

# An Improved Multi-Level Framework for Force-Directed Placement \*

Kristofer Vorwerk and Andrew Kennings  
Department of E&CE  
University of Waterloo  
Waterloo, Ontario, Canada  
{kpvorwer,akenning}@cheetah.vlsi.uwaterloo.ca

## Abstract

*One of the greatest impediments to achieving high quality placements using force-directed methods lies in the large amount of overlap initially present in these techniques. This overlap makes the determination of cell ordering difficult and can lead to the inadvertent separation of highly-connected cells by the spreading forces. We show that a multi-level clustering strategy can minimize the ill effects of overlap and improve the quality of placements generated by the force-directed tool FDP. Moreover, we present a means of improving initial cell ordering through the unification of min-cut partitioning and force-based placement, and describe an enhanced median improvement heuristic which further aids in minimizing HPWL. Numerical results are presented showing that our flow generates placements which are, on average, 15% better than mPG and 4% better than Capo 9.0 on mixed-size designs.*

## 1. Introduction

As problem instances have increased in size and complexity, placement has become the bottleneck in deep sub-micron design. Typically, placement seeks to minimize wire length subject to the constraint that cells be placed into prescribed locations without overlap. Several different approaches to this problem are possible, including simulated annealing [15], top-down partitioning [4, 11], and analytic techniques [7, 9, 16, 17]. Combinations of these techniques are often employed; for example, GORDIAN [12], GORDIAN-L [14], and BonnPlace [18] combine partitioning and analytic placement. Likewise, Dragon [13] combines partitioning and simulated annealing. These methods, however, have largely been employed for standard cell placement.

Several approaches have been proposed to address mixed-size placement. Very good results using a chiefly partitioning-based flow are reported in [2, 11], while [9] reports good results using analytic methods. Force-directed techniques [7, 17] are also of interest due to their seamless handling of macro cells, though run-times can be larger than partitioning-based strategies. Furthermore, the quality of placements obtained by force-directed methods falls short of recent work [2, 11].

This paper addresses the issue of quality in force-directed placement by describing three techniques to augment the open-source tool FDP [17]. Section 2 presents a brief overview of force-directed placement. In Section 3, we describe the implementation of a multi-level physical clustering strategy. Section 4 presents a novel unification of partitioning and force-directed methodologies which further improves quality. In Section 5, we discuss a ripple-move enhancement to the median improvement heuristic BoxPlace [17]. Finally, Section 6 presents numerical results and Section 7 concludes the work.<sup>1</sup>

## 2. Background

A circuit is modeled as a hypergraph  $G_h(V_h, E_h)$  with vertices  $V_h = \{v_1, v_2, \dots, v_n\}$  representing cells and hyperedges  $E_h = \{e_1, e_2, \dots, e_m\}$  corresponding to signal nets. Vertices are weighted by cell area while hyperedges are weighted according to criticalities or multiplicities. Vertices are either free or fixed. Circuit hypergraphs are typically transformed into graphs in which each hyperedge is represented by a set of equally-weighted edges. Cell placements in the  $x$ - and  $y$ -directions are captured by placement vectors  $x = (x_1, x_2, \dots, x_n)$  and  $y = (y_1, y_2, \dots, y_n)$ .

Wire length minimization is accomplished by solving the

\* This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant #203763-03, a grant from Altera Corporation, and a grant from CITO.

<sup>1</sup> Our source code has been included in FDP, and is available for free download at <http://gibbon.uwaterloo.ca>.

quadratic optimization problem ( $x$ -direction only) given by

$$\min_x \frac{1}{2} \mathbf{x}^T \mathbf{Q}_x \mathbf{x} + \mathbf{c}_x^T \mathbf{x} + \mathbf{f}_x^T \mathbf{x} + \mathbf{d}_x. \quad (1)$$

The matrix  $\mathbf{Q}_x$  is the Hessian which encapsulates the hyperedge connectivities. The vector  $\mathbf{c}_x$  is a result of fixed cell-to-free cell connections, and the vector  $\mathbf{d}_x$  is a result of fixed cell-to-fixed cell connections. The vector  $\mathbf{f}_x$  represents the vector of spreading forces which is used to perturb the placement to remove overlap. This optimization problem is strictly convex and has a unique minimizer given by the solution of a single, positive-definite system of linear equations,  $\mathbf{Q}_x \mathbf{x} + \mathbf{c}_x + \mathbf{f}_x = 0$ .

