A SimPLR Method for Routability-driven Placement

Myung-Chul Kim[*], Jin Hu[*], Dong-Jin Lee and Igor L. Markov
University of Michigan, Department of EECS, 2260 Hayward St, Ann Arbor, MI 48109-2121
{mckima, jinhu, ejdjsy, imarkov}@eecs.umich.edu

ABSTRACT

Highly-optimized placements may lead to irreparable routing congestion due to inadequate models of modern interconnect stacks and the impact of partial routing obstacles. Additional challenges in routability-driven placement include scalability to large netlists and limiting the complexity of software integration. Addressing these challenges, we develop *lookahead routing* to give the placer advance, firsthand knowledge of trouble spots, not distorted by crude congestion models. We also extend global placement to (i) spread cells apart in congested areas, and (ii) move cells together in less-congested areas to ensure short, routable interconnects and moderate runtime. While previous work adds isolated steps to global placement, our SIMultaneous PLace-and-Route tool SimPLR integrates a layer- and via-aware global router into a leading-edge, force-directed placer. The complexity of integration is mitigated by careful design of simple yet effective optimizations. On the ISPD 2011 Contest Benchmark Suite, with the official evaluation protocol, SimPLR outperforms every contestant on every benchmark.

1. INTRODUCTION

In earlier technology generations, placement and routing algorithms were designed and implemented in separate software tools, even when the user interface exposed a single optimization to chip designers. After logic synthesis, a placer generates row- and site-aligned, non-overlapping locations for cells with small interconnect length (HPWL)[2]. A global router then routes all signal nets, with small total wirelength, subject to track capacity constraints. Yet, common placement metrics no longer capture key aspects of solution quality at new technology nodes [2,27]. Wirelength-optimized placements often lead to routing failures when the placer is not aware of actual routes [11]. Prior work incorporates routing congestion analysis into global placement (see Section 2), but lacks in several aspects. **First**, simplified congestion models do not capture phenomena salient to modern layouts, e.g., the impact of non-uniform interconnect stacks and partial routing obstacles on congestion. **Second**, the placement techniques that best control whitespace allocation in response to congestion (min-cut and annealing-based) are no longer competitive on the largest layouts. **Third**, incremental post-placement optimization *alone* is often insufficient as it cannot change the structure of global placement.

Challenges in congestion estimation [2]. A successful estimator must account for up to twelve metal layers with wire widths and spacings that differ by up to 20×. Blockages and per-layer routing rules must be modeled as well. Other constraints include via spacing rules and limits on intra-gcell routing congestion. After the 2007/2008 ISPD contests, academic routers NTHU-Route 2.0 [8], NTUgr [13], FastRoute 4.0 [39], BFG-R [14] started to account for these issues. More recent routers — PGRIP [38], PGR (SGR) [23], GLADE [9,20] — have improved solution quality and runtime, and account for different layer directives.

Routability-driven placement can pursue several different optimization objectives, such as ensuring 100% routability, even at the cost of significant routing runtime. Alternatively, one can evaluate placements by a layer-aware global router with a short time-out, which nevertheless correlates with the final router (and is potentially based on the same software implementation). This intermediate objective is more amenable to optimizations in global placement because its quick evaluation facilitates a tight feedback loop. In other words, intermediate placements can be evaluated many times, allowing the global placer to make proper adjustments. As we show in Section 5, due to the correlation between the fast and the final router, resulting routability-driven placements may fare better even with respect to the former, more traditional objective. This approach also facilitates early estimation of circuit delay and power in terms of specific route topologies. On the other hand, biasing the global placer away from HPWL to more sophisticated routability metrics may adversely affect the global placer's overall optimization capabilities. In other words, if HPWL increases too much, routability metrics will also increase.

In this work, we directly address the challenges of routability-driven placement. **First**, we develop *lookahead routing*, which invokes a fast high-quality 3-d global router, to quickly estimate routability. Since the produced routes can be used as a routing solution, our method can accurately and quickly report both congestion and routed wirelength. **Second**, to produce competitive placements in terms of both routed wirelength and HPWL, we integrate our lookahead routing into a flat, quadratic global placer, and enhance placement iterations by gently coercing cell locations and relieve congestion while preserving interconnect length. In detailed placement, we do not change the objective functions as in [42], but *prohibit moves* that aggravate routability. In global placement, we temporarily inflate cells in highly-congested regions to reserve whitespace during global placement. Traditionally, this has been accomplished either by cell bloating [4, 12, 26] during/after global placement, or by whitespace allocation [21, 27, 40] after placement. We observe that wirelength-driven global placers typically limit area utilization by a given amount through the entire layout based on target density. Therefore, in addition to *cell bloating*, we *dynamically adjust the target density* based on total routed wirelength.[3] This technique preserves overall solution quality and allows the placer to move cells in uncongested regions closer. **Third**, we develop a simultaneous place-and-route framework for global placement as well as a routability-driven detailed placement algorithm.

Our proposed methodology offers several advantages. First, since we use a global router to estimate congestion, the routes for all nets are known. Second, by enabling the global placer to directly redistribute whitespace in response to routing congestion, we establish a more precise feedback loop (compared to add-on techniques proposed previously). Third, by using a variable target density, we are trading off wirelength for routing demand in congested regions.

[1]M.-C. Kim and J. Hu contributed equally to this work.

[2]Half-Perimeter WireLength, defined using the net bounding box

[3]Partitioning-based placers can adjust target density on a per region basis [4, 27]. In force-directed placers, this feature is more difficult to implement and seems unnecessary.

Our key contributions include:

- A method to control routability within the global placer while preserving solution quality by dynamically adjusting the global target density
- An effective cell-bloating technique by dynamically adjusting cell width based on design's perceived difficulty
- A simultaneous place-and-route framework
- A congestion-aware detailed placement algorithm that moves cells only when this does *not* hurt routability
- Empirical results on the ISPD 2011 contest benchmarks that outperform every competitor on every benchmark with an average 2.04× improvement (Table 3) and a greater improvement 8.43× (Table 5) with a stronger global router.

