Mixed-Size Placement with Fixed Macrocells using Grid-Warping

Zhong Xiu
Advanced Micro Devices
Sunnyvale, CA 94088 USA
zhong.xiu@amd.com

Rob A. Rutenbar
Dept. of ECE, Carnegie Mellon University
Pittsburgh, PA 15213 USA
rutenbar@ece.cmu.edu

Abstract

Grid-warping is a placement strategy based on a novel physical analogy: rather than move the gates to optimize their location, it elastically deforms a model of the 2-D chip surface on which the gates have been coarsely placed via a standard quadratic solve. Although the original warping idea works well for cell-based placement, it works poorly for mixed-size placements with large, fixed macrocells. The new problem is how to avoid elastically deforming gates into illegal overlaps with these background objects. We develop a new lightweight mechanism called "geometric hashing" which relocates gates to avoid these overlaps, but is efficient enough to embed directly in the nonlinear warping optimization. Results from a new placer (WARP3) running on the ISPD 2005 benchmark suite show both good quality and scalability.

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

General Terms

Algorithms, Design

Keywords

Algorithms, Placement, Mixed-Size Placement

1. Introduction

Placement remains an area of surprisingly active investigation. Several different approaches are still actively competing for primacy in speed, wirelength, scalability, timing closure, etc. We note that partition-based approaches (e.g., Capo [25], Feng Shui [4]), annealing approaches (e.g., Dragon [26]), quadratic approaches (e.g., BonnPlace [5]), analytical approaches (e.g., mPL [9], APlace [19], Kraftwerk [14][32] and NTUPlace [11][33]), and the subject of this paper, grid warping [30],[31], all continue to evolve and compete.

Grid-warping, introduced in [30], is a placement strategy based on a novel physical analogy: rather than move the gates to optimize their location, it elastically deforms a model of the 2-D chip surface on which the gates have been coarsely placed via a standard quadratic solve. The strategy transforms the high-dimensional problem of solving for the locations of every placeable object, into a much smaller problem of determining an appropriate elastic deformation to maximally improve a given initial placement. In practice, the technique resembles both the quadratic and analytical approaches. Like the quadratic technique, warping relies on a hierarchy of quadratic placement (QP) solves to recursively complete the layout. But like the analytical techniques, warping relies on a highly nonlinear optimization as a core mechanism to evolve the initial QP solution to a stage where an optimal recursive decomposition can be made.

A timing-driven component was added to a warping scheme in [31]. The problem we address in this paper is how we extend a grid-warping formulation to the *mixed-size* placement case. In most large ASIC and SOC-style designs, we see a range of component sizes: a moderate number of very large macrocells (for memories, hard-IP blocks such as processors and DSPs, etc.), a larger number of medium-sized cells which still snap into the standard cell row structure, but may span several cell rows, and a very large number of individual standard cells.

Addressing the mixed-size case proves to be a challenge for a warping placer. The reason is that warping is extremely adept at preserving the localities of the initial quadratic placement starting point. This is good for many small gates; this is not good when we inadvertently sweep individual gates on top of large macros. As a starting point, we address the case where the large macrocells are fixed during preplacement on the chip surface. We show how to extend the warping formulation to accommodate an arbitrary set of fixed macrocells.

The rest of the paper is organized as follows: Section 2 reviews related work and the basic grid-warping formulation. Section 3 describes a set of extensions to the core warping formulation that both improve wirelength and accommodate macrocells. Section 4 shows experimental results for a new placer -- WARP3 -- and comparisons against other published placers. Finally, Section 5 offers concluding remarks.

2. Background

2.1 Previous Work on Mixed-Size Placement

Mixed-size placement has recently drawn considerable attention, especially as larger designs integrate a larger set of memory and IP components with several million logic gates. The problem offers different challenges to different placer and legalizer strategies. We briefly review the landscape here.

Partition-based techniques have always been relatively accommodating of mixed-sized problems since they defer until the end of place-

ISPD'07, March 18-21, 2007, Austin, Texas, USA.

ment the physical assignment of objects to locations. The Capo partitioner has been used in conjunction with the Parquet floorplanner in [1], [2], [3] to place arbitrary macro blocks and standard cells without overlap. A novel geometric feature of the approach is that macros are *shredded* into small pieces connected by pseudo-wires, and this is used by the global placer to obtain an initial placement. In the second stage, the standard cells are merged into soft blocks, and a floorplanner generates valid locations of macros and soft blocks. In the final stage, the macro blocks are fixed, and cells in the soft blocks go through a detailed placement. Another partitioning placement tool, Feng Shui [21], uses recursive-bisection with iterative deletion, iterative repartitioning, relaxed rows not aligned with standard cell rows ("fractional cut"), and a simple Tetris-style approach to legalization.