In FDP, the BoxPlace heuristic is employed to (occasionally) move each cell to the median position of its connected nets, reducing wire length directly [17]. For each cell in the netlist, BoxPlace determines a range (or “box”) into which a cell should be moved to minimize wire length. In FDP, only six passes of the netlist are made in any call to BoxPlace; consequently, the amount of overlap that is reintroduced is kept within reasonable bounds. Moreover, cell spreading is re-evaluated after each pass—if more than 2% overlap is reintroduced, the previous cell positions are restored, and the algorithm stops.<sup>2</sup> BoxPlace is also used to compute “wire length minimizing” forces which are combined (vectorally) with spreading forces to achieve a mix between cell spreading and wire length minimization.

### 3. Clustering and the Multi-Level Paradigm

Empirical evidence suggests that the quality of placements obtained by FDP relies *heavily* on cells’ initial “left/right” (and “up/down”) ordering, which is determined chiefly by the solution to the first (unperturbed) quadratic problem (QP). We have observed that most circuits begin with significant cell overlap—between 85% and 98%—after solving the first QP, with many thousands of cells being placed on top of each other. While advocates of QP-based placement contend that these methods benefit from more detailed cell positions (versus their partitioning-based counterparts), it is arguable whether or not a placement with 85%–98% overlap offers much reliability in terms of cell positions. With so many overlapping cells, the determination of which cell should be placed to the left or right of another is governed primarily by the spreading forces. Furthermore, when a cell is placed in-between two highly-connected cells, a “barrier” is created which generates spreading forces that push the connected cells *apart*, despite their connectivity. While BoxPlace has been noted to improve this “left/right” ordering

and to reduce the “damage” to wire length due to spreading forces [17], the ordering problem persists due to the extremely large number of initially-overlapping cells.

We have found that a multi-level clustering approach can significantly improve FDP’s placement quality. Historically, multi-level clustering has been used with partitioning- and annealing-based methods [6]. Force-directed techniques, on the other hand, are usually seen as working best on flat netlists [7] or when used to seed a top-down placement strategy [12, 14, 19]. However, we have found that multi-level clustering minimizes the negative impact of spreading forces by helping to keep together highly-connected cells.

#### 3.1. Hybrid First Choice Clustering

We use the Hybrid First Choice [10] clustering method. In this approach, cells are initially placed onto a “free list” containing the set of cells which have not been clustered. The *affinity* for pairing a node  $i$  with a node  $j$  is calculated for all shared edges using

$$r_{ij} = \sum_{e \in E_h | i, j \in e} \frac{1}{|e| - 1}. \quad (2)$$

The algorithm repeatedly removes the node with the highest affinity from the free list and pairs it with the node that (originally) yielded this high affinity, even if that node has already been paired. Once a node has been paired, it is said to form a “cluster”. In First Choice Clustering, an unpaired node is always paired with either another unpaired node or a cluster. The term “hybrid” refers to the fact that this aggregation heuristic reorders cells in the free list to improve the likelihood of greedily forming the best-possible pairings as early as possible.

Limits are placed on both the number and area of clusters to ensure that netlists are not reduced in size too quickly. We have found that it suffices to *not* match a node  $i$  with a cluster  $j$  if the aggregate area of the pairing (the sum of the area of node  $i$  and all of the paired nodes contained within the cluster) would exceed a multiple of the average cell area of the netlist. In such a circumstance,  $i$  and  $j$  would not be matched, and node  $i$  would simply be paired with its next best affinity match which satisfies this area constraint. (We note that the affinity equation could alternatively be modified along the lines of [5] to encourage matchings between smaller cells.)

We augmented (2) to account for physical cell locations in the hopes of encouraging the formation of better clusters. That is, we use

$$r_{ij} = \lambda \times \sum_{e \in E_h | i, j \in e} \frac{1}{1 + |x_i - x_j| + |y_i - y_j| - \zeta} + (1 - \lambda) \times \sum_{e \in E_h | i, j \in e} \frac{1}{|e| - 1} \quad (3)$$

<sup>2</sup> In this paper, we measure cell overlap using Klee’s technique [17].

to compute the affinity  $r$  between two connected cells  $i$  and  $j$  located at  $(x_i, y_i)$  and  $(x_j, y_j)$ , respectively. The equation consists of two parts: the first assigns an affinity based on the distance between cells, and the second implements the edge-coarsening affinity of (2). In the physical clustering term of (3), the variable  $\zeta$  represents the minimum Manhattan distance between any two cells on a shared net. This ensures that the best possible physical pairings (i.e., those which are formed between the closest cells) receive an affinity of 1, as would a 2-pin net in the edge-coarsening term. The parameter  $\lambda$  is used to control the preference between physical and edge-based clustering. We have found that physical clustering works best as a means of “breaking ties” between nets which share the same edge-coarsening affinity. In practice,  $\lambda \approx 0.25$  does a good job of implementing this “tie-breaking” mechanism.