The remainder of this paper is structured as follows. In Section 2, we survey prior art on congestion-driven placement, and review the baseline algorithms that we use. In Sections 3 and 4, we introduce several new techniques to improve the routability of placements. In Section 5, we empirically validate proposed algorithms. Section 6 concludes our work and discusses its impact.

2. PRIOR WORK

We briefly review the baseline place-and-route algorithms used in our work, then survey prior art on congestion maps and congestion-driven placement. For further background on placement see [17, Chapter 4] and for global routing see [17, Chapter 5].

SimPL [19] is a flat, force-directed global placer. It maintains a lower-bound and an upper-bound placement; the final solution is generated when the two placements converge.[4] The upper-bound placement is generated by applying *lookahead legalization (LAL)*, which is based on top-down geometric partitioning and non-linear scaling. Using this information, the lower-bound placement is generated by minimizing the quadratic objective

$$\Phi_q(\vec{x},\vec{y}) = \sum_{i,j} w_{i,j}\left((x_i - x_j)^2 + (y_i - y_j)^2\right) \qquad (1)$$

using the Conjugate Gradient method [28]. Here, $\vec{x}$ and $\vec{y}$ are coordinate vectors of the cells' (x,y) locations, and $w_{i,j}$ represents the connectivity between cells i and j. Two of top three teams in the ISPD 2011 contest, including ours, relied on the SimPL algorithm.

FastPlace-DP [24] is a wirelength-driven detailed placer based on greedy algorithms. It uses (i) cell clustering, (ii) global cell swapping, (iii) vertical cell swapping, and (iv) local reordering to improve wirelength. To determine which cells should be swapped, FastPlace-DP estimates the reduction in wirelength from swapping cells i and j by

$$gain(i,j) = \sum_{n\in N_i}(W_n - W'_n) - \sum_{n\in N_j}(W_n - W'_n) \qquad (2)$$

where N_i and N_j are the nets connected to cells i and j, and W and W' are the wirelength measurements before and after the swap.

BFG-R [14] is a global router based on Lagrangian relaxation. It decomposes multi-pin nets into two-pin subnets using MSTs and then iteratively routes all subnets until no violations are present. BFG-R prices each (sub)net by summing up the cost of used edges

$$cost(e) = base_e + \lambda(e) \times C(e) \times \rho(e) \qquad (3)$$

where $base_e$ is the base edge cost, $\lambda(e)$ is the history cost, $C(e)$ is current congestion, and $\rho(e)$ is the runtime penalty factor.

[4]The wirelength gap between the upper-bound and lower-bound solutions is useful to formulate convergence criteria.

Congestion Maps. To estimate congestion, prior approaches can be divided into three main categories: (i) static congestion estimation, (ii) probabilistic congestion estimation, and (iii) global routing estimation. Traditionally, the first two options have been the most popular, but the last option has recently been gaining acceptance thanks to advanced global routers designed to handle greater layout complexity. Empirical evidence suggests that constructive methods (generating routes) are faster and more accurate than probabilistic methods [37]. Table 1 summarizes these approaches.

Congestion-driven Placement. Known placement optimizations can be classified as (i) performed during global placement, (ii) external optimizations applied to intermediate solutions, (iii) performed during detailed placement, and (iv) post-placement processing (see Table 2). During global placement, the two most popular optimizations are to (i) relocate movable cells, or (ii) relocate them after bloating. These changes require modifying the placer, potentially including the optimization function. When working with quadratic, mincut and annealing-based placers, the generic ideas above must be adapted case by case. Additional optimizations can be applied to intermediate placement solutions, and then passed on to the next step of the design flow. During detailed placement, the most common approach is to modify the objective function of cell-swapping to account for congestion. After placement, the solution undergoes a series of changes to improve routability.

GENERAL APPROACH	SPECIFIC TECHNIQUES
STATIC	using Rent's Rule [41]
	net bounding box w/ weighting [6]
	building Steiner trees [27]
	pin density [4]
	counting nets in each region [35]
PROBABILISTIC	(uniform) wire density [30]
	pseudo-constructive wirelength [18]
	probabilistic pattern routing [36]
CONSTRUCTIVE	generating routes using A*-search on a collapsed (2-d) routing grid [37]
	using FastRoute [39] within an integrated framework [11]

Table 1: Prior congestion-estimation techniques.

3. SIMULTANEOUS PLACE-AND-ROUTE

In this section, we introduce a methodology for simultaneous place-and-route and discuss its components (see Figure 1). Given a placement instance, the baseline global placer quickly produces a tentative solution. Then, we apply *lookahead routing (LAR)* by calling our global router to estimate routing congestion and wirelength. We use this information during global placement by means of *cell bloating* and *dynamically adjusting the target density*. After several iterations of global placement, where the placer "heals" the placement for wirelength, lookahead routing is invoked again, and such iterations continue until overflow stops improving. *Congestion-aware detailed placement* is performed to recover whitespace and improve routed wirelength while maintaining routability.

We achieve an initial placement solution once the wirelength gap between the upper-bound and lower-bound solutions is within 25% of the 10^{th} iteration's total wirelength (see Section 2). After cell bloating, we run four iterations of lookahead legalization. Our disjunctive convergence criterion checks for three conditions: (i) the overflow has improved less than 3% after two consecutive rounds of LAR, (ii) the total overflow is less than 1% of the total edge capacity, or (iii) the global placement timeout of 60 iterations.

