An Enhanced Multilevel Algorithm for Circuit Placement *

Tony F. Chan,† Jason Cong, Tim Kong,* Joseph R. Shinnerl, and Kenton Sze †
UCLA Computer Science Department {cong,shinnerl}@cs.ucla.edu
† UCLA Mathematics Department {chan,nksze}@math.ucla.edu
* Magma Design Automation kongtm@magma-da.com

ABSTRACT

This paper presents several important enhancements to the recently published multilevel placement package mPL [12]. The improvements include (i) unconstrained quadratic relaxation on small, noncontiguous subproblems at every level of the hierarchy; (ii) improved interpolation (declustering) based on techniques from algebraic multigrid (AMG), and (iii) iterated V-cycles with additional geometric information for aggregation in subsequent V-cycles. The enhanced version of mPL, named mPL2, improves the total wirelength result by about 12% compared to the original version. The attractive scalability properties of the mPL run time have been largely retained, and the overall run time remains very competitive. Compared to GORDIAN-L-DOMINO [25] on uniform-cell-size IBM/ISPD98 benchmarks, a speed-up of well over 8× on large circuits ($\geq 100,000$ cells or nets) is obtained along with an average improvement in total wirelength of about 2%. Compared to Dragon [32] on the same benchmarks, a speed-up of about 5× is obtained at the cost of about 4% increased wirelength. On the recently published PEKO synthetic benchmarks, mPL2 generates surprisingly high-quality placements — roughly 60% closer to the optimal than those produced by Capo 8.5 and Dragon — in run time about twice as long as Capo's and about 1/10th of Dragon's.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*placement and routing*; G.4 [**Mathematical Software**]: Algorithm Design and Analysis; J.6 [**Computer-Aided Engineering**]: Computer-Aided Design

*Financial Support from Semiconductor Research Consortium Contracts 98-TJ-686 and 2003-TJ-1019 and National Science Foundation Grant CCR-0096383 is gratefully acknowledged.

ICCAD '03 San Jose, California USA

Keywords

Placement, Multilevel Optimization, Quadratic Programming (QP), Algebraic Multigrid (AMG)

1. INTRODUCTION

Placement is a very important problem in VLSICAD, as it implicitly defines the interconnect, which has become the bottleneck in deep submicron designs. With problem instances approaching 10 million cells and nets, the complexity and difficulty of placement has also increased, making fast and scalable algorithms particularly important.

Traditionally, placement is separated into *global* and *detailed* phases. In global placement, an approximately uniform distribution of cells is sought. In detailed placement, nonoverlap constraints are strictly enforced. Both phases typically incorporate wirelength, timing, and routability into their objectives or constraints. Detailed placement is typically *constructive*: cells are explicitly arranged one by one in an overlap-free fashion.

The focus in mPL[1] is on global placement. The dominant paradigms for global placement algorithms include (i) recursive bisection or quadrisection, (ii) simulated annealing, (iii) analytical techniques based on linear and quadratic programming, and (iv) force-directed methods. Currently, most leading methods combine two or more of these approaches in a top-down framework [10, 32, 21, 31, 25, 19, 7].

mPL is based on multilevel optimization (Figure 1): recursive aggregation followed by interleaved optimization and disaggregation at every level [17]. Section 1.1 explains why this more general terminology is preferred to the simpler and more familiar "clustering" etc. mPL is designed for both scalability and correct handling of complex constraints, but the emphasis to date is wirelength minimization subject to nonoverlap constraints. Future versions will consider timing and routability.

1.1 The Multilevel Paradigm

The underlying ideas of multilevel methods originated in the development of algorithms for scalar elliptic partial differential equations (PDE) naturally discretized in space by regular grids [8, 3, 4, 27]. Since their inception, the methods have been successfully applied to many diverse problems in scientific computation, both discrete and continuous, including some in VLSICAD [17].

[1]Throughout this paper, the name "mPL" refers collectively to both mPL1 and mPL2.

In contrast to traditional top-down hierarchical approaches, multilevel algorithms construct their hierarchies from the bottom up by recursive aggregation [5, 6]. Aggregation (a.k.a. weighted aggregation) may be viewed as a general kind of *clustering* in which each finer-level element is associated by convex combination with several coarser-level elements rather than with just one. Disaggregation is called *interpolation*, as it transfers an approximate solution from a coarser domain to a finer one. In the symmetric positive-definite linear-system model problem $Ax = b$, aggregation ($R \in \mathbf{R}^{m \times n}$ with $m \approx n/2$) and interpolation (R^T) are formulated as linear operators, each the transpose of the other, so that the coarse-level equations can be written $RAR^T\hat{x} = Rb$, where $\hat{x}$ is the coarse-level solution, and $x = R^T\hat{x}$. In the case of linear systems of equations arising from discretized scalar elliptic PDEs, the benefits of weighted aggregation compared to simple clustering are mathematically well understood [30]. Clustering corresponds to piecewise-constant interpolation, weighted aggregation corresponds to piecewise-linear interpolation, and the latter produces superior error reduction and convergence rates independent of grid size. A key obstacle to the use of weighted aggregation in circuit placement, still unaddressed, is the problem of defining hyperedges for the aggregates without the average hyperedge degree or the relative number of hyperedges exploding at coarser levels.

