze” the cell locations and restart cell legalization until a legal soltion is obtained [29]. However, as we will show in Section III, this strategy may not find a legal solution either.

We enhanced this method by introducing a back-end contour, which is initialized as the left contour of macros if they are packed to the right. Fig. 7(a) illustrates the initialization of a back-end contour.

Before legalization, all the movable objects are sorted in ascending order of their left boundary. The placeable objects are examined one at a time. If the object is a cell, we scan each row and pick the site between the two contours that gives the shortest wirelength for the nets connected with it. The front-end on the target row is updated. If no site can be found for a cell, it will be temporarily put on its original location with its physical dimension ignored. This will result in cell area overflow in certain regions of the chip, which will be dealt with in the additional step that follows. If the object is a macro, it will only be considered for movement between the interval determined by the two contours. This restriction guarantees legality of macros obtained from II.A. An additional step after each macro legalization is to update the back-end contour of rows that the macro crosses, as shown in Fig. 7(b).

Depending on the global placement, if cell area overflow remains in certain part of the chip, we partition the chip into regions, and use the min-cost-max-flow formulation in [10, 12] to even out cells between different regions. Each region is represented as a node in a graph. A bi-directional edge is set up between each pair of adjacent regions, as illustrated by Fig 8. The node capacity is the difference between the region area and the total cell area in the region. The unit cost of an edge is the center-to-center distance between the two regions it connects. Since the edge cost is positive, the final solution has no cycles. A dynamic programming-based method is used to select the cells to move between regions. The occurrence of this situation depends partly on the global placement. Among the 18 examples we tested in Section III, five of them still need this adjustment. However, the wirelength usually increases after this adjustment.

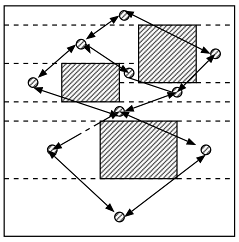
Fig. 8. Network flow based formulation to even out cells.

II.C. Further Wirelength Reduction

After a legal placement is obtained, the last step of our algorithm is to further reduce the wirelength. Here, we use a window spanning a single row or multiple rows and slide it across the chip. We enumerate all the possible configurations and pick the one giving the shortest wirelength of nets connected with the cells. After permutation is finished, the window is slid by half its width. This process is iterated until no further wirelength reduction is possible. This is the same process as that described in [16].

III. Experiment Results

In this section we evaluate the effectiveness of our detailed placement, XDP, using ICCAD04-MS. These circuits were introduced in [5] to test the placer's capability to handle macros of different shapes and aspect ratios. Table I describes the characteristics of ICCAD04-MS. All the experiments were done on an Intel Xeon processor with 2.4GHz and 2GB memory.

TABLE I
Characteristics of ICCAD04-MS. All macros in the circuits are movable.

circuit	#cell	#macro	#pad	#net	#row	utilization
ibm01	12507	246	246	14111	144	80%
ibm02	19343	271	259	19584	203	80%
ibm03	22854	290	283	27401	219	80%
ibm04	27221	295	287	31970	213	80%
ibm05	28147	0	1201	28446	148	80%
ibm06	32333	178	166	34826	204	80%
ibm07	45640	291	287	48117	240	80%
ibm08	51024	301	286	50513	256	80%
ibm09	53111	253	285	60902	293	80%
ibm10	68686	786	744	75196	482	80%
ibm11	70153	373	406	81454	322	80%
ibm12	70440	651	637	77240	425	80%
ibm13	83710	424	490	99666	350	80%
ibm14	147089	614	517	152772	375	80%
ibm15	161188	393	383	186608	422	80%
ibm16	182981	458	504	190048	507	80%
ibm17	184753	760	743	189581	454	80%
ibm18	210342	285	272	201920	406	80%

III.A. The Effectiveness of XDP

First, we compared the effectiveness of XDP with the detailed placement of Fengshui 5.0 [2], and APlace [1]. We generated two sets of global placements using APlace and mPL6.[6]

Table II lists the overall comparison of the three algorithms on global placements generated by APlace. Column "FWL" gives the final wirelength. Column "RT(s)" gives the runtime.[7] XDP produces results comparable to the native detailed placement of APlace, and 3% shorter wirelength compared to Fengshui. Table III lists the overall comparison of the three algorithms on global placements generated by mPL6. XDP is the only detailed placer that produces legal placement for all the examples, while Fengshui and APlace can not produce meaningful results for $1/3$ and $4/9$ of the examples, respectively. Furthermore, for those examples where legal results can be compared, XDP produces wirelength 3% shorter than APlace and Fengshui.