General Approach	Specific Techniques
During Global Placement	relocating the movable objects [15, 30, 31]
	cell bloating or cell inflation [4, 12]
	growing or shrinking placement regions [25]
Intermediate	local placement refinement [11]
During Detailed Placement	linear placement based on Steiner length in small windows [16, 27]
	incorporating congestion into the objective function [42]
	cell swapping based on only congestion and overlap [11]
After Placement	whitespace injection or reallocation [21, 27, 40]
	simulated annealing [10, 32]
	linear programming [22]
	network flows [33, 34]
	shifting modules by expanding global routing grid tiles [42]
	using pin density and congestion map to spread and bloat cells [26]

Table 2: Prior congestion-driven placement techniques.

3.1 Lookahead Routing (LAR)

To improve routability while preserving wirelength, global placement invokes lookahead routing. Unlike previous approaches, where only congestion information is reported, LAR estimates *both* interconnect length and routing congestion. Our router implementation accounts for (i) different wire widths and spacings, (ii) routing blockages, (iii) pins on different metal layers, and (iv) vias.

Wire widths and spacings at each metal layer are modeled separately. The resources consumed by a net are then estimated by

$$Usage(net) = \sum_{e \in net} minSpacing(l_e) + minWidth(l_e) \quad (4)$$

where net is the net routed, e is each edge in net, l_e is the metal layer that e is on, and $minSpacing(l_e)$ and $minWidth(l_e)$ are the respective minimum spacing and width required for l_e.

However, congestion estimates produced by this model can be misleading. For example, suppose two edges e_1 and e_2 on different metal layers are overflown, where e_1 is on Metal1, having $minSpacing(Metal1)+minWidth(Metal1) = 2$, and e_2 is on Metal4, having $minSpacing(Metal4)+minWidth(Metal4) = 8$. Suppose e_1 has two violating nets, yielding an overflow of 4, and e_2 has one violating net, yielding an overflow of 8. These actual overflows are now misleading, as e_1 is considered more congested, but its overflow is lower than that of e_2. Therefore, to accurately report congestion, we *normalize* the capacity for every edge e on metal layer l by

$$nCap(e_l) = \frac{Cap(e_l)}{minSpacing(l_e) + minWidth(l_e)} \quad (5)$$

where $Cap(e_l)$ is the original capacity of edge e on layer l. Note that when normalizing capacity, we also normalized, where each segment is defined as one routing segment, regardless of the layer.

Routing blockages are specified as physical locations in the layout area. Therefore, the routing resources blocked at each edge are proportional to the length of the blockage. However, if two obstacles overlap, the overlap is only counted once. To properly calculate capacity, we first take the union of all routing-obstacle shapes on each edge, and then consider each non-blocked region separately. For each non-blocked region r, the amount of usable capacity is

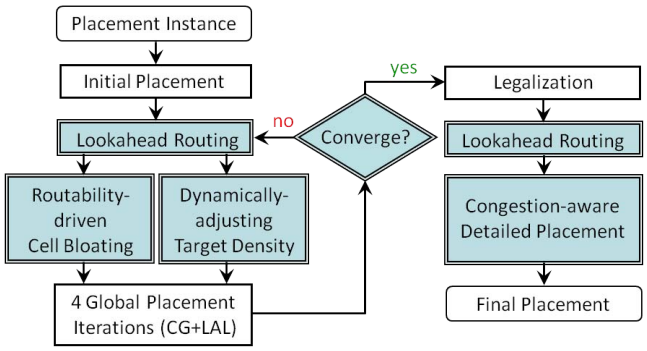


Figure 1: Our simultaneous place-and-route (SimPLR) flow. The baseline components are shown in transparent boxes, added routability-driven components have light-blue fill.

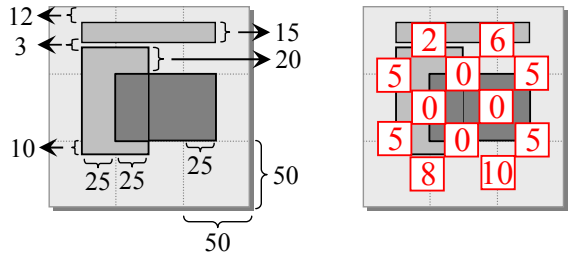
$$v(r_e) = \frac{dim(r_e)}{dim(e)} \quad (6)$$

where $dim(r)$ is the length of the non-blocked region on edge e, and $dim(e)$ is the length of e (i.e., height if e is a vertical edge, and width if e is a horizontal edge). Then, each edge's normalized capacity is

$$nCap(e_l) = \sum_{r_e \in R_e} \frac{v(r_e) \times Cap(e_l)}{minSpacing(l_e) + minWidth(l_e)} \quad (7)$$

The calculation of normalized capacity in the presence of routing obstacles is demonstrated in Fig. 2. In the example, let the length of every edge be 50, let the lower left coordinate be (0,0). Let the original edge capacity be 40, and let the minimum spacing plus the minimum width of this layer be 4. Since the vertical edge (50,0)-(50,50) has coordinates (50,40)-(50,50) blocked off, it only has $\frac{50-10}{50} = 80\%$ usable capacity. Since there is only one non-blocked region, the normalized capacity is $\frac{(80\% \times 40)}{4} = 8$. Similarly, the horizontal edge (50,50)-(100,50) has no usable capacity, as it is entirely blocked off, so its normalized capacity is 0.

Elevated pins. The ISPD 2011 contest benchmarks, derived from industrial ASICs and SoCs designs, include contact pins on multiple metal layers. This poses a challenge for traditional global routing techniques, where routing is first performed on a 2-d grid and then projected onto a 3-d grid. Therefore, we pursue a different strategy. We decompose all multi-pin nets into two-pin subnets, and perform 3-d maze routing.



$dim(e_v) = dim(e_h) = 50$ $\quad$ $origCap(e_v) = origCap(e_h) = 40$
$minWidth(e) + minSpacing(e) = 4$

Figure 2: Accounting for routing blockages, where $dim(e) = 50$ for each edge, two of three routing blockages overlap. On the left, the lengths of each routing blockage and non-blocked region are shown. On the right, the normalized capacities are calculated for each edge. With no blockages, an edge has a normalized capacity of 10.