### 3.2. Multi-Level Placement

Based on the aforementioned clustering strategy, we developed the multi-level flow presented in Figure 1. To our knowledge, this is the first paper to discuss the use of a multi-level scheme within the context of force-directed placement. (Only a small amount of non-multi-level clustering was previously employed in FDP.)

In our approach, a QP is solved to determine initial cell positions and the netlist is clustered to 5000 cells using the physical clustering technique. (In Section 4, we discuss how a QP augmented with cutlines derived from min-cut partitioning can further reduce the ambiguity in the initial cell positions.) At most 35% of cells are aggregated in one pass, so several “levels” of clustered netlists are created (forming a clustering “tree”), with the bottom-most netlist containing approximately 5000 cells. At each level, clusters are positioned at the average location of their contained cells. The most-clustered netlist is then placed to a stopping value of approximately 30% overlap, after which a small amount of greedy swapping is performed between clusters.

The netlist is subsequently declustered, with cells in the flat netlist placed at the centers of their former clusters. BoxPlace is called to move cells in the flat netlist to improve wire length. Subsequently, the entire circuit is *reclustered* up to a maximum of *one less* level than the number of levels required for the previous clustering tree. As before, force-directed placement is used to reduce overlap between the clusters to approximately 30% and the process repeats. The placer proceeds to legalization when there are no longer any clustered netlists remaining and overlap in the flat netlist has been reduced to approximately 30%.

Two points are worth noting about this approach. First, by repeatedly decrementing the maximum number of levels of clustering, FDP places increasingly larger netlists until it ultimately places the original (flat) netlist. Second, the

```

1 Procedure: MULTI-LEVEL PLACEMENT
2 begin
3   Determine initial cell positions; e.g, by solving a QP;
4   NumLevels  $\leftarrow \infty$ ;
5   while NumLevels > 0 do
6     Recluster netlist to at most NumLevels levels;
7     NumLevels  $\leftarrow$  number of levels in current clustering tree;
8     Place the most clustered netlist to  $\approx 30\%$  overlap;
9     Perform greedy improvement on clustered netlists;
10    Decluster netlist (placing cells at centers of clusters);
11    Delete the clustering tree;
12    Call BoxPlace on flat netlist;
13    NumLevels  $\leftarrow$  NumLevels - 1;
14  od
15 end

```

Figure 1. Pseudocode for our multi-level flow.

use of BoxPlace and reclustering allow the physical affinity term in (3) to more accurately decide which cells should be paired in subsequent clusters, which, in turn, improves the quality of the clustering.

### 3.3. Commentary

Figure 2 illustrates the reduction in overlap versus iteration count in FDP for a subset of the circuits in the ISPD02 IBM-MS benchmark [1]. (Other designs are omitted for clarity, but exhibit similar characteristics.) Three distinct phases are apparent in FDP’s multi-level flow. Placements begin with significant overlap and many iterations are required prior to any measurable cell spreading. As designs spread, a small amount of overlap is occasionally reintroduced due to the median improvement. Finally, overlap is re-introduced and removed as declustering and placement refinement on the declustered netlists are performed.

As mentioned in [8, 16], the run-time in force-directed techniques can be attributed, in part, to numerous iterations and the potentially large computational effort required per

