IPR: An Integrated Placement and Routing Algorithm

Min Pan
Cadence Design Systems, Inc.
San Jose, CA, 95134
minpan@cadence.com

Chris Chu
Electrical and Computer Engineering Dept.
Iowa State University, Ames, IA 50011
cnchu@iastate.edu

***Abstract*— In nanometer-scale VLSI technologies, several interconnect issues like routing congestion and interconnect delay have become the main concerns in placement. However, all previous placement approaches optimize some very primitive interconnect models during placement. These models are far from the actual interconnect implementation in the routing stage. As a result, placement solution considered to be good by primitive interconnect models may turn out to be poor after routing. In addition, the placement may not even be routable and timing closure may not be achievable.**

In this paper, we propose to address the inconsistency between the placement and routing objectives by fully integrating global routing into placement. As a first attempt to this novel approach, we focus on routability issue. We call the proposed algorithm for routing congestion minimization IPR (Integrated Placement and Routing). To ensure the algorithm to be computationally efficient, efficient placement and routing algorithms *FastPlace*, *FastDP* and *FastRoute* are integrated, and well-designed methods are proposed to integrate them efficiently and effectively. Experimental results show that IPR reduces overflow by 36%, global routing wirelength by 3.6%, and runtime by 36% comparing to ROOSTER [5], which is the previous best academic routability-driven placer.

CATEGORIES & SUBJECT DESCRIPTORS

B7.2 [Hardware]: Integrated Circuits - *Design Aids*

GENERAL TERMS

Algorithms, Design, Performance

KEYWORDS

Placement, Routing, Integration

I. INTRODUCTION

As VLSI technology enters nanometer regime, placement has become a critical step in VLSI design flow. The two major causes are both related to the increasing dominance of interconnect in nanometer-scale VLSI technologies. First, placement largely determines the performance of a circuit. In advanced VLSI technology, interconnect delay has become the determining factor of circuit performance. Placement decides the length and hence the delay of interconnect wires to a large extent. Therefore, a good placement solution can substantially improve the performance of a circuit. Second, placement also determines the chip size. Because of the shrinking of device size, the chip area is no longer determined by total cell area, but by the limited routing resources. Extra "white space" is commonly added to provide enough wire tracks to resolve routing congestion. It is typical that more than half of the modern chip is occupied by white space. Placement is an important step to reduce white space required by minimizing routing congestion and to allocate white space appropriately based on routing congestion.

Even though interconnect issues are the main concerns during placement, they are not appropriately handled in traditional physical design flow in which placement and routing are performed sequentially. In placement stage, the characteristics of interconnects are typically approximated by inaccurate models (e.g., half-perimeter of bounding rectangle, clique model, star model). These models are far from the actual implementation of interconnects in routing stage. Due to this inconsistency between the objectives optimized in placement stage and in routing stage, the placed circuit may not be routable and the estimated interconnect timing may not be achievable. Even if a placed design is routable and achieves timing closure, the placement and routing solutions generated by this sequential approach may be far from optimal.

In this paper, we address the routability issue by improving the ways routing congestion information are generated and utilized during placement. First, we discuss previous works on routability-driven placement. In placement algorithms, the most commonly used optimization objective is total wirelength. Wirelength minimization can be considered an indirect way to reduce routing congestion as total wirelength is equal to total routing demand. However, a placement with small wirelength may be unroutable due to an uneven routing demand distribution. Hence, many routability-driven placement algorithms are proposed to explicitly account for routing congestion in order to produce routable placements. Yang et al. [1] built congestion maps after global placement, and applied annealing moves to minimize a congestion metric. Another technique known as WSA [2] is applied after detailed placement. WSA uses congestion maps to identify areas with high congestion and injects whitespace into these areas in a top-down fashion. After whitespace allocation and legalization, window based detail placement techniques are applied to reduce wirelength. Cell bloating [3] and cell spreading [2] are whitespace allocation techniques by tying whitespace to specific cells. In [4], Jariwala and Lillis employed a single-trunk Steiner tree model to reduce congestion in FPGAs. Recently, Roy et al. [5] developed a placement technique called ROOSTER to optimize Steiner-tree wirelength together with whitespace allocation in a partitioning placement framework. ROOSTER improves overall Place-and-Route results over previous works.

All previous works tried to achieve routability by including some congestion estimation and using that information as a guide during placement. However, as pointed out in [6], it is impossible for congestion estimators to predict the routing congestion accurately because various routers will generate routing solutions with very different routing congestion. The only possible way to account for congestion accurately is to apply the same technique in both congestion estimation and routing. Therefore, in order to get accurate interconnect information, it is desirable to integrate routing into placement process.

Global routing allocates routing demand globally over the chip area. It generates interconnect information very close to the final routing implementation and is suitable for accurate estimation of interconnect wirelength, congestion, timing and buffering, etc. If we are able to integrate global routing into placement, accurate congestion based on routing of nets becomes available. Hence, placement solutions with routability guaranteed by a global routing solution can be obtained.