3.2 Congestion-based Cell Bloating

After lookahead routing, we inflate all cells located in congested regions. The congestion at gcell g, located at (x,y), is

$$C(g(x,y)) = \frac{Usage(g(x,y))}{Cap(g(x,y))} \tag{8}$$

where $nUsage$ and $nCap$ are respectively the normalized usage and capacity at $g(x,y)$. The usage at each gcell is defined as

$$\begin{aligned} Usage(g(x,y)) = & \\ & \max(nUsage(e(x \pm 1, y)), nCap(e(x \pm 1, y))) + \\ & \max(nUsage(e(x, y \pm 1)), nCap(e(x, y \pm 1))) \end{aligned} \tag{9}$$

and the capacity at each gcell is defined as

$$Cap(g(x,y)) = nCap\left(e(x \pm 1, y)\right) + nCap\left(e(x, y \pm 1)\right) \tag{10}$$

where $nUsage$ is the normalized usage for edge e, and $nCap$ is the normalized capacity for e. Therefore, if $C(g(x,y)) > 1$, then $g(x,y)$ is considered congested. If at least one of the neighboring edges is congested, then the gcell is considered congested.

Then, for every cell in each congested gcell, we apply cell bloating by setting the cell's new width to

$$\max(width(cell)+1,\ 1+\theta \cdot \Lambda(cell) \cdot C(g(x,y)) \cdot deg(cell)) \tag{11}$$

where $width$ is the current width of $cell$, θ is an adaptive function (described below), Λ is the number of times the $cell$ has been in a congested gcell, and deg denotes the cell degree (the number of cell pins connected to wires).

Our cell bloating approach is inspired by CRISP [26], but differs in three major ways. **First**, we apply cell bloating *during global placement*, while CRISP bloats cells *after placement*. We can therefore perform large-scale changes and, in our experience, our placer better adjusts to bloated cells, resulting in a smaller wirelength penalty. **Second**, we use *gcell-centric congestion* estimation, while CRISP uses *edge-centric congestion* estimation with a pin-density map. Our style of congestion estimation improves integration with a global router. Pin density has been a popular estimation technique for designs with relatively few metal layers. However, with modern 9+ layer interconnect stacks, it primarily affects the success of *detailed routing*, which is orthogonal to our work. **Third**, while CRISP relies on a constant θ, our θ is a routing-solution-dependent function (described below), and based on the design's estimated difficulty and its routability. The intuition is that if a design is difficult to place or route, cells in congested regions need additional whitespace. Therefore, cells in those regions should be more inflated. We define $\theta(G)$ as

$$\theta(G) = \max\left(0, \alpha \cdot \eta(G) \cdot \xi(G) + \beta\right) \tag{12}$$

where G is the set of all gcells, α and β are constants determined from linear regressions, $\eta(G)$ indicates how *hard* a design is (e.g., how much available routing capacity there is), which is relatively insensitive to the routed solution, and $\xi(G)$ indicates the routability of the design, and is based on lookahead routing. We define $\eta(G)$ and $\xi(G)$ as

$$\eta(G) = \sum_{g \in G} \frac{Usage(g)}{Cap(g)} \qquad \xi(G) = \frac{TOF(G)}{TCap(G)} \tag{13}$$

where $OF(G)$ and $Cap(G)$ are the respective total overflow and total capacity of all gcells in G. In our implementation, we empirically determined the values $\alpha = 0.017$ and $\beta = -0.01$ based on numerical regression (but not benchmark-specific tuning).

3.3 Dynamic Adjustment of Target Density

Target density (utilization) is one of the most critical parameters to trade off routability for wirelength in the final placement. However, finding the best target density for routability-driven placements remains an open problem. Unnecessarily high target density leads to better HPWL, but may also cause routing failures [1]. Lower target density, on the other hand, may increase the overall routed wirelength, which would lead to longer detours and consume more routing resources. We set the initial target density as

$$\gamma_{init} = D_{ut} + \min(\max(\gamma_0 - D_{ut}, 0\%), \omega_D) \tag{14}$$

where D_{ut} is the design utility (given), γ_0 is a prediction of a *good* target density, and ω_D is the target density lower bound. If D_{ut} is too low (e.g., less than 35%), then the target density should be higher to encourage cell clustering. Conversely, cells should be spread apart if D_{ut} is too high. Empirically, we observed that setting $\gamma_0 = 50\%$ when $\omega_D = 15\%$ provides a reasonable trade-off between routability and routed length.

After lookahead routing and cell bloating, the target density is updated as

$$\gamma = \min(\frac{area(C_m)}{area(D) - area(C_f)} + \phi, 95\%) \tag{15}$$

where C_m is the set of movable cells, C_f is the set of fixed cells, D is the design, $area$ returns the total area of input (bloated cells included), and ϕ is a constant that increases every time LAR reports an increase in routed wirelength. In our implementation, ϕ is initially $\gamma_{init} - D_{ut}$, and increases by 1% when wirelength increases.

4. CONGESTION-AWARE DETAILED PLACEMENT

Traditional wirelength-driven detailed placement may pack cells in regions that are difficult to route. In the context of the FastPlace-DP algorithm, we modify both global cell swapping and vertical cell swapping to be *congestion-aware* (see Algorithm 1) in two ways: (1) we only allow cell move that do not harm routability, (2) we encourage cells to move out of congested regions.