Intralevel optimization is called *relaxation*. In a continuous setting, relaxation has the key property that it smooths the error, quickly eliminating high-frequency error components and leaving only slowly varying low-frequency components best reduced at coarser scales. Traditionally, relaxation is inexpensive and localized; the most common form for an $n \times n$ linear system $\sum_j a_{ij}x_j = b_i$ is simply the Gauss-Seidel iteration $x_i^{(k+1)} = b_i - \sum_{j \neq i} a_{ij}x_j^{(k)}$ for $i = 1, \ldots, n$. In the specialized linear-system case, limited application of localized relaxation at each level of an aggregation-based hierarchy leads to accurate solutions in optimal-order run time. The principles governing the convergence in this case are well understood. In the nonlinear, discrete setting, more diverse forms of relaxation exist and may be tailored to the problem at hand; however, the goal is still to produce local improvement appropriate to the given scale. Global improvement comes from the combination of improvements across the entire collection of levels.

Recent success in multilevel hypergraph partitioning [24, 23, 9, 1] has drawn attention to the multilevel framework as a vehicle for placement. Recently published multilevel placement algorithms ultra-fast VPR [28] and mPL [12, 11] have demonstrated potential for dramatic improvement in speed and scalability. The focus of our current research is the use of multilevel optimization techniques to improve overall solution quality as well. Our work is built on the framework of mPL1, reviewed in the next subsection.

We use the following notation and terminology. Let $H = (V, E)$ denote the weighted hypergraph-netlist representation of the given circuit. Elements of V are rectangles of fixed height and width called *cells* or *modules* or *vertices*. Elements of E are subsets of V called *nets* or *hyperedges*. To each $e \in E$ may be associated a weight $w(e) \geq 0$ chosen to reflect the importance of net e with respect to timing constraints or other criteria. Let $N = |V|$, the number of modules in the circuit. The wirelength of net e is estimated to be the half-perimeter of the smallest rectangle circumscribing its cells, its "bounding box." The total wirelength is the sum of these estimates over all nets $e \in E$.

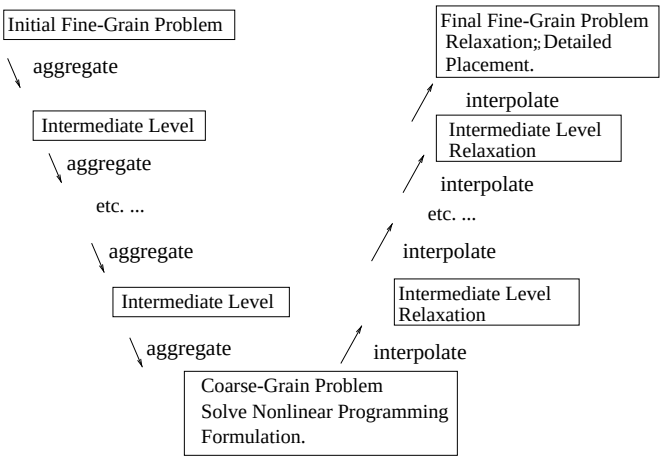

Figure 1: Multilevel Optimization V-Cycle

Representation $H_1 \equiv H$ is the *finest level* of the multilevel hierarchy. Initially, netlist connectivity is used to aggregate cells into clusters (Section 2). Hyperedges are then defined on the clusters in a natural way: if net $e = \{v_{i_1}, v_{i_2}, \ldots, v_{i_n}\}$, and the cells in net e are contained in clusters $\overline{v}_{j_1}, \ldots, \overline{v}_{j_m}$ with $m > 1$, then e is represented in the cluster hypergraph simply as $\overline{e} = \{\overline{v}_{j_1}, \ldots, \overline{v}_{j_m}\}$, where $1 \leq m \leq n$. If $m = 1$, net e is not transferred to the clusters. The resulting cluster hypergraph $H_2 \equiv (\overline{V}, \overline{E}) \equiv (V_2, E_2)$ is called a *coarsening* of H_1. Proceeding recursively, we obtain a hierarchy of hypergraph approximations $\{H_1, \ldots, H_L\}$, with $L \leq \log_2(N)$, and $H_L = (V_L, E_L)$ the coarsest, and therefore the smallest, approximation to H. In our implementations, we select L so that $|V_L| < 500$. As the representation at every level resembles the finest level in its structure, clusters are often also referred to simply as cells.