The force-directed methods, e.g., FastPlace [28], Kraftwerk [14], have been extended to work in the mixed-size case, exploiting the fact that they explicitly formulate countervailing forces to push small blocks off large blocks. The quadratic/recursive methods, such as BonnPlace [5], also handle the mixed-size case. BonnPlace first fixes the position of the macro cells, then places all the remaining standard cells (QP) and adjusts for capacity constraints using a novel transportation algorithm designed to avoid overlap while simultaneously minimizing overall wirelengths.

The smoothed analytical methods, e.g., APlace ([17], [18], [19]), mPL ([7], [8], [9], [10]), work extremely well here and seem to produce the best quality overall, since they explicitly formulate cell overlaps and drive both wirelength and rough placement legality simultaneously. The downside of these methods is their significant computational expense. To avoid the scalability issues with purely flat approaches, multilevel approaches with clustering/de-clustering techniques have been proposed to reduce runtime, e.g., [19].

2.2 Placement by Grid-Warping

The basic grid-warping formulation has been described in previous papers ([30] for standard cells, [31] for timing optimization). The goal in this paper is to develop a mixed-size capability in a warping formulation. Without this, grid-warping cannot be regarded as a practical, competitive placer strategy. Before going into details about our mixed-size placement formulation, we give a brief review of grid warping in this section.

The underlying idea of grid-warping is simple: rather than move the gates to optimize their location, one elastically deforms a model of the 2-D chip surface on which the gates have been quickly and coarsely placed. Put simply: warping moves the grid, not the gates. Rather than move each point individually, one "*stretches*" the underlying sheet until the points arrange themselves in a more optimal way.

Grid warping starts with a conventional quadratic placement (QP), in which each gate to be placed is represented as a dimensionless point

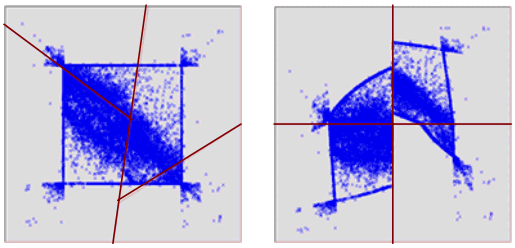

FIGURE 1. Warping applies three slicing-style cuts to the quadratic placement (left) then "un-deforms" the four resulting quadrilaterals back to a standard 2x2 quadrisection to move the gates (right).

connected to a set of appropriately weighted 2-point wires. Overall squared Euclidean wirelength is minimized. This QP serves as the initial placement of the "spots on the sheet" for the subsequent warping improvement step.

Then, conceptually, one defines a set of control points on the placement surface; warping elastically moves these control points to approximate some continuum deformation of the grid. As the grid deforms, the elastic placement sheet deforms, and gates move. A nonlinear optimizer drives this deformation process. This optimization is low-dimensional because relatively few features are needed to control the deformation. Optimization minimizes a cost function that is a weighted combination of exact half-perimeter wirelength and capacity penalty. To date, the best results have been obtained with a warping scheme that applies a set of slicing-style cuts, at arbitrary locations/angles, to the QP. For example, three cuts, (see Figure 1) partition the QP into a set of four arbitrary quadrilaterals. Gates move when each quadrilateral is "un-deformed" and elastically stretched to a standard balanced 2x2 quadrisection. The optimizer picks the locations of these cuts, typically visiting several thousand trial solutions before finding one that best optimizes wirelength and area balance.

Grid-warping still relies on recursive decomposition. To confine the cells inside each decomposed region, a global quadratic solve runs at the beginning of each recursive layer, following the style of [29], but also enforcing a center of gravity constraint in each subregion. This placement serves as another starting point for warping in each subregion. Ideas from mincut partitioning are also used to disambiguate gates placed very close to the cutlines. A local improvement step (inspired by [29]), called *re-warping*, is used at the end of each level to improve the wirelength. Two final pieces to mention are steps at the beginning and end of warping. A *pre-warping* step, which geometrically "conditions" the problem for an easier solution, spreads the gates more uniformly before each warping commences. And, like all analytical placers, warping requires a separate final legalization step. The implementations in [30] and [31] used DOMINO [13]. Figure 2 shows key steps in a grid-warp placement for the ibm06 benchmark from [30] using a 15-cut 4x4 warping.

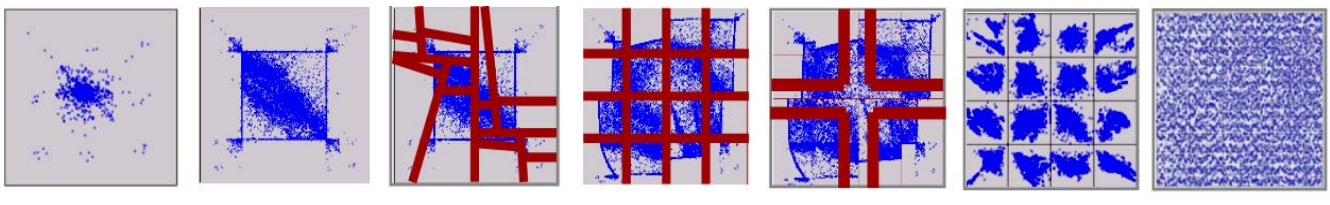

FIGURE 2. Progress through grid-warping flow for the ibm06 benchmark, using an 8x8 pre-warp grid, and a 4x4 unit slicing grid for warping

3. Mixed-Sized Placement by Grid Warping

3.1 Mixed-Size Model

For any method to handle mixed-size placement, the key issue is how to handle the big macro blocks. In this paper, we assume the positions of the large macro blocks are fixed prior to placement, and we extend warping to place all the other relatively small blocks and standard cells around these fixed background objects.