The subroutine perform_swap(c_i, c_j, $pred$) swaps two cells c_i and c_j if $pred$ is true.[5] For each movable cell c_i, we consider its

Algorithm 1 Congestion-aware Detailed Placement

```
1.  C_m = Set of all movable cells
2.  G_c = Set of all congested gcells
3.  C(i) = Congestion in the position cell i
4.  foreach c_i ∈ C_m
5.  find the optimal region R_i of c_i
6.  find b_swap, the benefit of swap with a cell c_j ∈ R_i
7.  find b_move, the benefit of move to a space s ∈ R_i
8.    if (c_i ∉ G_c && c_j ∉ G_c)
9.      if (b_swap ≥ b_move)
10.       perform_swap(c_i, c_j, (b_swap > 0))
11.     else
12.       perform_swap(c_i, s, (b_move > 0))
13.   else if (c_i ∈ G_c && c_j ∉ G_c)
14.     perform_swap(c_i, s, true)
15.   else if (c_i ∉ G_c && c_j ∈ G_c)
16.     perform_swap(c_i, c_j, (deg(c_i) < deg(c_j)))
17.   else
18.     if (C(c_i) > C(c_j))
19.       perform_swap(c_i, c_j, (deg(c_i) > deg(c_j)))
20.     else
21.       perform_swap(c_i, c_j, (deg(c_i) < deg(c_j)))
22. end foreach
```

[5] A single cell can "swap" with an empty location.

BENCHMARK (source: IBM Research)	#cells	SimPL with FastPlace-DP			Best in Contest		SimPL**R** with Ca-DP		
		RtWL	OF	Runtime	RtWL	OF	RtWL	OF	Runtime
SUPERBLUE1	847K	**14.32**	1354	**23.89**	14.70	108	14.48	**0**	51.69
SUPERBLUE2	1.01M	**27.10**	1191806	**37.87**	30.77	797898	29.20	**740050**	108.30
SUPERBLUE4	500K	**10.52**	45430	**7.54**	10.86	85538	10.68	**18444**	24.79
SUPERBLUE5	772K	**16.90**	272934	**23.53**	17.29	126186	16.98	**121894**	51.84
SUPERBLUE10	1.13M	**26.18**	**463858**	**33.68**	25.16	616742	26.69	567780	73.34
SUPERBLUE12	1.29M	**19.35**	1992246	**35.18**	22.89	415428	22.58	**181350**	43.32
SUPERBLUE15	1.12M	17.09	62274	**24.21**	17.91	125936	**17.07**	**49286**	43.33
SUPERBLUE18	483K	10.64	153556	**14.36**	**9.84**	31440	10.63	**21020**	21.38
Average		**0.96**×	**3.81**×	**0.52**×	**1.01**×	**2.04**×	**1.0**×	**1.0**×	**1.0**×
Geometric mean		**0.96**×	**2.63**×	**0.49**×	**1.01**×	**1.76**×	**1.0**×	**1.0**×	**1.0**×

Table 3: Routed wirelength (RtWL, ×10e6), routing overflow (OF), and runtime (in minutes) on ISPD 2011 benchmarks. The placements were evaluated by *coalesCgrip* [5] with a 15-min time-out.

BENCHMARK (source: IBM Research)	#cells	FastPlace-DP (after SimPLR)				Ca-DP (after SimPLR)			
		HPWL	RtWL	OF	Runtime	HPWL	ΔRtWL	ΔOF	Runtime
SUPERBLUE1	847K	**277.03**	**14.45**	**0**	**5.37**	279.01	0.376	**0**	9.83
SUPERBLUE2	1.01M	**657.03**	29.09	782348	**19.22**	660.09	**-0.195**	**-42298**	32.06
SUPERBLUE4	600K	231.78	10.71	22192	**2.96**	**231.44**	**-0.336**	**-3748**	4.62
SUPERBLUE5	772K	**354.23**	17.02	139012	**5.58**	355.05	**-0.386**	**-17118**	9.68
SUPERBLUE10	1.13M	**586.62**	**26.48**	**556678**	**7.99**	592.18	0.113	11102	18.26
SUPERBLUE12	1.29M	**376.59**	22.7	293516	**7.71**	377.27	**-0.119**	**-112166**	13.33
SUPERBLUE15	1.12M	**337.04**	**17.04**	56866	**6.60**	337.96	0.128	**-7580**	8.43
SUPERBLUE18	483K	**165.09**	10.64	23708	**2.92**	165.75	**-0.125**	**-2688**	4.44
Average		**1.00**×	**1.01**×	**1.18**×	**0.60**×	**1.00**×	**1.00**×	**1.00**×	**1.00**×
Geometric mean		**1.00**×	**1.01**×	**1.17**×	**0.60**×	**1.00**×	**1.00**×	**1.00**×	**1.00**×

Table 4: The impact of congestion-aware detailed placement on HPWL(×10e6), routed wirelength (×10e6), and overflow (OF) on ISPD 2011 benchmarks. Runtimes are given in minutes. Routing was performed by *coalesCgrip* [5] with a 15-min time-out.

best swap (with c_j) or move (with empty space s). If both actions result in positive gain, and both are in non-congested regions, then we revert to wirelength-driven decisions. If c_i is in a congested region and c_j is not, then we can improve routability by moving it to s. If c_i is not in a congested region, but c_i is and has fewer pins than c_j, we can potentially improve routability in subsequent moves if we decrease the number of routes that go through the congested region. Similarly, if both cells are in congested regions, then we only swap them if $deg(c_j) < deg(c_i)$. This ensures that the detailed placer does not harm routability.

5. EMPIRICAL VALIDATION

Our implementation is in C++, compiled with g++ 4.4.3, and validated on a 3.00 GHz Intel Core 2 CPU X9650 Linux workstation. We modified and integrated the (i) SimPL global placer [19], (ii) BFG-R global router [14], and (iii) FastPlace-DP detailed placer [24]. Significant changes were made to all three tools, and new algorithms were added, as described in Sections 3 and 4.

The evaluation of placement solutions was performed by coalesCgrip [5], which was mandated by the ISPD 2011 Routability-driven Placement Contest. CoalesCgrip was compiled with gcc 4.4.1, as specified by the contest organizers. Its runtime limit was set to 300 seconds for initial routing and 900 seconds for rip-up and reroute (RRR), which makes results machine-dependent. Therefore, we downloaded all placements produced by the top contestants, and reevaluated them on our workstation.