1.2 Review of mPL1

Aggregation in mPL1 [12, 11] is performed by edge-separability clustering [16]. At the coarsest level(s) only, a customized interior-point method is applied to a nonlinear programming formulation initialized by quadratic minimization with a single center-of-mass constraint, cell area interpreted as mass. The purpose is to obtain a solution of the highest possible quality at this level with only an approximately uniform area distribution. Interpolation or disaggregation from a coarser level to an adjacent finer level proceeds as follows: (i) the child clusters of a common parent are placed concentrically at that parent cluster's center; (ii) clusters are recursively partitioned in cutsize-minimizing fashion into blocks numbering 500 cells or fewer; (iii) a uniform slot grid is defined on each block, and displacement-minimizing linear assignment of cells to slots is performed under a typically invalid assumption of uniform cell sizes. Following interpolation, sweeps of randomized discrete cell exchanges are performed greedily on subsets whose union covers the entire circuit [20, 12]. The first version of mPL requires a detailed placement engine such as DOMINO [18] be run separately after global placement.

1.3 Improvements in mPL2

The following enhancements to mPL1 have been implemented in mPL2.

(i) First-Choice Clustering (FC) is used to aggregate cells and clusters [22, 23, 13] by a variety of affinity metrics (Section 2).

(ii) Unconstrained quadratic minimization is combined with bin-based area balancing on sequences of small subsets of clusters for scalable, continuous relaxation at *every* level (Section 3).

(iii) Weighted-average disaggregation is used instead of strict, concentric disaggregation to initially interpolate a coarse-level solution to its adjacent finer level (Section 4).

(iv) Multiple iterations over the multilevel hierarchy iteratively improve the multilevel solution (Section 5).

Overall, these enhancements improve the total wirelength obtained by mPL1 by about 12% at the cost of about 2.6× increase in run time. This factor, though perhaps larger than desired, does not appear to increase with circuit sizes, and the mPL2 run time remains very competitive, when compared with other well-known placers such as Gordian-L [25], Capo [10], and Dragon [32].

2. FIRST-CHOICE AGGREGATION

In this scheme, each vertex (cell) is paired with a neighboring vertex with which it shares the strongest "affinity," regardless of whether that adjacent vertex has already been paired with some other vertex [22, 23, 13]. Connected components defined by this association are used as clusters. In mPL2, the affinity r_{ij} between vertices i and j has the initial form

$$r_{ij} = \sum_{\{e \in E \mid i,j \in e\}} \frac{w(e)}{(|e|-1)\text{area}(e)},$$

where $H = (V, E)$ is the underlying weighted hypergraph representation; $w(e)$ is the weight associated to hyperedge e, area(e) denotes the sum of the areas of the cells (vertices) in e, and $|e|$ denotes the number of vertices in hyperedge e. The purpose of weighted area in the denominator is to give higher affinity to smaller cells joined by smaller nets. By encouraging the merging of smaller cells, we encourage a more even distribution of cluster areas.[2] However, there is still substantial area variation, and this variation tends to increase with each level of aggregation, so that even for initially uniform cell areas, cluster areas at the coarsest level may vary by factors of 1000 or more.

In V-cycles subsequent to the first, affinity r_{ij} is divided by the distance between v_i and v_j in order to preserve some placement information from previous cycles (Section 5). Clusters are placed at the weighted average of the components' positions. For the purpose of defining the coarse-level hyperedges, however, the mPL2 aggregation is still viewed as a simple clustering.

[2]This use of area distinguishes the mPL affinity from the mPG affinity [13].

3. QUADRATIC RELAXATION ON NONCONTIGUOUS SUBSETS (QRS)

At each level of the interpolation phase, sweeps of unconstrained quadratic relaxations on small, noncontiguous movable subsets of cells are applied. Let M denote a designated set of movable cells. We denote the set of nets containing at least one movable cell by $\mathcal{E}_M$ and the set of fixed cells in $\mathcal{E}_M$ by F. That is, $F = \bigcup_{e \in \mathcal{E}_M} e \setminus M$. For each movable set M, we minimize the total weighted quadratic-star wirelength of $\mathcal{E}_M$.