We classify the placement instances into two categories. If the height is over ten times larger than the height of the standard cells, we consider it to be a macro that requires a fixed pre-placement. All other cells are assumed to be movable, and will be snapped into one or more cells rows at the end of legalization. In practical usage, these macro placements usually come from designers doing some early floorplanning. For example, the bigblue3 benchmark from the ISPD 2005 benchmark suite has 1.09M instances, of which 1293 are fixed (pre-placed) macros, and 2485 are smaller (2-~10 cell rows in height) and will be placed with the individual gate level instances.

3.2 Starting Formulation

We start with the formulation of [31], with one useful improvement. We evolve the net model used during QP to improve both runtime and overall wirelength. As with all quadratic-style formulations, the net model consists of a vertical and a horizontal component, which can be computed independently. Here, we divide the quadratic placement (QP) into three categories: (a) *top-level* QP, which is used as the first QP for the whole-chip initial placement; (b) *local-improvement* QP, used in each of the subsequent local window improvement (re-warping, [31]) steps; and (c) *lower-level* QP, which is used in the second and later decomposition layers as the starting placement. The last category is more complicated than the first two categories, since it must confine the movement of all the gates -- placing the gates in the sub-region that they are assigned.

As suggested in [31], for the top-level and local-improvement QPs, we use the very efficient hybrid net model from FastPlace [27]. However, in contrast to both [27] and [31], we now use the more-efficient star model for *all* multi-pin nets, i.e., for *all* nets with three or more pins. For an n-pin net, the star model introduces a new variable, and gives each resulting connection a weight of: $n/(n-1)$. 2-pin nets still use the (now trivial) clique model, with a weight of 1.

For the lower-level QP in the second layer and later layers, we adapt the net-split technique from Vygen [29], as well as some ideas from BonnPlace [5]. Let us assume we are given a set S of intervals defined by the vertical cut lines. For each net N and each interval $I = [l_i, r_i] \in S$, let Ns denote the set of pins in N whose x-coordinates have to be placed within S. Let $L(S,N)$ be the number of pins of N left to l_i and $R(S,N)$ right hand to r_i. There are several cases:

1. If there are no movable cells in the interval I, we do not process it.
2. If $L(S,N)+R(S,N)+Ns<3$, this means there are less than 3 pins for this net N. In this case, the clique model is used no matter if there are 1 or 2 movable cells in interval I.
3. If $n=L(S,N)+R(S,N)+Ns>2$ and $Ns=1$, which means only one movable cell is in I, then this cell should be the center-of-gravity of all other pins. In this case, all other pins are fixed, they are either propagated to the left vertical cut or the right vertical cut of interval I. The star model is used, and the weight is set to $n/n-1$ (Figure 3 (a)).
4. If $n=L(S,N)+R(S,N)+Ns>2$ and $Ns>1$, which means there are more than one movable cell in I, then an extra variable is introduced and the star model is used. The new variable is connected to all the other pins, each has a weight of $n/n-1$ (Figure 3 (b)).

Basically speaking, we use terminal propagation techniques to propagate pins outside the interval to the left or right cut line first. Then both the movable cells and the fixed pins are treated the same. If the total number of pins is less than 3, we use the trivial clique model with connection weight 1; otherwise we switch to a star model with a weight of $n/n-1$ (n is the number of total pins). To explain why this is worth considering, we note that in the 2nd and later levels of global QPs, some nets are dominated by fixed pads and might have no movable cell or only one movable cell: we do not want to introduce new variables to deteriorate runtime. This actually makes our approach different with the traditional clique/star model. So the option of clique and star models for 3-pin nets can lead to very different results. Finally, we also abandon the center-of-gravity constraint from [29] for low-utilization regions since it does no good in this case.

In practice, this evolved net model shows surprising improvements in placement quality with very modest runtime cost. An obvious question is whether switching from clique to star for the 3-pin nets (in contrast to [27]) is worthwhile. Results in Table 1 and Table 2, show that star models for 3-pin nets produce significantly better placements at modestly increased runtime. So we use star models for all but 2-pin nets.