III.B. Impact of Individual Techniques

Next, we analyze the impact of each heuristic used in XDP by experimenting with global placements generated by mPL6. We first evaluate the effectiveness of LP during macro coordinate assignment. For comparison purpose, we implemented a greedy method for macro assignment. The nodes in the constraint graph are traversed in topological order, and their coordinates are chosen as the closest location allowed by the constraints. Note that once a macro coordinate is assigned, it will have a ripple effect on its successor's leftmost/lowest allowable coordinate.

Table IV summarizes the overall impact of LP. The column "GPWL" gives the global placement wirelength. The column "Greedy" corresponds to the greedy method. Column "LP"

[6]We did not obtain meaningful global placements from Fengshui 5.0 due to orientation specification issues.

[7]N/A or Overlap means the placer either crashed or produced illegal placement.

TABLE II
COMPARISION OF DETAILED PLACEMENT ALGORITHMS USING GLOBAL PLACEMENTS GENERATED BY APLACE.

circuit	GPWL	Aplace		Fengshui		XDP	
		FWL	RT(s)	FWL	RT(s)	FWL	RT(s)
ibm01	2.16E+06	2.14E+06	24	Overlap		2.08E+06	24
ibm02	4.84E+06	4.65E+06	50	N/A		4.65E+06	53
ibm03	6.95E+06	6.71E+06	58	N/A		6.73E+06	52
ibm04	7.57E+06	7.57E+06	62	N/A		7.48E+06	64
ibm05	9.83E+06	9.69E+06	54	9.84E+06	59	9.59E+06	72
ibm06	6.38E+06	6.02E+06	76	Overlap		6.11E+06	73
ibm07	1.04E+07	1.00E+07	111	Overlap		9.94E+06	119
ibm08	1.29E+07	1.25E+07	131	Overlap		1.24E+07	152
ibm09	1.26E+07	1.21E+07	154	Overlap		1.21E+07	140
ibm10	3.11E+07	2.88E+07	296	Overlap		2.90E+07	324
ibm11	1.96E+07	1.87E+07	215	Overlap		1.87E+07	199
ibm12	3.50E+07	3.34E+07	279	Overlap		3.38E+07	260
ibm13	2.34E+07	2.28E+07	279	Overlap		2.30E+07	239
ibm14	3.74E+07	3.59E+07	445	Overlap		3.55E+07	413
ibm15	4.88E+07	4.68E+07	648	Overlap		4.67E+07	565
ibm16	5.83E+07	5.45E+07	798	Overlap		5.43E+07	666
ibm17	6.69E+07	6.57E+07	735	Overlap		6.53E+07	667
ibm18	4.48E+07	4.20E+07	706	Overlap		4.17E+07	672
Avg.		1.00	1.00	1.03	0.73	1.00	0.98

corresponds to the LP-based method. Column "FWL" gives the final wirelength. Column "RT(s)" gives the runtime. The LP-based strategy helps to reduce the final wirelength by 1%, and the runtime by 2%.

In Table V we compare two alternative strategies in the cell legalization step. Column "Movable" corresponds to the strategy that allows macros to move when necessary, but only horizontally, as described in Section II.B. Column "Fixed" corresponds to the alternative which fixes macros after the macro legalization step. We list both the final wirelength and the runtime in columns labeled "FWL" and "RT(s)." It can be seen that giving flexibility to macros helps to reduce the final wirelength by 4%. The runtime for the movable strategy is longer by 11%. This suggests that it is useful to give flexibility to macros during legalization when there is plenty of white space in the placement examples.

Table VI shows the impact of back-end contour. Without it, only eight of the examples can be legalized successfully without adjusting the macros, because some macros are pushed outside the chip. With back-end contour for budgeting the legal sites on each row, we can legalize all examples.

