FastPlace 2.0: An Efficient Analytical Placer for Mixed-Mode Designs *

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 2.0* – an extension to the efficient analytical standard-cell placer - *FastPlace* [15], to address the mixed-mode placement problem. The main contributions of our work are: (1) Extensions to the global placement framework of *FastPlace* to handle mixed-mode designs. (2) An efficient and optimal minimum perturbation macro legalization algorithm that is applied after global placement to resolve overlaps among the macros. (3) An efficient legalization scheme to legalize the standard cells among the placeable segments created after fixing the movable macros. On the ISPD 02 Mixed-Size placement benchmarks [3], our algorithm is $16.8X$ and $7.8X$ faster than state-of-the-art academic placers *Capo 9.1* and *Fengshui 5.0* respectively. Correspondingly, we are on average, 12% and 3% better in terms of wirelength over the respective placers.

I. Introduction

The explosive growth in the size of integrated circuits has imposed enormous challenges on placement algorithms. Placement tools have to produce good-quality results satisfying various design objectives, such as timing, congestion etc. Simultaneously, they have to be computationally efficient to deliver these solutions in a reasonable amount of runtime.

As the time to market for designs is constantly shrinking, there has been a steady increase in the re-use of pre-designed or generated macro blocks like IP cores, embedded memories, analog blocks etc. Designs today often contain a combination of a large number of macro blocks and millions of standard cells. This design style, known as mixed-mode design or mixed-size design complicates the placement step and imposes a lot of difficulty on placement tools due to the varied sizes of the placeable components.

Traditionally, the mixed-mode placement problem was divided into two stages namely, floorplanning or block/module placement and cell placement. Large macro blocks were handled during the floorplanning stage followed by cell placement wherein the macro blocks were treated as fixed. Current designs can have thousands of large and medium sized macros along with millions of standard cells. As a result, traditional floorplanning techniques cannot scale to this problem both in terms of runtime as well as solution quality. With an ever-increasing trend toward mixed-mode design, it is necessary to have efficient techniques that can simultaneously handle this combination of placeable objects.

*This work was supported by the Semiconductor Research Corporation under Task ID: 1206.

Over the last few years, the mixed-mode placement problem has generated a lot of interest. Placement algorithms handling this problem employ various approaches including partitioning [2, 4, 11], clustering and simulated annealing [7] and analytical placement [6, 8, 10, 16–19]. Analytical placement techniques based on the force-directed method are promising for handling the mixed-mode placement problem. This is because force-directed methods can seamlessly handle the varied sizes of placeable objects without employing additional techniques like partitioning or clustering [8,18]. Secondly, they can be very efficient and scalable to handle large-scale placement problems [15].

In this paper we present *FastPlace 2.0*, an efficient analytical placer for mixed-mode designs. The main contributions of our work are:

- Extensions to the Cell Shifting technique of *FastPlace* [15] to handle mixed-mode designs.
- An efficient macro legalization algorithm that perturbs the macros by the minimum possible distance to resolve overlaps created during global placement. The macro legalization problem is solved by a floorplanning approach that uses the sequence pair to represent the relative positions of the macros. We prove that for a given sequence pair our algorithm is optimal. We then use simulated annealing to generate a good sequence pair and a non-overlapping placement of the macros with minimum perturbation from their global placement positions.
- An efficient legalization scheme that legalizes the standard cells among the placeable segments created after fixing the movable macros.

The rest of this paper is organized as follows: Section II gives an overview of *FastPlace* for standard-cell placement. Section III outlines the mixed-mode placement flow and describes the extensions to the global placement framework for handling mixed-mode designs. Section IV describes the legalization scheme for macros and standard cells. Section V describes the detailed placement technique. Experimental results are provided in Section VI followed by the conclusions in Section VII.

II. FastPlace: Standard-Cell Placement

In this section, we give an overview of the *FastPlace* analytical standard-cell placer described in [15]. *FastPlace* utilizes a quadratic wirelength objective function and is based on three

0-7803-9451-8/06/$20.00 ©2006 IEEE.

key features: Cell Shifting, Iterative Local Refinement and a Hybrid Net Model.