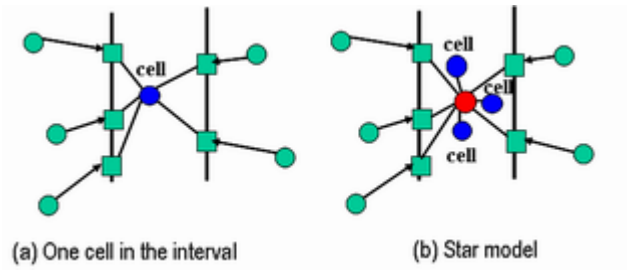

FIGURE 3. New Net Model: (a) only one cell is placed in the interval, it should be the center of gravity; (b) an extra cell is introduced if the number of pins is greater than 2 and there are more than 2 cells in the interval.

Design	star models for 3-pin nets		clique models for 3-pin nets	
	Wirelength	Time (s)	Wirelength	Time (s)
adaptec2	0.79728	41.99	1.42080	35.91
adaptec4	1.93699	110.70	2.40809	99.63
bigblue2	2.30286	87.20	2.53548	80.21
bigblue3	3.12168	234.14	4.06507	207.20
Ratio	**1.00**	**1.00**	**+36.0%**	**-11.0%**

TABLE 1. Placement results of the 2nd level global QP

Design	star models for 3-pin nets		clique models for 3-pin nets	
	Wirelength	Time (s)	Wirelength	Time (s)
adaptec2	0.78113	37.59	1.40459	34.37
adaptec4	2.02946	101.54	2.58556	96.24
bigblue2	2.00753	84.26	2.20845	71.40
bigblue3	2.93728	216.09	4.13523	187.73
Ratio	**1.00**	**1.00**	**+39.5%**	**-10.6%**

TABLE 2. Placement results of the 3rd level global QP

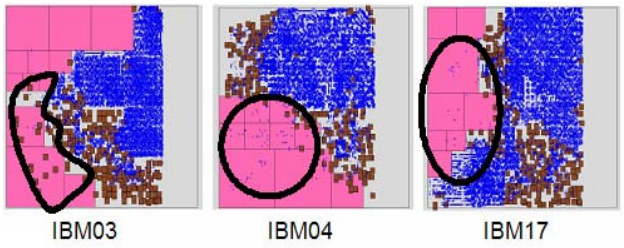

FIGURE 4. Pre-legalization results from ignoring the problem of warping small cells on top of fixed macrocells. Large-scale overlaps exist (circles), which backend legalization does poorly to resolve.

3.3 Handling Mixed-Size Case with Legalization Only

Before we embark on a set of potentially deep changes to the core warping formulation, it is worth asking if the *simplest* possible solution is a workable one. That is: can we *ignore* the problem of gate-level instances being inadvertently warped on top of fixed macrocells, and just resolve the overlaps at the end of grid warping, by letting the backend legalizer deal with these violations?

The answer, unfortunately, is no. Figure 4 shows some examples of the geometry of the problem, immediately after warping completes. We see relatively many small cells marooned in the middle of large macros. Current legalizers are designed to resolve modest amounts of illegal overlap; this much illegality tends to confound the legalizers, which produce as a result extremely poor final wirelengths.

3.4 Handling Mixed-Size Case with Geometric Hashing

The warping concept is exceptionally adept at keeping related gates close to each other during placement; this is a direct consequence of the "elastic sheet" deformation model. Unfortunately, this also means that it is difficult for force gates away from large, fixed background macros during placement evolution. Other placers based on QP with decomposition have formulated explicit repair steps that (i) minimally perturb the QP, while (ii) avoiding the macrocells. Vygen [29] and BonnPlace [5] use LP and network-flow ideas, respectively, for this purpose.

Our problem is different, and illustrated in Figure 5. Because warping is itself a nonlinear optimization, there can be several thousand unique warping solutions attempted (Figure 5(b)) before an optimal final warping is determined. Each trial warping solution may inadvertently deposit thousands of cells in randomly illegal overlaps. We do not have *one* illegal overlap scenario to resolve, as [29] and [5], per layer of the recursion hierarchy. We have thousands to resolve. We need a very simple, lightweight mechanism which can be inserted *inside* the warping optimization loop. In particular, we want wirelength calculations to at least roughly reflect the fact that depositing gates on top of macrocells likely has a wirelength impact -- the extra length needed to move those gates off the cell.

An extremely simple idea works well to resolve this problem. We focus on a "partial repair" strategy that greedily relocates gates and small blocks with problematic overlaps. We call this *geometric hashing*, for simplicity. Before warping, we impose a fine grid on the current QP solution, and for those grid cells that are partially or fully occluded by fixed macrocells, we statically compute and store the closest unobstructed macrocell boundary (Figure 5(d)). The idea is that, if any gate lands on top of a macrocell in this grid, we will simply sweep it over to the nearest macrocell boundary that is adjacent

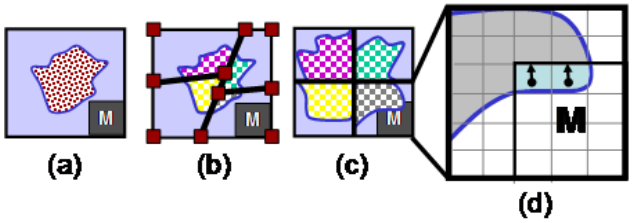

FIGURE 5. Geometric Hashing: (a) QP places a set of movable cells in a design with one macrocell "M". (b) Warping iteratively searches for cut locations that optimally deform the QP to (c) a standard quad-cut. Each deformation can overlap the fixed macrocell. (d) Overlapping cells are greedily relocated -- *hashed* -- to the nearest free boundary, as defined by a statically computed fine grid of minimal-distance perturbations.