For each net $e \in \mathcal{E}$, let $|e|$ denote the number of cells in net e, $(x(v), y(v))$ the center of cell v, and let

$$(x_e,\, y_e) = \left(\frac{1}{|e|} \sum_{\text{cells } v \in e} x(v), \quad \frac{1}{|e|} \sum_{\text{cells } v \in e} y(v) \right).$$

By weighted quadratic-star wirelength, we mean

$$\ell_2^* = \frac{1}{2} \sum_{e \in \mathcal{E}_M} \sum_{v \in e} \frac{(x(v) - x_e)^2}{|x^{(k)}(v) - x_e^{(k)}|} + \frac{(y(v) - y_e)^2}{|y^{(k)}(v) - y_e^{(k)}|}$$

where the fixed-constant displacement weights $1/|x^{(k)}(v) - x_e^{(k)}|$ and $1/|y^{(k)}(v) - y_e^{(k)}|$ are used at each iteration after the first, as in Gordian-L [29], in order to help the objective gradually approximate a linear-star wirelength as closely as possible in a smooth way. In the very first iteration, the weights are not used, and the objective is simply quadratic star wirelength. Currently we employ just five such iterations for each movable subset.

The movable subsets are obtained as segments of length 3 along a DFS vertex traversal of the netlist. The traversal begins with a vertex v such that the sum of the wirelengths of the nets containing v is maximal. Although larger movable subsets may produce lower wirelengths at the given level, they do not appear to be cost-effective within the multilevel framework.

Because there are typically many fixed cells in $\mathcal{E}_M$, the movable cells tend to remain separated and not move unreasonably far. However, some increase in overlap may be incurred, as there is currently no area-congestion modeling included in the subproblem objective. Following Mongrel [21], the movable cells are, therefore, not all moved to their relaxed locations right away. Instead, they are moved one at a time, and after each such move, the bin-based area density constraint(s) for the destination bin(s) are examined (a cell may be too large to fit in a single destination bin). If any such bin-density constraint is violated, its bin's area is reduced by cell-by-cell "ripple-move" propagation along monotone paths from overfull bins to underfull bins until feasibility is restored (Figure 2).

Currently, quadratic relaxation in mPL2 amounts simply to three cell displacements, each possibly followed by ripple-move corrections to area-congestion. Following each such subset relaxation, the net wirelength change is checked. If the relaxation increases the wirelength, its steps are reversed, and relaxation on the next movable subset in the DFS sequence of subsets proceeds. This monotone-non-increasing-wirelength strategy was observed to produce better results in our multilevel implementation than the alternative FM-like hill-climbing strategy used in Mongrel [21]. In that approach, all relaxations are initially accepted, but at the end of the entire sweep, the configuration following the one that produced the least wirelength is restored.

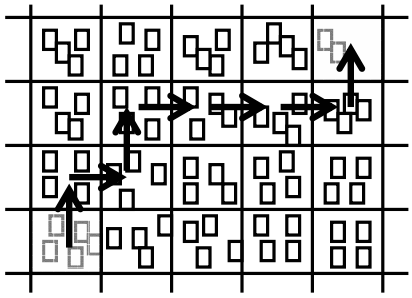

Figure 2: Ripple-move area-congestion control.

The impact of QRS + ripple-move overlap removal on mPL2's performance is illustrated in Table 1. Overall, wirelength is reduced by about 5.9% at a cost of about 60% increased overall run time. Further experiments (not shown) show that almost all the run time increase is due to the ripple-move overlap removal. To reduce ripple-move runtime, the bin sizes were increased linearly with respect to circuit size (number of cells), and searches for underfull bins were limited to windows of 25 bins × 25 bins. About 95% of searches within these windows are successful; other searches are simply terminated, and the overfull bin is left alone. However, we expect that further adjustments to the formulation of the quadratic subproblem will improve performance. In future work, we plan to incorporate overlap-area-based force-directed terms into the quadratic as in Kraftwerk [19].

circuit	wirelength improved	run time increased	bin capacity
ibm04	9.08%	29.60%	2x2
ibm07	8.77%	45.61%	2x2
ibm09	7.38%	47.75%	3x3
ibm10	3.55%	57.05%	3x3
ibm14	4.15%	65.88%	3x3
ibm16	4.17%	65.29%	4x4
ibm17	4.40%	66.57%	4x4
ibm18	5.95%	86.77%	4x4
averages	5.93%	58.07%	

Table 1: Impact of QRS relaxation + ripple-move overlap removal.

4. AMG-BASED INTERPOLATION

Although mPL2 still uses simple clustering rather than weighted aggregation to build its multilevel hierarchies, it does use weighted disaggregation (a.k.a. weighted interpolation) rather than simple declustering to take them apart (Figure 3). In declustering, the center of each child cluster is placed at the center of its parent. In mPL2, some components of most clusters are placed at the weighted averages of the positions of all clusters with which they share sufficient connectivity (Figure 3). In this way, possible damage caused by premature associations made during clustering is reduced. Further motivation for weighted interpolation comes from well established results for continuous problems,