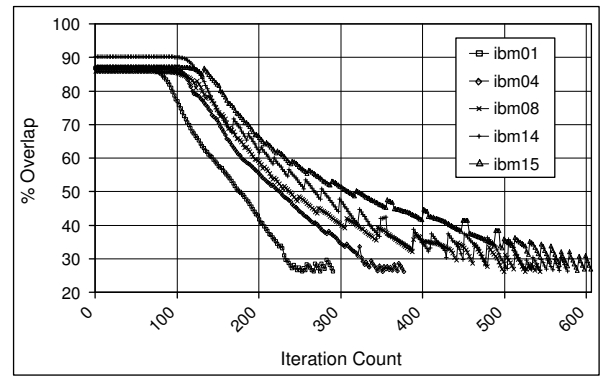


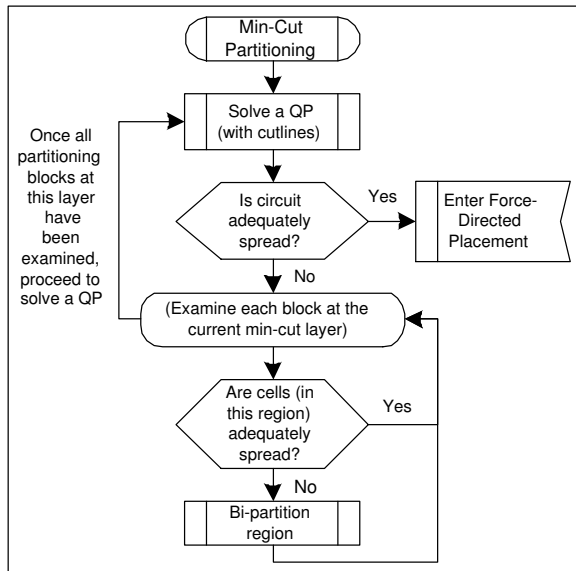
Figure 2. Overlap versus iteration count in FDP for different mixed-size circuits.

iteration. As suggested by Figure 2, there are a number of seemingly wasted iterations (during the first phase) prior to any significant overlap removal. This occurs because force weights are initially kept small and increased slowly to ensure that proper “left/right” ordering between cells is maintained [17]. With multi-level clustering, more iterations are typically required to purge overlap among the numerous clustered levels; however, the effort per iteration is reduced because of the smaller size of the clustered netlists.

#### 4. Unified Partitioning and Force-Directed Placement

We have found that a small amount of min-cut partitioning-based placement can improve circuits’ initial cell ordering, thereby leading to better results within FDP’s primarily force-directed design flow. In our implementation, the netlist is bi-partitioned *prior* to force-directed placement, as shown in Figure 3.

At each min-cut layer, we solve a QP augmented with cutlines from the partitioned regions. Then, we use the Klee’s measure technique [17] to assess the resultant amount of cell spreading in each partitioned region. We continue to recursively bi-partition the netlist to minimize cell overlap—based on empirical evidence, regions with macro cells are partitioned until they possess less than 20% overlap, while regions containing standard cells are partitioned to less than 90% overlap. These



**Figure 3. A small amount of min-cut partitioning employed *prior* to force-directed placement helps to improve cell ordering and reduce starting overlap.**

rules ensure that the largest macro cells are *separated* (allotted to different partition regions) prior to force-directed placement; in so doing, overlap is improved since macros are less likely to collapse on top of each other during the solution of the QP. After min-cut placement, FDP’s multi-level force-directed placement is employed to reduce the remaining overlap to approximately 30%. Thus, the partitioning problem *merely serves to enhance initial cell positions*. We note that our criteria for handing off a partially generated min-cut placement to FDP are quite different—but complementary—to those criteria where a min-cut placer should hand off to a floorplanner [2].

In this approach, quadratic placement and Klee’s measure technique are used in conjunction to assess overlap and identify potential candidate regions for partitioning. Likewise, min-cut partitioning helps the force-directed placer by providing better starting locations for cells. While this flow may seem similar to Capo Flow 2-A in [1] (which used Capo 8.8 and Kraftwerk-ECO), our approach applies partitioning to *selective* regions which are identified as high-overlap “problem zones”. (We also note that FDP appears to offer superior results compared to Kraftwerk.) We have observed that partitioning deeply (i.e., doing a primarily min-cut placement) followed by FDP achieves worse results than selectively partitioning those regions with high overlap and handing off to FDP once the problem areas have been adequately spread. This is similar, conceptually, to Capo 9.0 [2] in which a min-cut partitioner situates cells globally and a floorplanner refines their positions.

Using a partitioner to *seed* the force-directed placer explicitly helps with the cell ordering problem and improves the accuracy of FDP’s physical reclustering. Moreover, since cells are not constrained to partitioned regions during force-directed placement, it is felt that FDP is better able to replace cells previously assigned to sub-optimal regions during min-cut placement. Furthermore, placements exhibit less initial overlap and therefore converge to a fairly non-overlapping placement more quickly, effectively bypassing the first phase of placement illustrated in Figure 2.