The Cell Shifting technique is used to remove cell overlap and spread the cells over the core region. This technique roughly maintains the relative order of the cells as obtained by solving the quadratic program. During Cell Shifting the core region is binned and the utilization of each bin is computed. The cells are then spread depending on the utilization of their respective bins. The basic intuition behind Cell Shifting is to even out the utilization of adjacent bins. This is done by constructing an unequal bin structure from the regular bin structure. Cells are then mapped from the regular bin structure to the unequal bin structure. After each iteration of Cell Shifting, additional forces are added to the cells by way of pseudo nets connected to pseudo pins on the placement boundary. This prevents the cells from collapsing back to their previous positions during the next quadratic programming step.

The Iterative Local Refinement technique is used to reduce the wirelength of the placement based on the half-perimeter wirelength measure. This technique uses a greedy heuristic to move the cells based on a weighted score of the linear wirelength and the placement utilization.

FastPlace uses the pre-conditioned conjugate gradient method to minimize the quadratic objective function. The runtime of the solver is directly proportional to the number of non-zero entries in the connectivity matrix. To improve the speed of the solver, the algorithm uses a Hybrid Net Model to transform the circuit netlist for quadratic placement. [15] showed that the model results in a $2.95X$ reduction in the number of non-zero entries in the connectivity matrix and a $1.5X$ speed-up in the solver as compared to the clique model on the ISPD04 IBM Standard Cell Benchmarks [14].

For standard-cell placement, *FastPlace* achieves comparable placement solutions to other state-of-the-art academic placers, but in a significantly lesser runtime. We now build on this ultra-fast placement tool to handle mixed-mode designs.

III. Mixed-Mode Placement

Our mixed-mode placement flow is summarized in Figure 1. For the global placement stage, we employ the same top-level flow as [15]. During legalization, we first remove the overlaps among the macros and assign them to legal positions in the core region. Once legalized, the macro positions are fixed and they behave as placement blockages for all subsequent steps. These placement blockages fragment the rows in the core region into placeable segments. In the next step of legalization we move the standard cells among the placeable segments to satisfy their respective capacities. Finally, we legalize the standard cells within the segments. Following legalization we perform detailed placement on the standard cells to further reduce the wirelength of the placement.

Algorithm *Mixed-Mode Placement*
Stage 1: Global Placement
Step 1: Coarse Global Placement
Repeat
1. Solve the quadratic program
2. Perform Cell Shifting on standard cells and macro blocks and Add Spreading Forces

Until the placement is roughly even
Step 2: Wirelength Improved Global Placement
Repeat
1. Solve the quadratic program
2. Perform Iterative Local Refinement on standard cells and macro blocks
3. Perform Cell Shifting on standard cells and macro blocks and Add Spreading Forces

Until the placement is very even
Stage 2: Legalization
1. Legalize Macro Blocks
2. Fix Macros and move standard cells among placeable segments to satisfy segment capacity
3. Legalize standard cells within segments

Stage 3: Detailed Placement

Fig. 1. The *Mixed-Mode* placement flow.

A. Cell Shifting for Mixed-Mode Placement

As described in Section II, during Cell Shifting, the cells are spread over the core region by attempting to even out the utilization of adjacent bins in the regular bin structure. For standard-cell placement, the width of the bins in the regular bin structure is greater than the average cell width. Hence, the movement of any cell has an influence on the utilization of only the adjacent bins. On the other hand, for mixed-mode placement, the movement of a macro will influence the utilization of all the bins spanned by the macro. Therefore, to move a macro during Cell Shifting we need to consider a larger region that is proportional to the size of the macro.

Shifting of the macros follows the same two-step process as the standard cells. We first construct an unequal bin structure from the regular bin structure. The macros are then linearly mapped from the regular bin structure to the unequal bin structure. The only difference between Cell Shifting for the macros and the cells is the construction of the unequal bin structure. Since Cell Shifting is independent and similiar in the vertical and horizontal directions, we describe the technique for the horizontal direction. Figure 2 illustrates the construction of the unequal bin structure for horizontal shifting. From Figure 2(a), for the regular bin structure, let,

- N: Total number of bins spanned by the macro.
- x_span: Total number of columns spanned by the macro.
- OB_L: x-coordinate of the left boundary of the leftmost bins spanned by the macro.
- OB_R: x-coordinate of the right boundary of the rightmost bins spanned by the macro.
- U_C: Sum of the utilizations of the N bins spanned by the macro (shaded region with lines to the right bottom).
- U_L: Sum of the utilizations of N bins to the left of the macro. (shaded region with lines to the left bottom).
- U_R: Sum of the utilizations of N bins to the right of macro. (shaded region with lines to the left bottom).