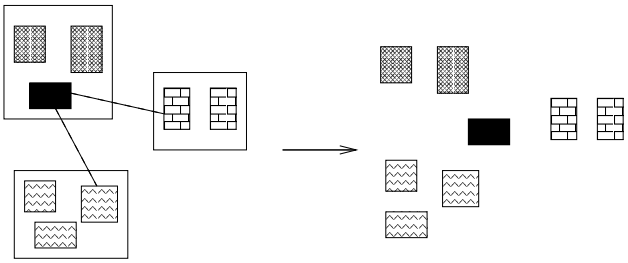

Figure 3: AMG-based Weighted Disaggregation

circuit	improvement by AMG: wirelength % improved	improvement by AMG: run time % increased
ibm04	4.93%	63.60%
ibm07	5.08%	10.35%
ibm09	0.78%	23.74%
ibm10	1.42%	12.56%
ibm14	5.73%	12.31%
ibm16	2.55%	13.43%
ibm17	1.10%	5.82%
ibm18	0.52%	10.98%
average	2.76%	19.10%

Table 2: Impact of AMG Interpolation; 1 V-cycle without QRS.

as mentioned in Section 1.1.

A graph model of connectivity is employed to define the interpolation: the weight of edge e_{ij} is

$$w(e_{ij}) = \sum_{\{e \in E \mid i,j \in e\}} \frac{w(e)}{(|e|-1)}. \quad (1)$$

For efficiency, only weights above a certain threshold (currently 1/4) are used. Finer-level vertices v_i with the highest total connectivity $\sum_j w(e_{ij})$ are designated as "C-points" and are given the positions of their parent clusters. The remaining points are designated as "F-points" and are placed at the weighted average of the positions of the C-points to which they are connected. Once an F-point has been placed, it can be treated like a C-point and used to influence the positioning of other F-points to which it has connections. Moreover, since the process depends on the vertex order, iterations may be used to allow all interconnected nodes to influence each others' positions. For this purpose, the nodes are ordered by decreasing connectivity $w(v_i) = \sum_j w(e_{ij})$, following (1).

The net impact of AMG-based disaggregation on mPL's performance is illustrated in Table 2. Overall, wirelength is reduced by about 3% at the very modest cost of about a 20% increase in run time, this increase decreasing with problem size.

5. MULTILEVEL FLOW

The final important enhancement to mPL2 is improved iteration flow. After the first V-cycle, an additional V-cycle is used to improve the result; the results are shown in Table 3. During the reaggregation phase, spatial proximity is

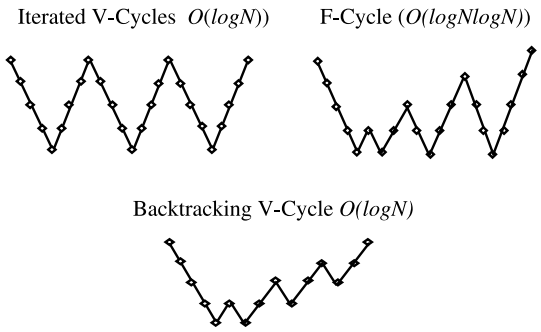

Figure 4: Iterated Multilevel Flow Alternatives

circuit	rel. time	% improve
ibm04	1.10	2.32%
ibm07	1.40	3.57%
ibm09	1.46	4.69%
ibm10	1.49	1.44%
ibm14	1.28	1.64%
ibm16	1.31	2.27%
ibm17	1.30	7.43%
ibm18	1.26	5.36%
average	1.33	3.59%

Table 3: Impact of second vcycle.

used in the FC affinity along with netlist connectivity. Thus, clusters are placed at the weighted average of their components' positions, the weights identical to those used in the interpolation (1). Relaxation on this modified hierarchy is then used to further reduce the wirelength. Experiments so far conducted suggest that, in the current implementation, a second V-cycle is more cost-effective than the more elaborate "F-cycle" (known as *full multigrid*, or *FMG*), in which every level of relaxation during the interpolation phase is followed by a complete reaggregation to the coarsest level (Figure 4). However, we continue to seek alternative and more diverse forms of relaxation for which the F-cycle may yet prove more effective. We also tried a compromise strategy of limited "backtracking" reaggregation but did not find it effective.

6. OTHER DIFFERENCES

There are a few other minor differences between mPL1 and mPL2.

1. mPL1 uses Domino [18] for detailed placement. For comparisons on benchmarks in Bookshelf format, a very simple detailed placement algorithm based on randomized local cell swaps is used in mPL2 instead of Domino.
2. Prior to Goto relaxation, mPL1 assumes all cells are the same size in order to partition them into blocks and then assign them to nearby slots. To lessen the impact of the uniform-cell-size assumption, mPL2 chops larger cells into uniformly sized pieces and allows the pieces to be assigned separately. A weighted, artificial net connecting the pieces is used to keep them from drifting too far apart.