#### 5. Better Median Improvement

We have observed that BoxPlace works well early in placement, when there exists between 70% and 100% overlap. However, once a circuit possesses less than  $\approx 70\%$ , BoxPlace tends to reintroduce too much overlap, and cell positions are generally restored. Thus, BoxPlace is not as effective in mid-to-late placement.

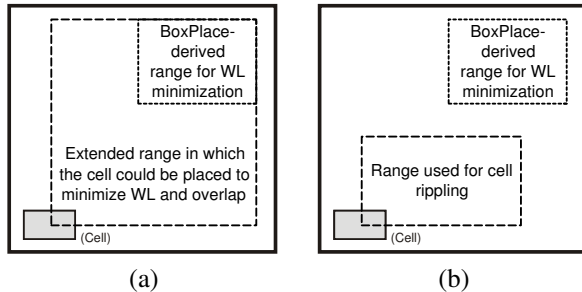
We enhanced BoxPlace so that it does not reintroduce as much overlap into the placement. Our method consists, in part, of overlaying a fine grid (which tracks cell positions) to ensure that cells are placed into relatively under-

occupied spots within their target ranges. However, this “occupancy grid” does not completely alleviate the reintroduction of overlap, as the ranges determined by BoxPlace for each cell may themselves be highly overlapping. So, in addition to the grid, we extend the *range* over which cells can move—that is, we extend the boxes such that any position between a cell’s current location and its range’s outer extremities are also included. This concept is illustrated in Figure 4 (a). Any movement of a cell in the *x*- or *y*-direction toward its minimizing “box” results in at least the same or better half-perimeter wire length (HPWL).

Using BoxPlace and the occupancy grid proposed here, we have developed a cell rippling strategy that can be employed in *every* iteration of force-directed placement. To accomplish this, we iterate through the netlist, using BoxPlace to compute a range for a given cell. However, we modify the range such that it emanates from the center of the cell up to, at most, three times the cell’s height or width. This concept is illustrated in Figure 4 (b). The more passes over the netlist that are performed, the more cells are “rippled” in the direction of minimum HPWL—due to the occupancy grid, overlap is not reintroduced, as cells are placed within the least overlapping areas within their range. Often, and especially late in placement, the least overlapping spot within a cell’s range is the same position in which the cell is presently located; thus, the algorithm usually does no worse, overlap-wise, than the current placement.

## 6. Experimental Results

We use a simple strategy to legalize placements produced by FDP [17]. Macro cells are processed using sequence pair analysis along the lines of [3]. Based on the results of the sequence pair analysis, macro cells are shifted to align with rows to remove overlap in the *y*-direction, and shifted to the left and right to remove overlap in the *x*-direction. Standard cells are then snapped to their closest rows to minimize total cell movement. Greedy juggling of standard cells is per-



**Figure 4. Illustration of the extended BoxPlace search range used (a) during individual calls to BoxPlace, and (b) when employed to ripple cells.**

formed between rows to meet width restrictions imposed by the fixed die. Finally, we apply greedy (same-size) swapping and single-row branch and bound on groups containing six or fewer standard cells. No attempt is currently made to optimize whitespace in the final placement.

### 6.1. Numerical Results

We compare FDP to mPG [5], Capo 9.0 (with feedback) [2] and Feng Shui 2.61-beta2 [11] on the IBM-MS mixed-size benchmark [1].<sup>3</sup> Results are presented in Table 1.<sup>4</sup> Run-times for Capo, Feng Shui, and FDP are observed on a dual-processor 3.2 GHz Xeon, and are reported in minutes. Run-times for mPG are observed on a 750 MHz Sun Blade 1000. Wire lengths correspond to HPWL divided by  $10^6$ . The column “FDP C+P” includes all of the multi-level clustering, BoxPlace, and partitioning enhancements discussed in this paper; the configuration for “FDP C” is similar, but partitioning is disabled.

We observe that our placer produces results that are, on average, 15% better than mPG and 4% better than Capo 9.0. The majority of the improvement stems from our multi-level clustering strategy, with our min-cut partitioning enhancements accounting for approximately 3%. We make particular note of our results compared to mPG, as it is also a multi-level placer.