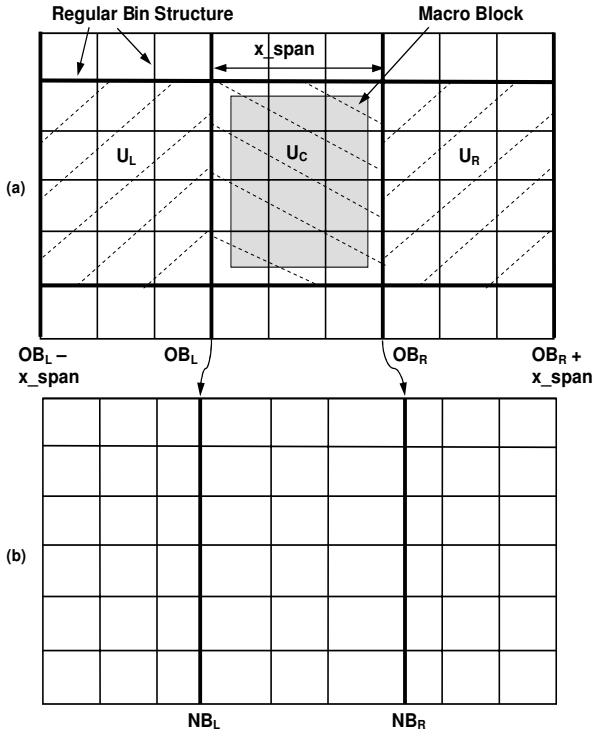

Fig. 2. (a) Regular bin structure (b) Unequal bin structure for macro block cell shifting.

From Figure 2(b), for the unequal bin structure, let,

- NB_L: x-coordinate of the left boundary of the leftmost bins spanned by the macro.
- NB_R: x-coordinate of the right boundary of the rightmost bins spanned by the macro.

Then,

$$NB_L = \frac{(OB_L - x\_span)(U_C + \delta) + OB_R(U_L + \delta)}{U_L + U_C + 2\delta}$$

and,

$$NB_R = \frac{OB_L(U_R + \delta) + (OB_R + x\_span)(U_C + \delta)}{U_R + U_C + 2\delta}$$

As in [15], the parameter δ is set to a value of 1.5 to prevent cross-over of bin boundaries in the unequal bin structure. For performing the linear mapping, if,

- x: x-coordinate of the macro before mapping.
- $x^{'}$: x-coordinate of the macro after mapping.

Then,

$$x^{'} = \frac{NB_R(x - OB_L) + NB_L(OB_R - x)}{OB_R - OB_L}$$

Once the macro is moved, we add the spreading force to the macro and update the connectivity matrix for the next quadratic programming step in the same fashion as [15].

IV. Legalization

A key issue with analytical placement is that it generates overlaps among the cells or macros that need to be resolved. We divide our legalization stage into two steps. First, we ignore all the standard cells and resolve overlaps among the macros and assign them to legal positions. In the next step, we fix the macros and legalize the standard cells. These steps are described in more detail below.

A. Macro Block Legalization

During legalization, we want to maintain the macro positions in the global placement solution as much as possible. If we denote the original position of a macro, determined by global placement, as its *target position*, then, the macro block legalization problem is to minimize the total perturbation of all the macros from their target positions such that there are no overlaps among them.

This problem is solved by using a fixed-outline floorplanning approach. We use the sequence pair [12] to represent the floorplan and enforce the non-overlapping constraints among the macros. We can also easily incorporate other floorplanning representations in our approach. We formally describe the problem of finding a minimum perturbation placement for a given sequence pair below.

Minimum Perturbation Floorplan Realization (MPFR) Problem:

Given: n macros with target coordinates (x_i^*, y_i^*) for $i = 1, \ldots, n$ and a sequence pair (p, q).

Determine: Legalized coordinates (x_i, y_i) s.t. $\sum_{i=1}^{n} |x_i - x_i^*| + |y_i - y_i^*|$ is minimized.

In the following sub-sections we first describe the *Iterative Clustering Algorithm* that is used to solve the *MPFR* problem. We then describe the top-level flow for macro legalization using simulated annealing. Since the horizontal and vertical non-overlapping constraints can be handled independently, we only discuss the horizontal problem.