Our implementation of SimPLR uses BFG-R for LAR instead of coalesCgrip, which was not available in source code. Empirically, our router accurately predicts the regions of congestion reported by coalesCgrip while allowing us to implement our proposed interface that minimizes runtime overhead. Since our strong results are achieved *without* running coalesCgrip during global placement, SimPLR does not seem to require the knowledge of a specific downstream router. Though using different routers in one flow may not be ideal, this is not uncommon in multivendor industry flows, and our results indicate that such configurations can be successful.

Progress of global placement is illustrated in Figure 4 for the SimPL (with target density 50%) and SimPLR algorithms. Before the first invocation of lookahead routing (LAR) in SimPLR, the two progress identically since the initial target density (γ_{init}) of SimPLR is computed by Equation 14 to be 50%. The first invocation of LAR with subsequent cell bloating does not significantly impact wirelength, due to $\Lambda = 0$ in Equation 11. Lookahead legalization produces higher HPWL after the second LAR, but the impact on quadratic placement is small, and the disruption in roughly legalized placement is quickly healed. SimPLR invokes LAR 3-6 times per benchmark, which takes 27-58% of total runtime, averaging at 47.88%, and currently runs 2× slower than SimPL. Yet, SimPLR was among the two fastest placers at the ISPD 2011 contest.

Congestion-aware detailed placement (Ca-DP) is evaluated in Table 4. We report the (i) recovered HPWL, (ii) recovered routed length, and (iii) total overflow improvement using Ca-DP, versus FastPlace-DP [24]. Ca-DP barely changes HPWL and routed wirelength, but improves overflow by $1.18\times$.

Routability is reported in Table 3 and Figure 3: SimPLR consistently reduces total overflow across all benchmarks and makes SUPERBLUE1 fully routable. On the remaining benchmarks, compared to baseline wirelength-driven placer SimPL, we improve total overflow by 3.81× on average. Compared to the top results from the ISPD 2011 Contest, we produce the smallest overflow on all benchmarks, for an average 2.04× reduction. These results are further improved in Table 5 as discussed in conclusions.

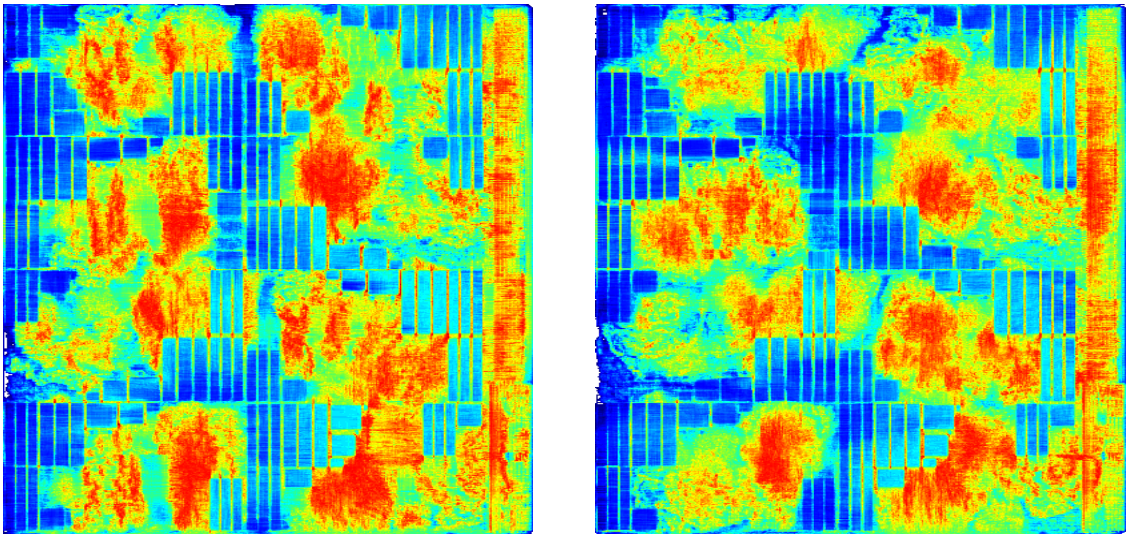


Figure 3: Congestion maps for SUPERBLUE15 for the best-reported placement at the ISPD 2011 contest (left) and SimPLR (right). Isolated red regions indicate peak congestion, dark-blue rectangles show unused routing resources.

6. CONCLUSIONS

Tight integration of major CAD tools is sometimes frowned upon in the industry because it may sharply increase software complexity, introduce subtle discrepancies and complicate software maintenance. However, such integration is highly sought in place-and-route, where high-performance global placers often generate hard-to-route solutions, creating unnecessary complications for downstream tools. The strategy pursued in our work is to give the placer advance, firsthand access to tentative net routes and resulting actual congestion maps (rather than crude estimates), as well as the ability to respond early and often. We believe that our proposed integration of global routing into global placement, *based on lookahead routing of upper-bound placements in the SimPL* algorithm, offers a particularly promising and "clean" path to effective simultaneous place-and-route. By communicating through a lightweight interface, the placer and the router quickly exchange multiple updates to cell locations and net routes, while maintaining the software infrastructure separated. Despite this software separation, the evolutions of routes and cell placements are coupled and occur simultaneously. Empirical data show that our techniques improve routability, reducing total overflow compared to top results from the ISPD 2011 contest, which represent prior art and several competing new techniques developed by our colleagues across the world.