On the other hand, our tool produces results that are 5% worse than Feng Shui. While the wire lengths of Feng Shui are excellent, differences in whitespace allocation may account for some of the discrepancies in quality. In addition, Feng Shui excels on ibm14, ibm17, and ibm18—these circuits possess the lowest ratios of macro cell-to-standard cell area in the benchmark suite (26.72%, 23.78%, and 11.97%, respectively). As a result, it is possible that Feng Shui’s tighter packing and better detailed improvement contribute to some of the differences in these cases.

The average run-time for “FDP C+P” versus Capo and Feng Shui is 1.58 and 2.61, respectively. The average run-time ratio of “FDP C+P” versus “FDP C” is 0.94. While the min-cut partitioning decreases FDP’s iteration count, its run-time advantage is mostly offset by the effort required to do the partitioning.

Although we present only mixed-size results in this paper, our modifications are by no means limited to such designs—they are equally admissible for standard cell circuits. Using the same standard cell benchmark as in [17], our average quality versus Capo for “FDP C+P” is 0.98,

<sup>3</sup> Currently, FDP does not optimize cell orientations; thus, we use the IBM-MS circuits *without* pins.

<sup>4</sup> While FastPlace [16] purports excellent run times, the tool is not available for download, and is not yet capable of placing mixed-size problems.

**Table 1. Mixed-size results comparing mPG, Capo, and Feng Shui to FDP with clustering (“C”) and both clustering and initial partitioning (“C+P”).**

| Circuit | mPG   |     | Capo 9.0 |     | FS 2.6 beta2 |     | FDP C |     | FDP C+P |     | FDP C+P vs. |      |      |       |
|---------|-------|-----|----------|-----|--------------|-----|-------|-----|---------|-----|-------------|------|------|-------|
|         | WL    | CPU | WL       | CPU | WL           | CPU | WL    | CPU | WL      | CPU | mPG         | Capo | FS   | FDP C |
| ibm01   | 3.01  | 18  | 2.65     | 4   | 2.52         | 2   | 2.50  | 7   | 2.45    | 7   | 0.81        | 0.92 | 0.97 | 0.98  |
| ibm02   | 7.42  | 32  | 5.27     | 4   | 5.20         | 4   | 5.52  | 12  | 5.61    | 13  | 0.76        | 1.06 | 1.08 | 1.02  |
| ibm03   | 11.20 | 32  | 10.89    | 17  | 7.79         | 4   | 8.57  | 13  | 7.96    | 12  | 0.71        | 0.73 | 1.02 | 0.93  |
| ibm04   | 10.50 | 42  | 9.46     | 13  | 8.64         | 5   | 9.13  | 15  | 8.63    | 13  | 0.82        | 0.91 | 1.00 | 0.94  |
| ibm06   | 9.21  | 45  | 7.26     | 10  | 7.20         | 7   | 8.24  | 19  | 7.27    | 15  | 0.79        | 1.00 | 1.01 | 0.88  |
| ibm07   | 13.70 | 68  | 11.93    | 11  | 11.47        | 9   | 12.40 | 28  | 12.39   | 24  | 0.90        | 1.04 | 1.08 | 1.00  |
| ibm08   | 16.40 | 82  | 14.74    | 36  | 13.50        | 11  | 14.81 | 27  | 14.38   | 25  | 0.88        | 0.98 | 1.07 | 0.97  |
| ibm09   | 18.60 | 84  | 16.04    | 19  | 13.95        | 10  | 15.34 | 26  | 14.67   | 24  | 0.79        | 0.91 | 1.05 | 0.96  |
| ibm10   | 43.60 | 172 | 35.60    | 36  | 39.52        | 16  | 35.02 | 47  | 34.08   | 39  | 0.78        | 0.96 | 0.86 | 0.97  |
| ibm11   | 26.50 | 112 | 22.22    | 35  | 19.58        | 14  | 21.72 | 33  | 20.55   | 31  | 0.78        | 0.92 | 1.05 | 0.95  |
| ibm12   | 44.30 | 153 | 42.44    | 32  | 38.35        | 16  | 43.42 | 57  | 39.79   | 61  | 0.90        | 0.94 | 1.04 | 0.92  |
| ibm13   | 37.70 | 151 | 29.32    | 46  | 24.79        | 18  | 26.28 | 44  | 26.14   | 41  | 0.69        | 0.89 | 1.05 | 0.99  |
| ibm14   | 43.50 | 276 | 40.78    | 45  | 37.92        | 36  | 41.96 | 63  | 42.53   | 70  | 0.98        | 1.04 | 1.12 | 1.01  |
| ibm15   | 65.50 | 285 | 58.92    | 76  | 52.48        | 45  | 55.14 | 107 | 55.93   | 93  | 0.85        | 0.95 | 1.07 | 1.01  |
| ibm16   | 72.40 | 436 | 66.08    | 88  | 60.55        | 49  | 66.45 | 103 | 65.60   | 97  | 0.91        | 0.99 | 1.08 | 0.99  |
| ibm17   | 78.50 | 606 | 78.97    | 55  | 71.56        | 55  | 79.64 | 137 | 79.85   | 122 | 1.02        | 1.01 | 1.12 | 1.00  |
| ibm18   | 50.70 | 437 | 49.84    | 48  | 44.00        | 60  | 51.90 | 169 | 50.88   | 174 | 1.00        | 1.02 | 1.16 | 0.98  |
| Avg.    |       |     |          |     |              |     |       |     |         |     | 0.85        | 0.96 | 1.05 | 0.97  |