Neither of these modifications was observed to have more than 1–2% total impact on either wirelength or run time.

7. RESULTS

The performance of mPL2 was compared to that of a few leading academic placement tools on two different sets of benchmarks:

(i) The IBM/ISPD98 suite [2] with all cells uniformly sized, in either PROUD or Bookshelf format.

(ii) The PEKO suite [14] (`http://ballade.cs.ucla.edu/~pubbench/peko.htm`). These circuits are constructed to match the net-degree distribution of the IBM/ISPD98 suite, but the cells are placed in a way that guarantees each individual net has minimal wirelength.

Table 4 compares the performance of mPL2 to that of Capo [10] and Dragon [32] on the IBM/ISPD98 benchmarks [2] with all cells of uniform size. On the IBM/ISPD98 benchmarks, mPL2's overall performance is roughly in between those of Capo 8.5 and Dragon. Capo 8.5 demonstrates superior speed, but at some cost in solution quality. Dragon generates placements of superior quality, but at some cost in speed and scalability. mPL2 obtains quality roughly comparable to Dragon's (about 4% longer wirelength) in about 1/5 the run time, with relatively less run time on larger designs. Similarly, the mPL2 run time is about twice as long as the Capo 8.5 run time, but mPL2 obtains superior solutions, about 9% less wirelength than Capo 8.5 on average.

Figures 5, 6, and 7 compare the performance of mPL2 to that of mPL1, Capo, and Dragon on the PEKO benchmarks. Because Dragon's run time on these benchmarks is very long, its graph is displayed separately from the others in order that they remain visually distinguishable. The graphs show that, on PEKO, mPL2's total wirelength is roughly 60% closer to the optimal than that produced by Capo 8.5 or Dragon, and mPL2's run time is about twice as long as Capo's and about 1/10th of Dragon's.

We attribute mPL2's superior performance on the PEKO benchmarks to two factors.

1. The initial cluster hierarchy formed by aggregating cells and clusters according to their hypergraph connectivity. Because all connections in the PEKO benchmarks are local, FC clustering (Section 2) is particularly effective here at identifying good groupings. Although Dragon and Capo both rely on multilevel partitioning to form their hierarchies, they do not make detailed use of the aggregation hierarchy from the multilevel partitioning in their top-down optimization phases.
2. The use of a quadratic wirelength model in the nonlinear-programming engine at the coarsest level. On the IBM/ISPD98 benchmarks, the effect of skipping nonlinear programming at the coarsest level is only a modest increase (0–4%) in the final, total wirelength. On the PEKO benchmarks, however, skipping nonlinear programming at the coarsest level results in 25–61% increased total wirelength at the finest level. The data from these experiments are not shown due to page limits.

Comparing the PEKO results to the IBM/ISPD98 results, we draw two tentative conclusions. First, recursive clustering is an extremely effective means of organizing optimization for circuits that ultimately can be placed using only short, local interconnects. For circuits requiring longer connections, however, more general organizing strategies may be needed. Second, optimal performance may require the incorporation of both quadratic and linear wirelength models into the optimization. Although a quadratic model may overpenalize long wires and thereby lead to inferior solutions for "real" benchmarks, a linear wirelength model may not be aggressive enough in positioning tightly connected cells sufficiently close together.

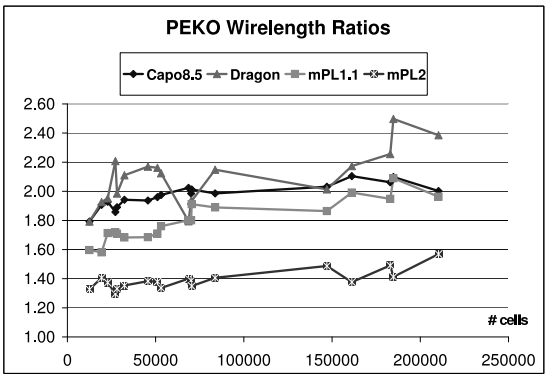

Figure 5: mPL2 vs. mPL1 vs. Capo 8.5 vs. Dragon on PEKO; wirelength-to-optimal-wirelength ratio vs. #cells

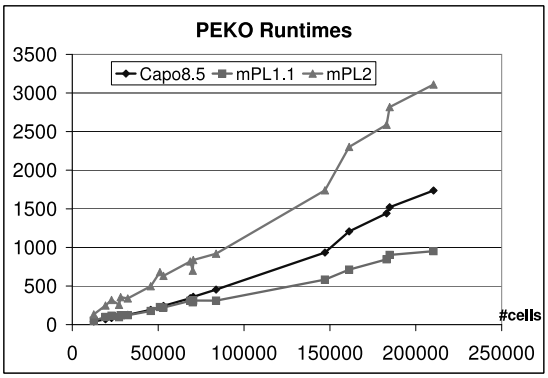

Figure 6: mPL2 vs. mPL1 vs. Capo 8.5 on PEKO; run time vs. #cells

8. FUTURE WORK

To incorporate a timing-driven focus in mPL, we plan to adopt an iterative net-reweighting strategy capable of accurate critical-path estimation [26]. To improve routability, we can incorporate pin counts into the edge weights used for the ripple-move congestion-relief (Section 3). Preliminary experiments using the timing- and congestion-analysis engines of mPG [15] at each level of the cluster hierarchy are encouraging; our work in this direction is ongoing.