A.1 Iterative Clustering Algorithm

The basic idea of the *Iterative Clustering Algorithm* is that if we know which macros abut with each other to form a cluster in the optimal solution, then the position of the cluster is easy to find. To determine which macros should be grouped in the same cluster, we always shift all clusters to their optimal positions. In doing so if there are any overlaps among some clusters, then we know that these clusters should be merged to form larger clusters. In Figure 3 we give the pseudo-code of the Iterative Clustering Algorithm.

From Figure 3, in step 1, immediate neighbours of macros are those that can potentially abut. They are associated with the non-transitive edges in the constraint graph. The immediate neighbours of all macros can be found in $O(n^2)$ time. In steps 3-4, the macros are placed one at a time from left to right (i.e., according to the sequence p). Then the clustering is updated according to steps 5-11. The condition in step 5 and the closest cluster in step 6 can be determined by considering the constraints of the immediate left neighbours of modules in C. The shifting in step 8 is easy according to the following lemma.

Iterative Clustering Algorithm:

1. Find the immediate left and right neighbours of all macros
2. **for** $i = 1$ **to** n
3. Place macro p_i in its target position
4. Let C be a new cluster consisting of p_i
5. **while** C overlaps with other clusters **do**
6. Merge C with the closest cluster on its left
7. Let C be the new cluster formed
8. Shift C to its optimal position
9. **if** macro m in C is at its target position **do**
10. Detach m from C if necessary and goto step 8
11. **endwhile**
12. **endfor**

Fig. 3. Iterative Clustering Algorithm.

Lemma 1 *For a cluster C, its position is optimal if the number of macros perturbed to the left from their target positions is equal to the number perturbed to the right.*

Note that since we add macros from left to right, macros will always be added to the right of a stationary cluster. So the clusters will always shift left. Therefore, it is very easy to find the correct shift amount of the newly formed clusters. In step 9, after shifting a cluster C, a macro $m \in C$ may potentially reach its target position. If m does not have any other macros in the same cluster to its right then it should be detached from the cluster. If not, m will move with the cluster during subsequent steps and hence its position will not be optimal in the final solution. The condition to detach m can be checked by looking at its immediate right neighbours.

Although the while loop in steps 5-11 looks complicated, we can show with careful implementation and analysis that the runtime complexity of the Iterative Clustering Algorithm is $O(n^2)$. We show in Section VI that its runtime is insignificant in practice.

A.2 Macro Legalization by Simulated Annealing

The aim of the macro legalization algorithm is to obtain a sequence pair such that the corresponding placement obtained from the *Iterative Clustering Algorithm* will resolve overlaps among the macros with minimum perturbation from the global placement solution. Another factor to be considered during placement is that the macros have to be placed in legal positions within the core region. Hence, the cost function is defined as a weighted sum of the total perturbation along with a penalty for being out of bound. We use simulated annealing to search for a sequence pair with low cost.

If (p, q) represents the sequence pair. Then, the initial sequence for p/q is generated by sorting the macros in ascending order according to the Manhattan distance from the upper left / lower left corner to their target positions. This sequence pair closely corresponds to the original placement and is usually quite good. Hence, a low-temperature annealing is sufficient to generate a good result. Besides, we restrict each annealing move to randomly exchange two adjacent macros in one of the two sequences so as to not disturb the current solution significantly.

In Figures 4 and 5 we plot the placement of the macros before and after legalization for the circuit ibm01. From the two figures, we can see that the macros have moved by a very small amount from the global placement solution.

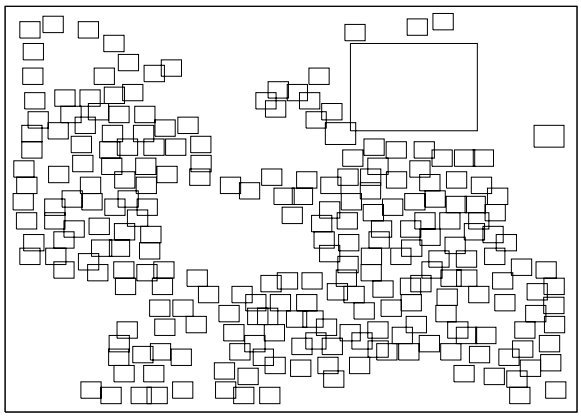

Fig. 4. Circuit ibm01 before legalization of movable macros.