with run-times being 2.85 times those of Capo, on average. (We do not compare explicitly to Feng Shui, as it crashed on four of the eighteen circuits.) This represents a  $\approx 2\%$  improvement in quality over FDP’s previous standard-cell results [17].

## 7. Conclusions

This paper described a multi-level clustering strategy, a means of unifying partitioning and force-directed placement, and a rippling technique to further improve wire length. With all features combined, we produced results that were 4% better, on average, than Capo 9.0 (with feedback) and 15% better than mPG. We feel that there are still opportunities to improve the quality and performance of our flow, as well as to incorporate congestion minimization and timing within the infrastructure.

## References

- [1] S. Adya and I. Markov. Combinatorial techniques for mixed-size placement. *Trans. on DAES*, 2004. To appear.
- [2] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov. Unification of partitioning, placement and floorplanning. In *Proc. of ICCAD*, November 2004.
- [3] S. N. Adya and I. L. Markov. Fixed-outline floorplanning: Enabling hierarchical design. *Trans. on VLSI*, 11(6):1120–1135, December 2003.
- [4] A. E. Caldwell, A. B. Kahng, and I. Markov. Can recursive bisection alone produce routable placements? In *Proc. of DAC*, pages 477–482. ACM Press, 2000.
- [5] T. F. Chan, J. Cong, T. Kong, J. R. Shinnerl, and K. Sze. An enhanced multilevel algorithm for circuit placement. In *Proc. of ICCAD*, November 2003.
- [6] C.-C. Chang, J. Cong, and X. Yuan. Multi-level placement for large-scale IC designs. In *Proc. of ASPDAC*, pages 325–330, 2003.
- [7] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. of DAC*, pages 269–274. ACM Press, 1998.
- [8] S.-W. Hur, T. Cao, K. Rajagopal, Y. Parasuram, A. Chowdhary, V. Tiourin, and B. Halpin. Force directed Mongrel with physical net constraints. In *Proc. of DAC*, pages 214–219. ACM Press, 2003.
- [9] A. B. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. In *Proc. of ISPD*, pages 18–25, April 2004.
- [10] G. Karypis. *Multilevel Optimization and VLSICAD*, chapter 3. Kluwer Academic Publishers, Boston, 2002.
- [11] A. Khatkhat, 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. of ISPD*, April 2004.
- [12] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *Trans. on CAD*, 10(3):356–365, March 1991.
- [13] X. Y. M. Wang and M. Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. of ICCAD*, November 2000.
- [14] G. Sigl, K. Doll, and F. Johannes. Analytical placement: A linear or a quadratic objective function? In *Proc. of DAC*, pages 427–432, June 1991.
- [15] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *Proc. of DAC*, pages 211–215, 1995.
- [16] N. Viswanathan and C. C.-N. Chu. Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In *Proc. of ISPD*, April 2004.
- [17] K. Vorwerk, A. Kennings, and A. Vannelli. Engineering details of a stable analytic placer. In *Proc. of ICCAD*, November 2004.
- [18] J. Vygen. Algorithms for large-scale flat placement. In *Proc. of DAC*, pages 746–751. ACM Press, 1997.
- [19] J. Vygen. Four-way partitioning of two-dimensional sets. Unpublished Manuscript, 2000.