The ISPD 2011 experimental protocol evaluated placements with only very short routing runs of coalesCgrip. To illustrate the full potential of SimPLR, we performed additional experiments, where coalesCgrip was given 12 and 24 hours. The results reported in Table 5 were obtained on a 2.53 GHz Intel Xeon CPU E5540 Linux workstation. While none of the IBM-released benchmarks could be completed without overflows at the ISPD 2011 contest, we have now completed half of them. Our results[6] show that the advantage of SimPLR solutions grows significantly when the evaluating router is used at its full strength. *Thus, modern place-and-route leaves room for improvement in both gate locations and wire routes, but such improvements are best achieved in cooperation.* Such optimizations use physical resources more efficiently, enable smaller dies, and decrease IC manufacturing cost [26].

[6]The placement solutions produced by SimPLR on ISPD 2011 benchmarks, as well as resulting routes, are available on demand.

Given that SimPLR internally invokes a full-fledged router with a limited number of iterations (BFG-R) that produces a *valid* routing solution, better optimized results can be requested at the cost of greater runtime. Runtime can be improved by making LAR more incremental. This SimPLR framework can target specific nets and facilitates several further extensions, especially in timing optimization, where the placer's early and direct access to actual routes can improve the accuracy of delay estimation.

Acknowledgments. We are grateful to Chris Chu and Natarajan Viswanathan for giving us access to FastPlace-DP.

7. REFERENCES

[1] S. N. Adya, I. L. Markov, and P. G. Villarrubia, "On Whitespace and Stability in Physical Synthesis", *Integration* 39(4), pp. 340-362, 2006.

[2] C. J. Alpert, Z. Li, M. D. Moffitt, G.-J. Nam, J. A. Roy, and G. Tellez, "What Makes a Design Difficult to Route", *ISPD*, pp. 7-12, 2010.

[3] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar (eds.), *Handbook of Algorithms for VLSI Physical Design Automation*, CRC Press, 2008.

[4] U. Brenner and A. Rohe, "An Effective Congestion Driven Placement Framework", *ISPD*, pp. 6-11, 2002.

[5] coalesCgrip: A Tool for Routing Congestion Analysis. `homepages.cae.wisc.edu/~adavoodi/gr/cgrip.htm`.

[6] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov, A. Zelikovsky, "On Wirelength Estimations for Row-based Placement," *TCAD* 18(9), pp. 1265-1278, 1999.

[7] T. F. Chan, J. Cong, M. Romesis, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: Enhanced Multilevel Mixed-size Placement with Congestion Control", *Modern Circuit Placement*, pp. 247-288, 2007.

[8] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang, "NTHU-Route 2.0: A Fast and Stable Global Router", *ICCAD*, pp. 338-343, 2008.

[9] Y.-J. Chang, T.-H. Lee, and T.-C. Wang, "GLADE: A Modern Global Router Considering Layer Objectives", *ICCAD*, pp. 319-323, 2010.

[10] C. E. Cheng, "RISA: Accurate and Efficient Placement Routability Modeling", *ICCAD*, pp. 650-695, 1994.

[11] C. Chu and M. Pan, "IPR: An Integrated Placement and Routing Algorithm", *DAC*, pp. 59-62, 2007.

[12] W. Hou, H. Yu, X. Hong, Y. Cai, W. Wu, J. Gu, and W. H. Kao, "A New Congestion-driven Placement Algorithm Based on Cell Inflation", *ASP-DAC*, pp. 723-728, 2001.

[13] C.-H. Hsu, H.-Y. Chen, and Y.-W. Chang, "Multi-layer Global Routing Considering Via and Wire Capacities", *ICCAD*, pp. 350-355, 2008.

[14] J. Hu, J. A. Roy, and I. L. Markov, "Completing High-quality Routes", *ISPD*, pp. 35-41, 2010.

[15] Z.-W. Jiang, B.-Y. Su, and Y.-W. Chang, "Routability-driven Analytical Placement by Net Overlapping Removal for Large-scale Mixed-size Designs", *DAC*, pp. 167-172, 2008.

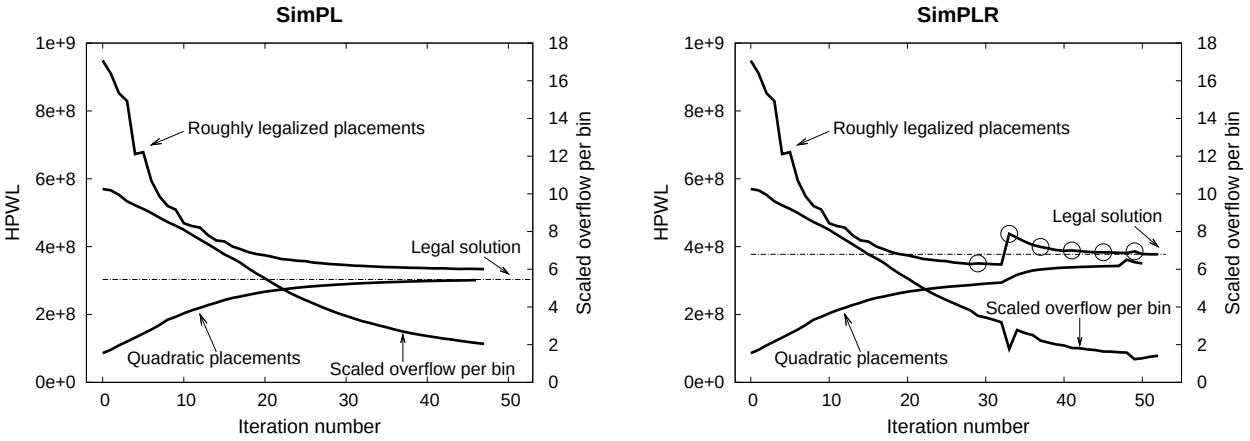


Figure 4: Progress of SimPL and SimPLR algorithms plotted against iteration counts (SUPERBLUE12). Each invocation of lookahead routing is marked with a circle. The second invocation of LAR and subsequent cell bloating visibly disrupt the quality of roughly legalized placements, with a smaller impact on quadratic placement.