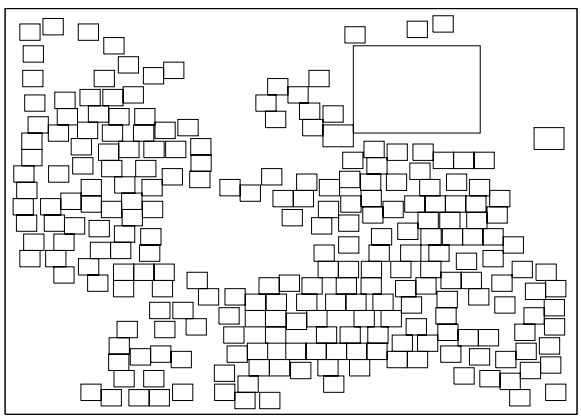

Fig. 5. Circuit ibm01 after legalization of movable macros.

B. Standard Cell Legalization

Once the overlaps among the movable macros have been resolved, we fix their positions for all subsequent steps and treat them as placement blockages. We then divide each row in the core region into placeable segments based on the overlap of the blockages with the row. A placeable segment is defined as the maximal part of a row that is not covered by a placement blockage. We then move the standard cells among the placeable segments to satisfy their respective capacities. Finally, we legalize the standard cells within the segments.

To move the cells among the placeable segments, we use a greedy heuristic similar to the Iterative Local Refinement technique of [15]. For every cell present in a segment, we compute 8 *scores* based on moving the cell to its nearest 8 neighboring segments. For calculating the score, we assume that a cell is moving from its current position in a *source* segment to the nearest possible position in the *target* segment. Each score is

TABLE I
PLACEMENT BENCHMARK STATISTICS.

Circuit	#Cells	#Macros	#Pads	#Nets	%Cell Area	%Macro Area
ibm01	12260	246	246	14111	37.23	42.76
ibm02	19071	271	259	19584	24.69	55.31
ibm03	22563	290	283	27401	30.04	49.96
ibm04	26925	295	287	31970	38.03	41.98
ibm05	28146	0	1201	28446	80.01	0.00
ibm06	32154	178	166	34826	34.60	45.41
ibm07	45348	291	287	48117	44.07	35.93
ibm08	50722	301	286	50513	38.79	41.20
ibm09	52857	253	285	60902	40.18	39.82
ibm10	67899	786	744	75196	20.34	59.66
ibm11	69779	373	406	81454	42.36	37.63
ibm12	69788	651	637	77240	28.35	51.65
ibm13	83285	424	490	99666	43.82	36.18
ibm14	146474	614	517	152772	60.36	19.64
ibm15	160794	393	383	186608	53.26	26.74
ibm16	182522	458	504	190048	42.11	37.89
ibm17	183992	760	743	189581	62.80	17.20
ibm18	210056	285	272	201920	71.31	8.69

TABLE II
BREAK-UP OF TOTAL RUNTIME (ALL VALUES IN SECONDS)

Ckt	Stage 1	Stage 2		Stage 3	Total Time
	Global Placement	Legalize Macros	Legalize Cells	Detailed Placement	
ibm01	6.73	0.85	0.93	1.97	10.48
ibm02	21.29	1.01	2.34	9.22	33.86
ibm03	17.96	1.00	2.23	6.16	27.35
ibm04	28.16	1.07	2.60	6.50	38.33
ibm05	17.54	0.00	2.34	14.15	34.03
ibm06	24.86	0.39	3.34	8.88	37.47
ibm07	84.49	1.21	5.37	14.10	105.17
ibm08	79.20	1.10	7.62	33.30	121.22
ibm09	66.80	0.68	10.21	15.91	93.60
ibm10	104.10	5.52	12.47	39.66	161.75
ibm11	96.55	1.96	10.07	23.32	131.90
ibm12	116.91	3.73	11.17	57.48	189.29
ibm13	116.87	2.30	19.22	31.49	169.88
ibm14	220.86	5.40	23.21	66.45	315.92
ibm15	305.02	2.03	31.08	76.83	414.96
ibm16	265.54	2.74	31.70	117.25	417.23
ibm17	425.82	7.94	46.30	143.81	623.87
ibm18	526.18	1.18	37.00	204.83	769.19

a weighted sum of two 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 segments. 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 component. If all the scores are negative, the cell will remain in the current segment. Otherwise, it will move to the target segment with the highest score for the move. During one iteration, we traverse through all the segments in the core region and follow the above steps for cell movement. Subsequently, this iteration is repeated until all the segments are within their respective capacities. We then assign the cells to legal positions within each segment.

