An Efficient Analytical Placement Algorithm for Mixed-Mode Designs in the presence of Placement Blockages

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
Presenter: Natarajan Viswanathan

***Abstract*— This paper presents *FastPlace 2.0*, an efficient analytical placement algorithm to handle the mixed-mode placement problem. The main contributions of this paper are: (1) Extensions to the global placement framework of the standard-cell placer - *FastPlace* [1] to handle mixed-mode designs. (2) An efficient and optimal minimum perturbation legalization algorithm, applied after global placement, to resolve overlaps among the macros. (3) An efficient legalization scheme to legalize the standard cells among the segments created after fixing the movable macros. On the ISPD 02 Mixed-Size placement benchmarks [2], the algorithm is $16.8X$ and $7.8X$ faster than academic placers *Capo 9.1* and *Fengshui 5.0* respectively. Correspondingly, it results in 11% and 2% better wirelength over the respective placers.**

I. Introduction

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. 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.

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 [3]–[5], clustering and simulated annealing [6] and analytical placement [7]–[13]. 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 [7], [12]. Also, they can be very efficient and scalable to handle large-scale problems [1].

In this paper we extend the efficient analytical placer - *FastPlace* [1] to handle mixed-mode designs. The main contributions of our work are:

- Extensions to the Cell Shifting technique of *FastPlace* to handle mixed-mode designs.
- An efficient and optimal minimum perturbation legalization algorithm to resolve overlaps among the macros created during global placement. This problem is solved by using a floorplanning approach. Based on the global placement solution, we use a sequence pair to represent the relative positions of the macros. We subsequently prove that for a given sequence pair our algorithm is optimal and will result in a non-overlapping placement of the macros with minimum movement 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 outlines the mixed-mode placement flow and describes the extensions to the global placement framework of [1] to handle mixed-mode designs. Section III describes the legalization scheme for macros and standard cells. Section IV describes the detailed placement technique. Finally, experimental results are provided in Section V.

II. 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 [1]. 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 obstacles for all subsequent steps. These placement obstacles 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.

A. Cell Shifting for Mixed-Mode Placement

As described in [1], during Cell Shifting, the core region is binned and the utilization of each bin is computed. The cells are then spread out based on their respective bins and its current utilization. This is done by attempting to even out the utilization of adjacent bins. For standard-cells, the width of a bin 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, the movement of a macro will influence the utilization of all the bins spanned by it. Therefore, a larger region proportional

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 fbw.

to the area of the macro needs to be considered for shifting 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. 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 bin spanned by the macro.
- OB_R: x-coordinate of the right boundary of the rightmost bin spanned by the macro.
- U_C: Sum of the utilizations of all the bins spanned by the macro (hatched lines to the bottom right).
- U_L: Sum of the utilizations of N bins to the left of the macro. (hatched lines to the bottom left)
- U_R: Sum of the utilizations of N bins to the right of macro. (hatched lines to the bottom left)

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

- NB_L: x-coordinate of the left boundary of the leftmost bin spanned by the macro.
- NB_R: x-coordinate of the right boundary of the rightmost bin 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 [1], 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.

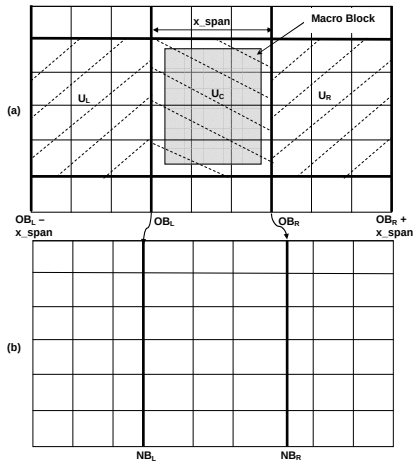

Fig. 2. (a) Regular bin structure (b) Unequal bin structure for macro block cell shifting.

- $x\prime$: x-coordinate of the macro after mapping.

Then,

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

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 [1].

III. Legalization

A key issue with analytical placement techniques is that they have 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 in the following sub-sections.

A. Macro Block Legalization

We want to maintain the macro positions in the global placement solution as much as possible during legalization. The original position of a macro, determined by global placement, is called its *target position*. The macro block legalization problem is to minimize the total perturbation of all 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 (SP) [14] to represent the floorplan and enforce the non-overlapping constraints among the macros. The aim of the floorplanning algorithm is to obtain a SP such that the corresponding placement will cause minimum perturbation of the macros and place them in legal positions within the core region. Hence, the cost function is defined as a weighted sum of the total perturbation and a penalty for being out of bound. We use simulated annealing to search for a SP with low cost.

If (p, q) represents the SP. 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 SP 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.

We formally describe the problem of finding a minimum perturbation placement for a given SP below. Since the horizontal and vertical non-overlapping constraints can be handled independently and the horizontal and vertical problems are similar, we only discuss the horizontal problem.

Minimum Perturbation Floorplan Realization (MPFR) Problem:

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

Determine: Legalized coordinates x_i s.t. $\sum |x_i - x_i^*|$ is minimized.