ALGORITHM 1: Mixed-Size Placement Algorithm -- WARP3

1. Run the quadratic placement algorithm with hybrid net model
2. Run the checker for the utilization and placement
3. Do pre-warping or not according to the result of 2
4. Run the nonlinear grid-warping loop with new cost function and "geometric hashing"
5. Paritioning improvement with relaxed balance
6. Repeat
   - 6.1: The global quadratic placement with net-split techniques and hybrid net models
   - 6.2: In each sub-region, repeat step 2, 3 and 4, with cells outside this sub-region propagated to the boundary of this sub-region
   - 6.3: If not the final layer, run re-warping and geometric hashing

   Until each grid cell has no more than 30 cells

---

to free space. Note that we make no attempt to optimize density or wirelength in this local solution, just legality. We refer to this as geometric *hashing* because we repair overlap violations by simply hashing into this simple 2-D grid structure, looking up the nearest free boundary edge, and relocating the gate appropriately. Since this is simple to compute, we perform geometric hashing for *every* trial warping solution, i.e., we compute the wirelength and capacity penalty terms of the cost function after all violating gates have been hashed to legality.

In addition, before we descend into each newly warped region and start placing it in finer detail, we do a single global, top-level repair step to relocate only the difficult overlaps to the nearest free space. In other words, we *hash* overlapped cells off. At the end of each recursion level, we check if each movable cell is on top of any fixed macro cell. If so, we again hash it to the nearest macrocell boundary that has enough space for the cell.

This simple but useful heuristic has very low complexity but works very well for overlap removal. Experiments show that with geometric hashing, the average wirelength before legalization is increased around 1% but the final legalized wirelength is decreased about 4-5%

3.5 Better Consideration of Capacity

With geometric hashing in place, placing the remaining smaller blocks with grid warping is mainly an exercise in bookkeeping: we need to account for cell sizes accurately in each placement step after the initial quadratic point placement. Thus, we warp, and partition-

improve, and re-warp, while ensuring that any sub-region is not filled over its capacity, and we carefully account for the areas used by both movable cells and fixed macros. As always, these sorts of low-level engineering changes touch several other parts of the placer. We summarize these impacts here:

- **Pre-warping**: Pre-warping is just a pre-conditioning process for us, which aims to uniformly distribute very dense clusters. [30] used pre-warping to deal with the high-utilization ISPD02 benchmark set. However, for relatively low-utilization designs such as the large ISPD05 netlists, there can be a deleterious impact on quality. For example, suppose the QP places all the cells in one of the four quadrants of the current decomposition, and the total cell area does not exceed the area of this quadrant. Pre-warping, by design, would still stretch the entire QP across all four quadrants uniformly, making the wirelength worse and relying on the later warping and re-warping engines to fix this stretch. To avoid this, a new utilization checker runs before the pre-warping stage and checks the utilization of the area: if it is below a threshold and the original QP placement does not violate the capacity restriction for any of the four quadrants, pre-warping will not be conducted.

- **Cost function**: In [30], a two-sided "bath-tub" style cost function was used to assure cells uniformly distributed to sub-regions. Regions that were over-filled, and regions that were under-filled, were penalized equally. This is efficient when dealing with high-utilization netlists like the ISPD02 designs, but causes very bad results for low-utilization designs such as ISPD05. Hence, we replaced this with a single-sided cost function, i.e., under-filled regions receive no capacity penalty in the overall warping cost function, but over-filled regions are penalized as in [30].

- **Partitioning improvement**: [30] used hMetis [20] in the partitioning improvement step to repartition the cells placed near the cut lines. We still use hMetis here to further improve the wirelength, but with more care not to violate possibly asymmetric capacity constraints on opposite sides of the cutline(s). We carefully balance the capacity here, so that when hMetis is used, the total area of all cells in each region does not exceed the capacity of that region. Of course, fixed macros are also taken into consideration when we compute the capacity.

- **Re-warping**: In the re-warping stage [31], cells inside a small 2x2 window are thrown together and placed, warped again to achieve some improvements. All off the above mentioned modifications (to prewarping, to cost function, to partitioning improvement) are all applied here. Re-warping can still greatly change the placement inside this window, and accept better placements. In practice this step gains us a lot quality, especially for low-utilization designs.