IV. CONCLUSION AND FUTURE WORK

A three-step mixed-size detailed placement, named XDP, is described in this paper. A combination of constraint graph and linear programming is used to remove the overlap between macros. Standard cells are legalized by an enhanced greedy method. Sliding-window-based cell swapping is applied to further reduce wirelength. The impact of individual techniques is analyzed and quantified. When applied to global placement results generated by APlace, XDP can produce wirelength comparable to the native detailed placement of APlace, and 3% shorter wirelength compared to Fengshui 5.0. When applied to global placements generated by mPL6, XDP is the only detailed placement that successfully produce a legal placement for all the examples, while APlace and Fengshui fail for $4/9$ and $1/3$ of the examples. For cases where legal placements can be compared, the wirelength produced by XDP is shorter by 3% on average. Furthermore, XDP displayed a higher robustness by covering a broader spectrum of examples by different global placement tools. Future work includes extention to placement instances where both movable macros and fixed macros are present. Consideration for routability and performance may also be included.

TABLE III
COMPARISON OF DETAILED PLACEMENT ALGORITHMS USING GLOBAL PLACEMENTS GENERATED BY MPL6.

circuit	GPWL	Aplace		Fengshui		XDP	
		FWL	RT(s)	FWL	RT(s)	FWL	RT(s)
ibm01	2.10E+06	N/A		2.23E+06	13	2.18E+06	37
ibm02	4.54E+06	5.12E+06	70	5.09E+06	22	4.74E+06	70
ibm03	6.94E+06	N/A		Overlap		6.64E+06	54
ibm04	7.37E+06	7.84E+06	98	N/A		7.53E+06	75
ibm05	9.36E+06	9.79E+06	80	9.80E+06	38	9.73E+06	66
ibm06	6.39E+06	N/A		N/A		5.97E+06	64
ibm07	1.00E+07	1.04E+07	150	Overlap		1.01E+07	120
ibm08	1.25E+07	1.25E+07	192	1.25E+07	68	1.19E+07	157
ibm09	1.37E+07	1.32E+07	233	1.38E+07	81	1.27E+07	146
ibm10	3.01E+07	N/A		Overlap	105	2.95E+07	321
ibm11	1.76E+07	N/A		1.91E+07	131	1.82E+07	206
ibm12	3.67E+07	N/A		N/A		3.44E+07	330
ibm13	2.26E+07	N/A		2.40E+07	145	2.36E+07	242
ibm14	3.62E+07	N/A		3.65E+07	540	3.53E+07	443
ibm15	5.57E+07	5.27E+07	1230	5.24E+07	420	5.00E+07	552
ibm16	5.73E+07	5.55E+07	1425	5.59E+07	447	5.30E+07	671
ibm17	6.67E+07	6.70E+07	1103	6.71E+07	493	6.53E+07	923
ibm18	4.41E+07	4.41E+07	1472	4.42E+07	550	4.31E+07	724
Avg.		1.03	1.52	1.03	0.60	1.00	1.00

TABLE IV IMPACT OF LP-BASED MACRO COORDINATE ASSIGNMENT.

circuit	GPWL	Greedy		LP	
		FWL	RT(s)	FWL	RT(s)
ibm01	2.10E+06	2.22E+06	30	2.18E+06	37
ibm02	4.54E+06	5.00E+06	87	4.77E+06	70
ibm03	6.94E+06	6.67E+06	58	6.68E+06	54
ibm04	7.37E+06	7.52E+06	66	7.59E+06	75
ibm05	9.36E+06	9.76E+06	66	9.76E+06	66
ibm06	6.39E+06	6.00E+06	66	6.06E+06	64
ibm07	1.00E+07	1.01E+07	123	1.02E+07	120
ibm08	1.25E+07	1.21E+07	152	1.19E+07	157
ibm09	1.37E+07	1.29E+07	145	1.28E+07	146
ibm10	3.01E+07	2.91E+07	340	2.90E+07	321
ibm11	1.76E+07	1.82E+07	195	1.80E+07	206
ibm12	3.67E+07	3.52E+07	380	3.48E+07	330
ibm13	2.26E+07	2.35E+07	242	2.34E+07	242
ibm14	3.62E+07	3.55E+07	452	3.54E+07	443
ibm15	5.57E+07	5.04E+07	650	5.03E+07	552
ibm16	5.73E+07	5.32E+07	665	5.31E+07	671
ibm17	6.67E+07	6.52E+07	948	6.52E+07	923
ibm18	4.41E+07	4.32E+07	715	4.31E+07	724
Avg.		1.01	1.02	1.00	1.00

ACKNOWLEDGEMENTS

This research is partially supported by Semiconductor Research Corporation under grant 2003-TJ-1091, supported by the National Science Foundation under grant CCF 0430077, and supported by Magma Design Automation under UC Micro Program 05-096. The authors would like to thank Prof. Andrew Kahng and Mr. Qinke Wang for providing APlace, Prof. Patrick Madden for providing Fengshui for use.