We now present an $O(n^2)$ optimal algorithm called *Iterative Clustering Algorithm* to solve the Minimum Perturbation Floorplan Realization Problem. The basic idea of the algorithm is that if we know which macros should be abutted 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. The pseudocode of the algorithm is given below:

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**

In step 1, immediate neighbours of macros are those that can potentially abut. They are associated with the non-transistive 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.

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. After shifting a cluster, a macro m may potentially reach its target position and should be detached from the cluster if it does not have other macros in the same cluster on its right. This condition can be checked by looking at the immediate right neighbours of m.

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 $O(n^2)$. Figures 3 and 4 show the placement of the macros before and after legalization for the circuit ibm01. It can be observed that the macros have moved by a very small amount from their original positions obtained after global placement.

B. Standard Cell Legalization

Once the overlaps among the 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 segments to satisfy their respective capacities. Finally, we legalize the standard cells within the segments.

For moving a cell, we compute 8 *scores* based on moving it to its nearest 8 neighboring segments. For calculating the score, we assume that the cell is moving from its current position in a *source* segment to the nearest possible position in the *target* segment. Each score is 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 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 placement 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.

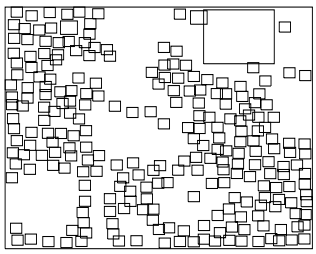

Fig. 3. Circuit ibm01 before legalization of movable macros.

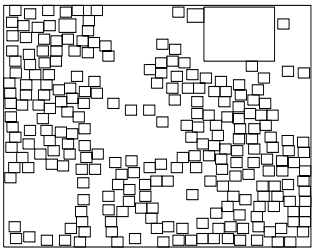

Fig. 4. Circuit ibm01 after legalization of movable macros.

IV. Detailed Placement

The aim of the detailed placement stage is to further reduced the wirelength of the placement. We adopt the detailed placement algorithm described in [15] 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 [16] to swap the standard cells for wirelength reduction. This technique operates on the entire placement 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.

TABLE I

Placement benchmark statistics.

Circuit	#Cells	#Macros	#Pads	#Nets	%Cell Area	%Macro Area
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
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
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

V. Experimental Results

Our algorithm was tested on the ISPD02 IBM-MS Mixed-size Placement Benchmarks [2], [3], [17]. 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.

In Table II, we compare our placer with *Capo 9.1* and *Fengshui 5.0*, which are updated versions of the tools published in [5] and [4] respectively. Both placers are run in their default mode. All the runtimes are on an Intel Xeon, 3.06GHz CPU. From Table II, we are on average, 11% and 2% better in terms of wirelength over *Capo 9.1* and *Fengshui 5.0* respectively. Correspondingly, we are $16.8X$ and $7.8X$ faster.

TABLE II

Comparison of our placement results with Capo 9.1 (CP) and Fengshui 5.0 (FS). (* Average is over all 18 circuits of the benchmark suite)

Ckt	HPWL			RunTime				
	Our	$\frac{CP}{Our}$	$\frac{FS}{Our}$	Our (sec)	CP (sec)	$\frac{CP}{Our}$	FS (sec)	$\frac{FS}{Our}$
ibm04	8.18	1.11	1.04	38	771	20.29	323	8.50
ibm05	10.24	1.00	0.96	35	684	19.54	372	10.63
ibm06	6.01	1.25	1.14	37	809	21.86	437	11.81
ibm10	31.78	1.17	1.10	180	2666	14.81	1085	6.03
ibm11	20.84	1.05	0.94	129	2172	16.84	891	6.91
ibm12	34.79	1.15	1.04	181	3413	18.86	1011	5.59
ibm16	58.95	1.20	1.02	422	7211	17.09	3626	8.59
ibm17	70.14	1.08	0.99	680	6782	9.97	3935	5.79
ibm18	46.08	1.03	0.96	770	5163	6.71	3471	4.51
Average*		**1.11**	**1.02**			**16.83**		**7.83**

References

[1] 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*, 2004, pp. 26–33.

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

[3] ——, "Consistent placement of macro-blocks using floorplanning and standard-cell placement," in *Proc. Intl. Symp. on Physical Design*, 2002, pp. 12–17.

[4] 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*, 2004, pp. 84–89.

[5] 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*, 2004, pp. 550–557.

[6] 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.*, 2003, pp. 325–330.

[7] H. Eisenmann and F. Johannes, "Generic global placement and floorplanning," in *Proc. ACM/IEEE Design Automation Conf.*, 1998, pp. 269–274.

[8] 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.*, 2000, pp. 271–276.

[9] 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*, 2004, pp. 565–572.

[10] 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*, 2004, pp. 573–580.

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

[12] 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*, 2005, pp. 193–199.

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

[14] 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*, vol. 15, no. 12, pp. 1518–1524, Dec. 1996.

[15] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, To Appear, 2005.

[16] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *IEEE Trans. Circuits and Systems*, vol. CAS-28, no. 1, pp. 12–18, 1981.

[17] S. N. Adya and I. L. Markov, "Combinatorial techniques for mixed-size placement," *ACM Trans. Design Automation of Electronics Systems*, vol. 10, no. 1, pp. 58–90, Jan. 2005.