V. DETAILED PLACEMENT

The aim of the detailed placement stage is to further reduced the wirelength of the placement. We adopt the *FastDP* detailed placement algorithm described in [13] for the same. The detailed placement algorithm is based on four key techniques: global swap, vertical swap, local re-ordering and single-segment clustering. All the techniques act only on the standard cells and do not modify the positions of the macro blocks.

Briefly, the global swap uses the median idea of [9] to swap the standard cells for wirelength reduction. This technique operates on the entire core region. The vertical swap is similar to the global swap but it only considers cells in adjacent rows for swapping. The local re-ordering technique picks a subset of cells within a segment and tries out all possible left-right orderings of the cells to pick the one giving the best possible wirelength. Finally, retaining the order determined by local re-ordering, an optimal single-segment clustering algorithm is used to cluster the cells within a segment for further wirelength reduction.

VI. EXPERIMENTAL RESULTS

Our algorithm was tested on the ISPD02 IBM-MS Mixed-size Placement Benchmarks [3–5]. These designs are relatively large and contain many macro blocks and standard cells. All macro blocks are assumed to be hard blocks with fixed aspect ratios. Each design contains around 20% of whitespace. The circuit characteristics listed in Table I include the number of cells, macros, pads, nets and the area occupied by the cells and macro blocks as a percentage of the total placement area.

Table II gives the break-up of the runtime of *FastPlace 2.0* for the 18 benchmark circuits. From Column 3, it can be seen that on average, the macro block legalization algorithm takes only 1.9% of the total runtime over the 18 benchmark circuits. This demonstrates that the runtime of the algorithm is negligible compared to the other parts of the flow and it is highly efficient in resolving the overlaps among the macros.

In Table III, we compare *FastPlace 2.0* with various academic placers. Results for the *capo-parquet-capo* flow [4], *mPG-MS* [7], *Fengshui 2.4* [11], and *BonnPlace* [6] are as reported in the respective publications. We do not report runtimes for these placers as they were run on different machines. For runtime comparison we run *Capo 9.1* and *Fengshui 5.0*, which are updated versions of the tools published in [2] and [11] respectively. Both placers are run in their default mode. All experiments are run on an Intel Xeon, 3.06GHz CPU.

From Table III, we are on average, 12% and 3% better in terms of wirelength over *Capo 9.1* and *Fengshui 5.0* respectively. Correspondingly, we are $16.8X$ and $7.8X$ faster. We are on average 2% more in terms of wirelength as compared to $BonnPlace$. But accounting for the differences in the processors based on data obtained from [1], we are approximately $20X$ faster.

TABLE III
COMPARISON OF OUR PLACEMENT RESULTS WITH VARIOUS ACADEMIC PLACERS.
CAPO-I, MPG-MS, FENGSHUI 2.4 (FS 2.4), CAPO 9.1, FENGSHUI 5.0 (FS 5.0) AND BONNPLACE (BP)

Ckt	Half Perimeter Wirelength							RunTime				
	Our	$\frac{Capo-I}{Our}$ [4]	$\frac{mPG-MS}{Our}$ [7]	$\frac{FS2.4}{Our}$ [11]	$\frac{Capo9.1}{Our}$ [2]	$\frac{FS5.0}{Our}$	$\frac{BP}{Our}$ [6]	Our (sec)	Capo 9.1 (sec)	$\frac{Capo9.1}{Our}$	FS 5.0 (sec)	$\frac{FS5.0}{Our}$
ibm01	2.45	1.62	1.23	0.98	1.05	1.01	0.92	10	219	20.90	142	13.55
ibm02	4.91	1.70	1.51	1.09	1.06	1.08	1.00	34	457	13.50	245	7.24
ibm03	7.32	1.66	1.53	1.03	1.20	1.16	0.96	27	735	26.87	284	10.38
ibm04	8.14	1.66	1.29	0.98	1.11	1.05	1.01	38	771	20.11	323	8.43
ibm05	10.24	1.12	1.06	0.99	1.00	0.96	0.98	34	684	20.10	372	10.93
ibm06	6.01	1.71	1.53	1.13	1.25	1.14	1.09	37	809	21.59	437	11.66
ibm07	10.99	1.43	1.25	1.07	1.11	1.05	0.95	105	1236	11.75	586	5.57
ibm08	12.38	1.71	1.32	1.10	1.13	1.04	1.02	121	1322	10.91	647	5.34
ibm09	13.79	1.42	1.35	1.00	1.11	1.00	0.96	94	1375	14.69	660	7.05
ibm10	31.65	1.92	1.38	1.18	1.18	1.11	1.04	162	2666	16.48	1085	6.71
ibm11	20.30	1.40	1.31	0.98	1.08	0.97	0.94	132	2172	16.47	891	6.76
ibm12	34.18	1.51	1.30	1.04	1.17	1.06	0.93	189	3413	18.03	1011	5.34
ibm13	25.21	1.56	1.50	0.99	1.16	0.98	0.96	170	4288	25.24	1189	7.00
ibm14	37.76	1.49	1.15	1.02	1.07	1.03	1.00	316	5091	16.11	2553	8.08
ibm15	52.56	1.34	1.25	0.99	1.13	0.97	0.94	415	6399	15.42	3171	7.64
ibm16	58.37	N/A	1.24	1.05	1.21	1.03	0.99	417	7211	17.28	3626	8.69
ibm17	69.89	1.32	1.12	1.01	1.08	0.99	0.95	624	6782	10.87	3935	6.31
ibm18	45.39	1.21	1.12	0.99	1.05	0.98	1.01	769	5163	6.71	3471	4.51
Average		**1.52**	**1.30**	**1.03**	**1.12**	**1.03**	**0.98**			**16.84**		**7.84**