TABLE V IMPACT OF MOVABLE MACROS DURING LEGALIZATION.

circuit	GPWL	Fixed		Movable	
		FWL	RT(s)	FWL	RT(s)
ibm01	2.10E+06	2.25E+06	37	2.18E+06	37
ibm02	4.54E+06	4.83E+06	62	4.77E+06	70
ibm03	6.94E+06	6.93E+06	63	6.68E+06	54
ibm04	7.37E+06	7.97E+06	67	7.59E+06	75
ibm05	9.36E+06	9.75E+06	68	9.76E+06	66
ibm06	6.39E+06	6.21E+06	66	6.06E+06	64
ibm07	1.00E+07	1.09E+07	138	1.02E+07	120
ibm08	1.25E+07	1.18E+07	117	1.19E+07	157
ibm09	1.37E+07	1.31E+07	123	1.28E+07	146
ibm10	3.01E+07	3.10E+07	236	2.90E+07	321
ibm11	1.76E+07	1.90E+07	175	1.80E+07	206
ibm12	3.67E+07	3.90E+07	320	3.48E+07	330
ibm13	2.26E+07	2.53E+07	244	2.34E+07	242
ibm14	3.62E+07	3.62E+07	354	3.54E+07	443
ibm15	5.57E+07	5.13E+07	490	5.03E+07	552
ibm16	5.73E+07	5.34E+07	466	5.31E+07	671
ibm17	6.67E+07	6.65E+07	746	6.52E+07	923
ibm18	4.41E+07	4.45E+07	635	4.31E+07	724
Avg.		1.04	0.89	1.00	1.00

TABLE VI IMPACT OF BACKEND CONTOUR.

circuit	GPWL	w/o backend		w/ backend	
		FWL	RT(s)	FWL	RT(s)
ibm01	2.10E+06	N/A		2.18E+06	37
ibm02	4.54E+06	N/A		4.77E+06	70
ibm03	6.94E+06	N/A		6.68E+06	54
ibm04	7.37E+06	N/A		7.59E+06	75
ibm05	9.36E+06	9.76E+06	67	9.76E+06	66
ibm06	6.39E+06	6.06E+06	65	6.06E+06	64
ibm07	1.00E+07	N/A		1.02E+07	120
ibm08	1.25E+07	N/A		1.19E+07	157
ibm09	1.37E+07	N/A		1.28E+07	146
ibm10	3.01E+07	N/A		2.90E+07	321
ibm11	1.76E+07	1.80E+07	205	1.80E+07	206
ibm12	3.67E+07	N/A		3.48E+07	330
ibm13	2.26E+07	N/A		2.34E+07	242
ibm14	3.62E+07	3.54E+07	445	3.54E+07	443
ibm15	5.57E+07	5.03E+07	565	5.03E+07	552
ibm16	5.73E+07	5.31E+07	674	5.31E+07	671
ibm17	6.67E+07	6.52E+07	943	6.52E+07	923
ibm18	4.41E+07	4.31E+07	729	4.31E+07	724

REFERENCES

[1] A. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," in *Proc. Intl Symposium on Physical Design*, pp. 18–25, 2004.

[2] A. Agnihotri, S. Ono, and P. H. Madden, "Recursive bisection placement: Feng shui 5.0 implementation details," in *Proc. Intl Symposium on Physical Design*, pp. 230–232, April 2005.

[3] T. Chan, J. Cong, M. Romesis, J. Shinnerl, K. Sze, and M. Xie, "mpl6: A robust multilevel mixed-size placement engine.," in *Proc. Intl Symposium on Physical Design*, pp. 227–229, April 2005.

[4] S. N. Adya and I. L. Markov., "Consistent placement of macro-blocks using floorplanning and standard-cell placement," in *Proc. Intl Symposium on Physical Design*, pp. 12–17, April 2002.