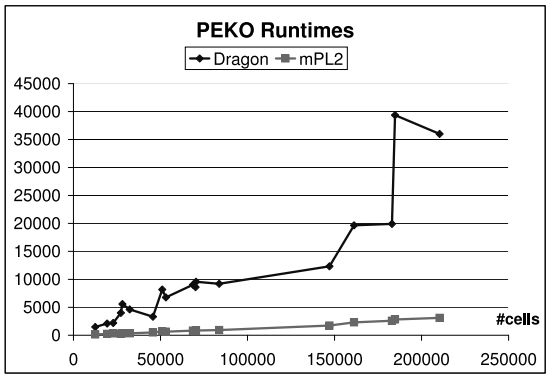

Figure 7: mPL2 vs. Dragon on PEKO: run time

Currently, mPL2 performs significantly worse on circuits with large variations in cell sizes than on circuits with nearly uniform cell sizes. In order to simplify the transition from continuous refinement to nearly overlap-free discrete refinement, variations in cell sizes are only crudely approximated. During slot assignment and ripple-move area-congestion relief, mPL2 divides larger clusters into collections of interconnected smaller ones (Section 6) and then assumes all clusters have the same size. Moreover, all pin locations are placed at cell centers during both global and detailed placement; the cost of this simplification is also much higher in the nonuniform case. We expect further work to bring mPL's performance in the general non-uniform cell-size case in line with its performance in the uniform cell-size case.

Leading contemporary algorithms for hypergraph coarsening tend to produce very dense hypergraphs at coarser levels, because (i) the average number of nodes in a hyperedge and (ii) the ratio of the number of hyperedges to the number of nodes both tend to increase during coarsening. These higher interconnect densities may degrade the performance of the standard optimizations so severely as to render them ineffective at coarser levels. We expect that further work on hypergraph coarsening will lead to improved algorithms for circuit placement.

9. CONCLUSIONS

We believe that multilevel optimization is one of the most promising techniques for large-scale circuit placement. In principle, any of the standard techiques for circuit placement might well be integrated into a multilevel approach, but, in practice, considerable effort is required to adapt them in a way that avoids premature optimization and supports the multilevel flow. The work described here shows how QRS-based relaxation, in combination with AMG-style weighted

		wirelength (WL)					runtime (s)				
circuit	#cells	mPL2	mPL1	Capo8.5	Dragon	Gord-L	mPL2	mPL1	Capo8.5	Dragon	Gord-L
ibm04	27220	1.00	1.18	1.12	0.97	1.05	1.00	0.31	0.53	3.03	1.90
ibm07	45639	1.00	1.14	1.12	0.95	1.05	1.00	0.34	0.60	3.33	3.77
ibm09	53110	1.00	1.14	1.12	1.01	1.04	1.00	0.33	0.67	5.40	4.90
ibm10	68685	1.00	1.11	1.10	0.99	0.99	1.00	0.31	0.55	4.70	6.54
ibm14	147088	1.00	1.16	1.08	0.95	1.04	1.00	0.41	0.57	3.02	8.28
ibm16	182980	1.00	1.07	1.06	0.90	1.00	1.00	0.46	0.54	6.83	11.76
ibm17	184752	1.00	1.18	1.10	0.98	0.98	1.00	0.44	0.43	6.82	10.41
ibm18	210341	1.00	1.11	1.10	0.96	1.03	1.00	0.42	0.43	6.10	13.43
averages		1.00	1.14	1.10	0.96	1.02	1.00	0.38	0.54	4.91	7.62

Table 4: mPL2 vs. mPL1+Domino, Capo 8.5, Dragon, and Gordian-L+Domino on uniform-cell-size IBM/ISPD98 circuits. All values are normalized with respect to mPL2.

interpolation and multiple V-cycle iterations, can be used to improve the solution produced by a multilevel placer.

10. REFERENCES

[1] C. Alpert, J.-H. Huang, and A. Kahng. Multilevel circuit partitioning. In *Proc. 34th IEEE/ACM Design Automation Conf.*, 1997.

[2] C. J. Alpert. The ISPD98 circuit benchmark suite. In *Proc. Intl Symposium on Physical Design*, pages 80–85, 1998.