VII. CONCLUSION AND FUTURE WORK

In this paper we extend the efficient analytical placement tool *FastPlace* to handle mixed-mode designs. The current implementation handles the wirelength minimization problem. It produces better results than state-of-the-art academic placers in a significantly lesser runtime.

Routability and timing are key concerns for industrial designs. Future extensions to our work would be in considering the problem of timing driven placement and routability driven placement. Also, the current implementation does not handle rotation and mirroring for the macro blocks. We will be working on handling these constraints for mixed mode placement in the future.

REFERENCES

[1] Standard performance evaluation corporation. `http://www.spec.org/`.

[2] S. N. Adya, S. Chaturvedi, J. A. Roy, D. Papa, and I. L. Markov. Unification of partitioning, floorplanning and placement. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pages 550–557, 2004.

[3] S. N. Adya and I. L. Markov. ISPD02 IBM-MS Mixed-size Placement Benchmarks. `http://vlsicad.eecs.umich.edu/BK/ISPD02bench/`.

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

[5] S. N. Adya and I. L. Markov. Combinatorial techniques for mixed-size placement. *ACM Trans. Design Automation of Electronics Systems*, 10(1):58–90, January 2005.

[6] U. Brenner and M. Struzyna. Faster and better global placement by a new transportation algorithm. In *Proc. ACM/IEEE Design Automation Conf.*, pages 591–596, 2005.

[7] C. C. Chang, J. Cong, and X. Yuan. Multi-level placement for large-scale mixed-size IC designs. In *Proc. Asia and South Pacific Design Automation Conf.*, pages 325–330, 2003.

[8] H. Eisenmann and F. Johannes. Generic global placement and floorplanning. In *Proc. ACM/IEEE Design Automation Conf.*, pages 269–274, 1998.

[9] S. Goto. An efficient algorithm for the two-dimensional placement problem in electrical circuit layout. *IEEE Trans. Circuits and Systems*, CAS-28(1):12–18, 1981.

[10] A. B. Kahng and Q. Wang. An analytical placer for mixed-size placement and timing-driven placement. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pages 565–572, 2004.

[11] 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. Symp. on Physical Design*, pages 84–89, 2004.

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

[13] M. Pan, N. Viswanathan, and C. Chu. An efficient and effective detailed placement algorithm. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pages 48–55, 2005.

[14] N. Viswanathan and C. C.-N. Chu. ISPD04 IBM Standard Cell Benchmarks with Pads. `http://www.public.iastate.edu/~nataraj/ISPD04_Bench.html`.

[15] N. Viswanathan and C. C.-N. Chu. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In *Proc. Intl. Symp. on Physical Design*, pages 26–33, 2004.

[16] K. Vorwerk and A. Kennings. An improved multi-level framework for force-directed placement. In *Proc. Conf. on Design Automation and Test in Europe*, pages 902–907, 2005.

[17] K. Vorwerk, A. Kennings, and A. Vannelli. Engineering details of a stable force-directed placer. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pages 573–580, 2004.

[18] 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. Intl. Symp. on Physical Design*, pages 193–199, 2005.

[19] 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.*, pages 271–276, 2000.