The overall flow for the mixed-size case, with all these extensions and with the key geometric hashing steps, appears in Algorithm 1.

3.6 Legalization

Legalization is widely understood to be a more challenging problem in the mixed-size case [12]. This observation conforms to our experience with the warping-based strategy as well. As a result, we abandoned the Domino-based flow of [31] and adopted a more successful three-step flow, using ideas from Feng Shui 5.1 [4] and the cell-swap method of FastPlace [24]. The flow consists of three steps:

1. **Global legalization:** We use Feng Shui 5.1 [4] for first-pass legalization.
2. **Local repair:** There may still be overlaps after the first step. If so, or if some cells are outside the chip boundary, we use a simple greedy scheme to repair. We move violating cells into the nearest row with the least displacement, and adjust other cells in those rows as needed.
3. **Wirelength optimization:** We use the swap method from FastPlace [24] as a last-phase to reclaim any wirelength lost in the first two geometric legalization steps.

Our warping placements are generally not much disturbed by this new legalization scheme (step 1 and step 2), i.e., we generally do not need to change the placement much to make it legal. As we will see later from the experimental results, the legalization step only increases the wirelength by 1~2% typically.

4. Experimental Results

The ideas of the previous section have been implemented in a new placer called WARP3. We first use the circuits from the ISPD02 mixed-size placement benchmarks [16] as our testcases. This suite of benchmarks range from 12K to about 200K cells. The total area of macro blocks and standard cells occupy 80% of the chip area, which represents high-utilization benchmarks. Table 3 shows the results and

Design	Feng Shui 5.1		BonnPlace		WARP3	
	Wirelen	CPU (s)	Wirelen	CPU (s)	Wirelen	CPU (s)
ibm01	0.242	134	0.226	360	0.232	203
ibm02	0.501	238	0.493	600	0.496	401
ibm03	0.826	271	0.701	660	0.711	408
ibm04	0.873	306	0.823	780	0.789	453
ibm05	0.984	322	1.002	780	1.018	462
ibm06	0.691	409	0.655	780	0.642	746
ibm07	1.097	558	1.041	1200	1.032	1133
ibm08	1.369	642	1.268	1800	1.289	1434
ibm09	1.363	646	1.327	2100	1.310	1472
ibm10	3.368	982	3.292	1920	3.120	2259
ibm11	1.973	872	1.915	2220	1.954	1911
ibm12	3.514	1023	3.190	2880	3.445	2311
ibm13	2.380	1130	2.431	2880	2.394	2501
ibm14	3.826	2082	3.782	3600	3.853	7549
ibm15	5.037	2672	4.931	5100	5.041	9244
ibm16	5.835	3239	5.788	6600	5.891	10537
ibm17	7.017	3238	6.665	11940	6.762	11574
ibm18	4.499	3350	4.574	5340	4.418	13115
Ratio	**1.037**	**0.47**	**0.997**	**--**	**1.000**	**1.00**

TABLE 3. Placement results comparing Feng Shui 5.1 [4], BonnPlace [5] & Warp3

Design	WARP3			APlace
	Global Wirelength	Legalized Wirelength	Final Wirelength	Wirelength
adaptec2	1.0273	1.0447	0.9720	0.8731
adaptec4	2.1258	2.1892	2.0455	1.8731
bigblue1	1.0639	1.0659	1.0181	0.9464
bigblue2	1.7764	1.8526	1.6950	1.4382
bigblue3	3.6878	3.6959	3.5783	3.5789
bigblue4	8.9466	9.1896	8.5981	8.3321
Ratio	**1.043**	**1.063**	**1.000**	**0.927**

TABLE 4. Placement results comparing Warp3 with APlace [19]