[5] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov, "Unification of partitioning, placement and floorplanning," in *Proc. Int. Conf. on Computer Aided Design*, pp. 550–557, November 2004.

[6] C.-C. Chang, J. Cong, and X. Yuan, "Multilevel placement for large-scale mixed-size ic designs," in *Proc. Asia South Pacific Design Automation Conf.*, pp. 325–330, January 2003.

[7] T. Taghavi, X. Yang, and B.-K. Choi, "Dragon 2005: Large-scale mized-size placement tool," in *Proc. Intl Symposium on Physical Design*, April 2005.

[8] J. Cong, M. Romesis, and J. Shinnerl, "Robust mixed-size placement under tight white-space constraints.," in *Proc. Int. Conf. on Computer Aided Design*, pp. 165–173, November 2005.

[9] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. 35th ACM/IEEE Design Automation Conference*, pp. 269–274, 1998.

[10] J. Vygen, "Algorithms for large-scale flat placement," in *Proc. 34th ACM/IEEE Design Automation Conference*, pp. 746–751, 1997.

[11] J. Vygen, "Algorithms for detailed placement of standard cells," in *Proc. Conf. Design, Automation and Test in Europe*, pp. 321–324, 1998.

[12] U. Brenner, A. Pauli, and J. Vygen, "Almost optimum placement legalization by minimum cost flow and dynamic programming," in *Proc. Intl Symposium on Physical Design*, pp. 2–9, April 2004.

[13] K. P. Vorwerk, A. Kennings, and A. Vannelli, "Engineering details of a stable force-directed placer," in *Proc. Int. Conf. on Computer Aided Design*, pp. 573–580, November 2004.

[14] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proc. Intl Symposium on Physical Design*, April 2005.

[15] B. Yao, H. Chen, C.-K. Cheng, N.-C. Chou, L.-T. Liu, and P. Suaris, "Unified quadratic programming approach for mixed mode placement," in *Proc. Int'l Symposium on Physical Design*, April 2005.

[16] A. Khatkhate, C. Li, A. R. Agnihotri, M. C. Yildiz, S. Ono, C.-K. Koh, and P. H. Madden, "Recursive bisection based mixed block placement," in *Proc. Intl Symposium on Physical Design*, pp. 84–89, April 2004.

[17] K. Doll, F. Johannes, and K. Antreich, "Iterative placement improvement by network flow methods," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 10, October 1994.

[18] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 472–479, November 1995.

[19] G. D. Micheli, "Performance-oriented synthesis of large-scale domino cmos circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, pp. 751–765, 1987.

[20] S. K, A. Wang, R. Brayton, and A. Sangiovanni-Vincentelli, "Timing optimization of combinatorial logic," in *Proc. Int. Conf. on Computer Aided Design*, pp. 282–285, November 1988.

[21] K. J. Singh, *Performance Optimization for Digital Circuits*. PhD thesis, Department of Computer Science, University of California Berkeley, 1992.

[22] S. Xu, *Synthesis for Hign-Density and High-Performance FPGA*. PhD thesis, Computer Science Department, University of California, Los Angeles, 2000.

[23] S. Nag and K. Chaudhary, "Post-placement residual-overlap removal with minimal movement," in *Proc. Conf. Design, Automation and Test in Europe*, pp. 581–586, 1999.

[24] R.Okuda, T. Sato, H. Onodera, and K. Tamaru, "An efficient algorithm for layout compaction problem with symmetry constraints," in *Proc. Int. Conf. on Computer Aided Design*, pp. 148–153, November 1989.

[25] S.-W. Hur and J. Lillis, "Mongrel: Hybrid techniques for standard-cell placement," in *Proc. IEEE International Conference on Computer Aided Design*, (San Jose, CA), pp. 165–170, Nov 2000.

[26] X. Tang, R. Tian, and M. D. F. Wong, "Optimal redistribution of white space for wire length minimization," in *Proc. Asia South Pacific Design Automation Conf.*, pp. 412–417, January 2005.

[27] M. Csaba, "Fast cholesky factorization for interior point methods of linear programming," *Computers and Mathematics with Applications*, vol. 31, pp. 49–51, 1996.

[28] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design," *US Patent No. 6,370,673*, 2002.

[29] A. Kahng, S. Reda, and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," in *Proc. Int. Conf. on Computer Aided Design*, pp. 891–899, November 2005.