[3] A. Brandt. Multi-level adaptive solutions to boundary value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[4] A. Brandt. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comp.*, 19:23–56, 1986.

[5] A. Brandt. Multiscale scientific computation: Review 2001. In T. Barth, R. Haimes, and T. Chan, editors, *Multiscale and Multiresolution Methods*. Springer Verlag, 2001.

[6] A. Brandt and D. Ron. *Multigrid Solvers and Multilevel Optimization Strategies*, chapter 1 of *Multilevel Optimization and VLSICAD*. Kluwer Academic Publishers, Boston, 2002.

[7] U. Brenner and A. Rohe. An effective congestion-driven placement framework. In *Proc. International Symposium on Physical Design*, Apr 2002.

[8] W. L. Briggs, S. F. McCormick, and V. E. Henson. *A Multigrid Tutorial*. SIAM, Philadelphia, second edition, 2000.

[9] A. Caldwell, A.B.Kahng, and I. Markov. Improved algorithms for hypergraph partitioning. In *Proc. IEEE/ACM Asia South Pacific Design Automation Conf.*, 2000.

[10] A. Caldwell, A. Kahng, and I. Markov. Can recursive bisection alone produce routable placements? In *Proc. 37th IEEE/ACM Design Automation Conf.*, 2000.

[11] T. Chan, J. Cong, T. Kong, and J. Shinnerl. *Multilevel Circuit Placement*, chapter 4 of *Multilevel Optimization in VLSICAD*. Kluwer Academic Publishers, Boston, 2003.

[12] T. Chan, J. Cong, T. Kong, and J. Shinnerl. Multilevel optimization for large-scale circuit placement. In *Proc. IEEE International Conference on Computer Aided Design*, pages 171–176, San Jose, CA, Nov 2000.

[13] C. Chang, J. Cong, Z. Pan, and X. Yuan. Physical hierarchy generation with routing congestion control. In *Proc. ACM International Symposium on Physical Design*, pages 36–41, San Diego, CA, Apr 2002.

[14] C. Chang, J. Cong, and M. Xie. Optimality and scalability study of existing placement algorithms. In *Asia South Pacific Design Automation Conference*, pages 325–330, Kitakyushu, Japan, Jan 2003.

[15] C.-C. Chang, J. Cong, D. Pan, and X. Yuan. Multilevel global placement with congestion control. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(4):395–409, April 2003.

[16] J. Cong and S. Lim. Performance-driven multiway partitioning. In *Proc. IEEE/ACM Asia South Pacific Design Automation Conf.*, 2000.

[17] J. Cong and J. R. Shinnerl, editors. *Multilevel Optimization in VLSICAD*. Kluwer Academic Publishers, Boston, 2003.

[18] K. Doll, F. Johannes, and K. Antreich. Iterative placement improvement by network flow methods. *IEEE Transactions on Computer-Aided Design*, 13(10), Oct 1994.

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

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

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

[22] G. Karypis. Multilevel algorithms for multi-constraint hypergraph partitioning. Technical Report 99-034, Department of Computer Science, University of Minnesota, Minneapolis, 1999.

[23] G. Karypis. *Multilevel Hypergraph Partitioning*, chapter 3 of *Multilevel Optimization and VLSICAD*. Kluwer Academic Publishers, Boston, 2002.

[24] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi

domain. In *Proc. 34th ACM/IEEE Design Automation Conference*, pages 526–529, 1997.

[25] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. Gordian: Vlsi placement by quadratic programming and slicing optimization. *IEEE Trans. on Computer-Aided Design*, CAD-10:356–365, 1991.

[26] T. Kong. A novel net weighting algorithm for timing-driven placement. In *Proc. IEEE International Conference on Computer Aided Design*, pages 172–176, 2002.

[27] S. McCormick, editor. *Multigrid Methods*. SIAM, Philadelphia, 1987.

[28] Y. Sankar and J. Rose. Trading quality for compile time: Ultra-fast placement for FPGAs. In *FPGA '99, ACM Symp. on FPGAs*, pages 157–166, 1999.

[29] G. Sigl, K. Doll, and F. M. Johannes. Analytical placement: A linear or a quadratic objective function? In *Proc. 28th ACM/IEEE Design Automation Conference*, pages 427–432, 1991.

[30] K. Stueben. A review of algebraic multigrid. Technical Report 69, GMD National Research Center for Information Technology, Bonn, 1999.

[31] W.-J. Sun and C. Sechen. Efficient and effective placement for very large circuits. *IEEE Trans. on Computer-Aided Design*, pages 349–359, Mar 1995.

[32] M. Wang, X. Yang, and M. Sarrafzadeh. Dragon2000: Fast standard-cell placement for large circuits. *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 260–263, Apr 2000.