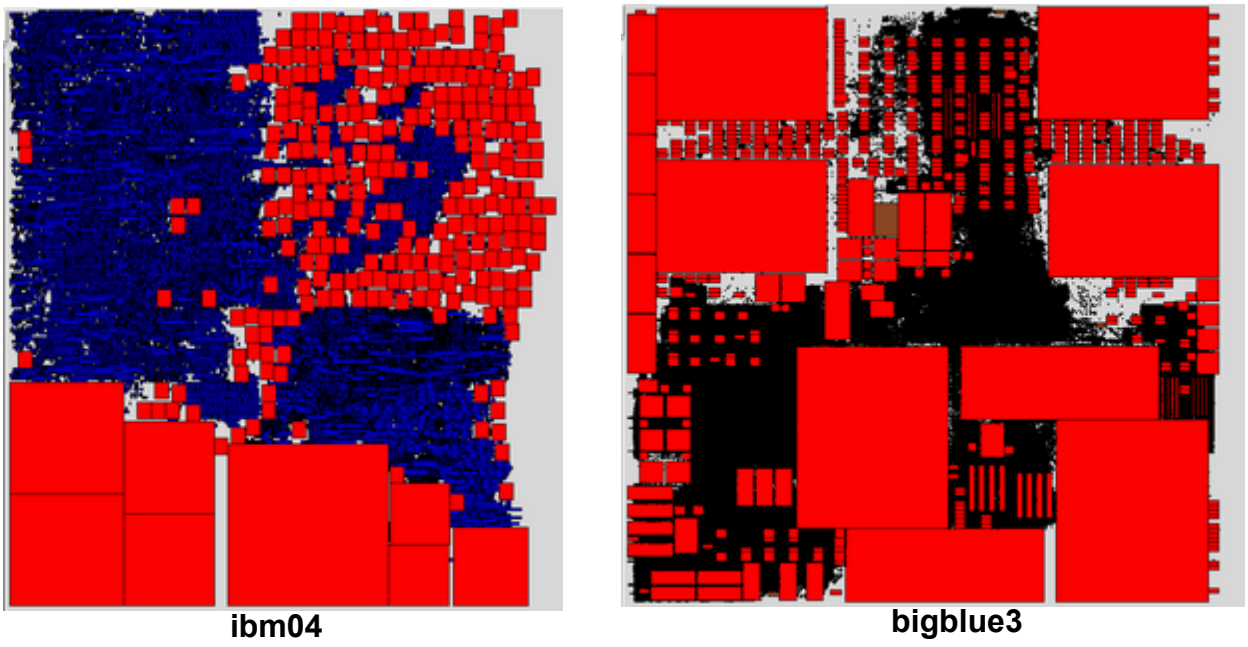

FIGURE 6. Final Warp3 placements of *ibm04* (left), and *bigblue3* (right).

compares WARP3 with Feng Shui 5.1 [4], and BonnPlace [5], which run all the same benchmarks. We run WARP3 and Feng Shui 5.1 both on a 2.0 GHz CPU LINUX machine, CPU times for BonnPlace are not precisely comparable since they represent not only a different processor (IBM P650 at 1.45GHz) but an explicitly parallelized implementation running on a 4-processor server.

From the results, WARP3 has 3.7% shorter wirelength than Feng Shui 5.1 and is essentially identical with BonnPlace. For CPU time, WARP3 is about 2 times slower than Feng Shui 5.1 and, despite problems of comparison with a 4-processor parallelized BonnPlace, seems quite competitive on time as well. We think this speaks well of the simplicity of the geometric hashing scheme.

We also ran WARP3 on the recently released ISPD 2005 benchmarks [22]. This suite of benchmarks range from about 255K cells to over 2M cells. Generally these designs have a lower utilization, and each circuit represents a special testcase, e.g., adaptec2 has a large block in the center at the chip area, bigblue3 has a few thousand movable macro cells. We compared our results with APlace [18], which by far has the best results, and we show results in Table 4. WARP3 is about 7~8% worse than APlace on these benchmarks, which we think is reasonably good. For the runtime, the total runtime of WARP3 on these six benchmarks is 41.75 hours on a 2.8GHz CPU LINUX machine. (The flat version of APlace [18] takes more CPU than this for the largest individual benchmarks in [22]. However, the clustered versions from [19] are much faster.) We do note that some of wirelength results (e.g., bigblue3) are slightly better than APlace, though we must also note that we have yet to implement any routability density optimizations. Overall we regard these results as a very satisfactory first extension of the grid warping platform to the important mixed-size case.

Finally, Figure 6 shows final layouts for two benchmarks, ibm04 and bigblue3, allowing one to see how small gates and medium macros have been successfully placed around the large fixed macro blocks.

5. Conclusions

The challenge for grid-warping the mixed-size case, with fixed macrocells, is how not to warp gates into deep overlap violations with these background objects, since these cannot be easily repaired in final legalization. We presented a set of mechanisms, most notably the geometric hashing idea, to extend a warping placer to handle fixed macrocells. Experimental results show our algorithm is competitive.

Our future work includes ongoing improvements to placer runtime and quality, mechanisms for handling routing congestion, and also exploration of "hybrid" strategies that retain the advantages of the low-dimensional warping, but augment them with the quality-of-results advantages of "smoothed" analytical methods.

Acknowledgments

We thank Paul Villarrubia from IBM and Lou Scheffer from Cadence for may insightful discussions on our mixed-size placement algorithms. This work was supported by the Pittsburgh Digital Greenhouse and The Technology Collaborative.

References

[1] S. N. Adya and I. L. Markov, "Consistent placement of macro-blocks using floorplanning and standard-cell placement," *Proc. ACM ISPD*, April 2002.

[2] S. N. Adya, I. L. Markov and P. Villarrubia, "On whitespace and stability in mixed-size placement and physical synthesis," *Proc. ACM/IEEE ICCAD*, Nov. 2003.

[3] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa and I. L. Markov, "Unification of partitioning, placement and floorplanning," *Proc. ACM/IEEE ICCAD*, Nov. 2004.

[4] A. Agnihotri, S. Ono and P. Madden, "Recursive bisection placement: Feng Shui 5.0 implementation details," *Proc. ACM ISPD*, April 2005.