BENCHMARK	Best in Contest				SimPLR with Ca-DP			
	coalesCgrip **12hr**		coalesCgrip **24hr**		coalesCgrip **12hr**		coalesCgrip **24hr**	
	RtWL	OF	RtWL	OF	RtWL	OF	RtWL	OF
SUPERBLUE1	15.02	**0**	15.02	**0**	15.02	**0**	15.02	**0**
SUPERBLUE2	31.11	41408	31.20	16768	29.39	**39132**	29.52	**9646**
SUPERBLUE4	10.87	2466	10.88	1262	10.67	**44**	10.67	**36**
SUPERBLUE5	17.38	19112	17.49	10582	17.11	**9510**	17.18	**4392**
SUPERBLUE10	25.09	31320	25.16	**10652**	26.68	**22268**	26.73	11104
SUPERBLUE12	22.99	5130	23.02	594	23.00	**3302**	23.04	**0**
SUPERBLUE15	17.97	4448	17.98	2264	17.09	**0**	17.98	**0**
SUPERBLUE18	9.86	**0**	9.86	**0**	10.63	**0**	10.63	**0**
Average	**1.00**×	**19.41**×	**1.00**×	**8.43**×	**1.00**×	**4.12**×	**1.00**×	**1.00**×
Geometric mean	**1.00**×	**9.00**×	**1.00**×	**3.09**×	**1.00**×	**2.99**×	**1.00**×	**1.00**×

Table 5: Routed wirelength (RtWL, ×10e6) and routing overflow (OF) on ISPD 2011 benchmarks. Routing was performed by *coalesCgrip* [5] with a longer time-out than in Tables 3 and 4. Average and geometric means are calculated excluding routable benchmarks, which under-represents the impact of proposed techniques.

[16] D. Jariwala and J. Lillis, "RBI: Simultaneous Placement and Routing Optimization Technique", *TCAD* 26(1), pp. 127-141, 2007.

[17] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, "VLSI Physical Design: From Graph Partitioning to Timing Closure", Springer 2011.

[18] A. B. Kahng and X. Xu, "Accurate Pseudo-constructive Wirelength and Congestion Estimation" *SLIP*, pp. 61-68, 2003.

[19] M.-C. Kim, D.-J. Lee, and I. L. Markov, "SimPL: An Effective Placement Algorithm", *ICCAD*, pp. 649-656, 2010.

[20] T.-H. Lee, Y.-J. Chang, and T.-C. Wang, "An Enhanced Global Router with Consideration of General Layer Directives", *ISPD*, pp. 53-60, 2011.

[21] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden, "Routability-driven Placement and White Space Allocation", *ICCAD*, pp. 394-401, 2004.

[22] Z. Li, W. Wu, and X. Hong, "Congestion Driven Incremental Placement Algorithm for Standard Cell Layout", *ASP-DAC*, pp. 723-728, 2003.

[23] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "Multi-threaded Collision-aware Global Routing with Bounded-length Maze Routing", *DAC*, pp. 200-205, 2010.

[24] M. Pan, N. Viswanathan, and C. Chu, "An Efficient and Effective Detailed Placement Algorithm", *ICCAD*, pp. 48-55, 2005.

[25] P. N. Parakh, R. B. Brown, and K. A. Sakallah, "Congestion Driven Quadratic Placement", *DAC*, pp. 275-278, 1998.

[26] J. A. Roy et al, "CRISP: Congestion Reduction by Iterated Spreading during Placement", *ICCAD*, pp. 357-362, 2009.

[27] J. A. Roy, J. F. Lu, and I. L. Markov, "Seeing the Forest and the Trees: Steiner Wirelength Optimization in Placement", *TCAD* 23(4), pp. 632-644, 2007.

[28] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

[29] N. Selvakkumaran, P. N. Parakh, and G. Karypis, "Perimeter-Degree: A Priori Metric for Directly Measuring and Homogenizing Interconnection Complexity in Multilevel Placement", *SLIP*, pp. 53-59, 2003.

[30] P. Spindler, F. M. Johannes, "Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement", *DATE*, pp. 1226-1231, 2007.

[31] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control", *ASP-DAC*, pp. 135-140, 2007.

[32] M. Wang and M. Sarrafzadeh, "On the Behaviour of Congestion Minimization during Placement", *ISPD*, pp. 145-150, 1999.

[33] M. Wang and M. Sarrafzadeh, "Model and Minimization of Routing Congestion", *ASP-DAC*, pp. 185-190, 2000.

[34] M. Wang X. Yang, M. Sarrafzadeh, "Congestion Minimization during Placement", in *TCAD* 19(10), pp. 1140-1148, 2000.

[35] M. Wang, X. Yang, K. Eguro, and M. Sarrafzadeh, "Multicenter Congestion Estimation and Minimization During Placement", *ISPD*, pp. 147-152, 2000.

[36] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic Congestion Prediction", *ISPD*, pp. 204-209, 2004.

[37] J. Westra and P. Groeneveld, "Is Probabilistic Congestion Estimation Worthwhile?" *SLIP*, pp. 99-106, 2005.

[38] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "A Parallel Integer Programming Approach to Global Routing", *DAC*, pp. 194-199, 2010.

[39] Y. Xu, Y. Zhang, and C. Chu, "FastRoute 4.0: Global Router with Efficient Via Minimization", *ASP-DAC*, pp. 576-581, 2009.

[40] X. Yang, B.-K. Choi, M. Sarrafzadeh, "Routability-driven White Space Allocation for Fixed-die Standard-cell Placement", *TCAD* 22(4), pp. 410-419, 2003.

[41] X. Yang, R. Kastner, and M. Sarrafzadeh, "Congestion Estimation During Top-down Placement", *TCAD* 21(1), pp. 72-80, 2002.

[42] Y. Zhang and C. Chu, "CROP: Fast and Effective Congestion Refinement of Placement", *ICCAD*, pp. 344-350, 2009.