[5] U. Brenner and M. Struzyna, "Faster and better global placement by a new transportation algorithm," *Proc. ACM/IEEE DAC*, June 2005.

[6] A. Caldwell, A. Kahng, I. Markov, "Can recursive bisection alone produce routable placements?" *Proc. ACM/IEEE DAC*, June 2000.

[7] T. F. Chan, J. Cong, T. Kong, J. R. Shinner, "Multilevel optimization for large-scale circuit placement.," *Proc. ACM/IEEE ICCAD*, Nov. 2000.

[8] T. F. Chan, J. Cong, T. Kong, J. R. Shinner, K. Sze, "An enhanced multilevel algorithm for circuit placement," *Proc. ACM/IEEE ICCAD*, Nov. 2003

[9] T. F. Chan, J. Cong, M. Romesis, J. R. Shinnerl, K. Sze and M. Xie, "mPL6: A robust multilevel mixed-size placement engine," *Proc. ACM ISPD*, April, 2005.
[10] C.-C. Chang, J. Cong and X. Yuan, "Multilevel generalized force-directed method for circuit placement," *Proc. ACM ASPDAC*, Jan. 2003.
[11] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang and Y.-W. Chang, "NTUplace: A ratio partitioning based placement algorithm for large-scale mixed-size designs," *Proc. ACM ISPD*, April, 2005.
[12] J. Cong, M. Romesis and J. R. Shinnerl, "Robust mixed-size placement under tight white-space constraints," *Proc. ACM/IEEE ICCAD*, Nov. 2005.
[13] K. Doll, F. Johannes and K. Antreich, "Iterative placement improvement by network flow methods," *IEEE Tran. on CAD of Integrated Circuits and Systems*, vol. 13(10), 1994.
[14] H. Eisenmann, F. M. Johannes, "Generic global placement and floorplanning," *Proc. ACM/IEEE DAC*, June 1998.
[15] B. Hu, Y. Zeng and M. Marek-Sadowsak, "mFAR: Fixed-points-addition-based VLSI placement algorithm," *Proc. ACM ISPD*, April 2005.
[16] ISPD02 benchmarks: http://vlsicad.eecs.umich.edu/BK/ISPD02bench/
[17] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytical placer," *Proc. ACM ISPD*, April 2004.
[18] A. B. Kahng and Q. Wang, "An analytic placer for mixed-size placement and timing-driven placement," *Proc. ACM/IEEE ICCAD*, Nov. 2004.
[19] A. B. Kahng, S. Reda and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," *Proc. ACM/IEEE ICCAD*, Nov. 2005
[20] G. Karypis, R. Agarwal, V. Kumar, S. Shekhar, "Multilevel hypergraph partitioning: Applications in VLSI design," *Proc. ACM/IEEE DAC*, June 1997.
[21] 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," *Proc. ACM ISPD*, April 2004.
[22] G.-J. Nam, C. Alpert, P. Villarrubia, B. Winter and M. Yildiz, "The ISPD2005 placement contest and benchmark suite," *Proc. ACM ISPD*, April 2005.
[23] B. Obermeier, H. Ranke and F. M. Johannes, "Kraftwerk - A versatile placement approach," *Proc. ACM ISPD*, April 2005.
[24] M. Pan, N. Viswanathan and C. Chu, "An efficient and effective detailed placement algorithm," *Proc. ACM/IEEE ICCAD*, Nov. 2005.
[25] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu and I. L. Markov, "Capo: Robust and scalable open-source min-cut floorplacer," *Proc. ACM ISPD*, April 2005.
[26] T. Taghavi, X. Yang, B. K. Choi, M. Wang and M. Sarrafzadeh, "Dragon2005: Large-scale mixed-size placement tool," *Proc. ACM ISPD*, April 2001.
[27] N. Viswanathan and C. Chu, "FastPlace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model," *Proc. ACM ISPD*, April 2004.
[28] N. Viswanathan and C. Chu, "FastPlace: An analytical placer for mixed-mode designs," *Proc. ACM ISPD*, April 2005.
[29] J. Vygen, "Algorithms for large-scale flat placement," *Proc ACM/IEEE DAC*, June 1998.
[30] Z. Xiu, J. Ma, S. Fowler and R. Rutenbar, "Large-scale placement by grid warping," *Proc. ACM/IEEE DAC*, June 2004.
[31] Z. Xiu and R. Rutenbar, "Timing-driven placement by grid warping," *Proc. ACM/IEEE DAC*, June 2005.
[32] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen and Y.-W. Chen, "A high-quality mixed-size analytical placer considering preplaced blocks and density constraints", *Proc. ACM/IEEE ICCAD*, Nov. 2006.
[33] P. Spindler and F. M. Johannes, "Fast and robust quadratic placement combined with an exact linear net model", *Proc. ACM/IEEE ICCAD*, Nov. 2006.