However, the major obstacle for this integration is runtime complexity. Since both placement and routing are computationally expensive problems, integrating them together will make it much more difficult to manage. Due to the high runtime of traditional global routers, it is impractical to perform global routing repeatedly during placement. Therefore, we need to have very efficient placement and routing algorithms, as well as well-designed methods to integrate them efficiently and effectively.

In this paper, we propose a novel integrated placement and routing algorithm called IPR, based on efficient global placement, detailed placement and global routing algorithms FastPlace [7], [10], FastDP [11] and FastRoute [6]. In this algorithm, global routing is closely integrated into the iterative placement process. Congestion maps are generated based on actual routing and frequently updated as the placement algorithms change cell locations. The congestion map is used to direct the placement optimization to reduce routing congestion. Here, we are optimizing the routing congestion directly instead of indirect objectives like whitespace allocation. Therefore, we are able to get high-quality placement solutions in terms of routability. As far as we know, this is the first work which fully integrates placement and routing in the general ASIC design flow.

The major contributions of this paper are:

- An integrated placement and routing framework.
- A Steiner wirelength driven local refinement technique to optimize the Steiner-tree wirelength and to evenly distribute the cells in global placement stage.
- A routability-driven refinement (RDR) technique to directly optimize the routability in global placement stage.
- A routability-driven global swap (RGS) technique to improve routability by swapping cells globally in detailed placement stage.
- A routability-driven local swap (RLS) technique to improve routability by swapping neighboring cells in detailed placement stage.

Experimental results show that the new integrated placement and routing algorithm can achieve significantly better placement solutions in terms of routability in less runtime compared to ROOSTER. In addition, we show that even with the same placement framework as IPR, using simple probabilistic congestion model instead of global routing is not able to direct the placement to achieve the high-quality placement and routing solution as in the integrated approach. This justifies the necessity of integrating global routing with placement to generate accurate interconnect information and direct placement process.

DAC 2007, June 4-8, 2007, San Diego, California, USA

Note that although we only focus on routability issue in this paper, other issues such as timing, buffering can also be handled within this framework because routing information is available during the placement process.

The remainder of the paper is organized as follows. In Section II, we first review the efficient placement and routing algorithms FastPlace, FastDP and FastRoute. Then we discuss the issues with placement and routing integration and give the flow of our integrated approach. In Section III, the techniques to integrate placement and routing are described in detail. In Section IV, we perform experiments and show the comparison results. Finally, the paper concludes with a summary of results and directions of future work.

II. Overview of IPR

In this section, we first review the efficient placement and global routing algorithms FastPlace, FastDP and FastRoute. Our integrated placement and routing approach is based on them. Then we discuss the major difficulties for integration. Finally, we give the flow of our integrated placement and routing algorithm.

A. FastPlace, FastDP and FastRoute

FastPlace [7] is a very efficient wirelength-driven placement algorithm. It was first proposed to handle standard cell designs and later extended to handle mixed-size designs [10]. In general, it is a force-directed placement algorithm. It formulates the placement problem as convex quadratic programming. In order to achieve an even placement, a cell shifting technique is proposed to spread out the cells. In addition, half-perimeter wirelength (HPWL) is optimized by an iterative local refinement (ILR) technique. FastPlace can generate good placement solutions in terms of HPWL with very fast runtime.

FastDP [11] is an efficient and effective detailed placer. It works on a legalized placement and iteratively reduces HPWL. In the main loop, global swap, vertical swap and local re-ordering techniques are applied to reduce the wirelength. For the cells in the same placeable segment, a single-segment clustering technique is proposed to find the optimal cell positions respecting their original ordering.

FastRoute [6] is a very fast high-quality global router. Different from traditional global routers, it focuses on Steiner tree structures to resolve congestion. Congestion driven Steiner tree topology construction algorithm and edge shifting technique are proposed to construct good Steiner trees. Later in [12], monotonic routing and multi-source multi-sink maze routing techniques are proposed to enhance the pattern routing and traditional maze routing to achieve high-quality routing solutions. Experimental results show that FastRoute generates much better global routing solutions with tens to hundreds of times speedup compared to all other state-of-the-art academic global routers.

IPR is based on these efficient placement and routing techniques. Their efficiency is one of the reasons that we are able to integrate placement and routing with reasonable runtime.

B. Integration Issues

Although we have the placement and routing techniques mentioned in Section II-A, it is not trivial to assemble them together to get an integrated approach. There are several issues for integrated placement and routing to obtain high-quality solutions with affordable runtime.

First, during the iterative placement process, cells are moving all the time. As they change positions, we need to redo the routing. Although we have a very fast global routing algorithm, simply performing global routing after movement of each cell is not feasible because each cell will change positions thousands of times during placement and the cell number is huge. Therefore, we perform incremental rip-up and reroute to keep the routing updated. In this method, when we moving a cell, only the routing trees/branches connecting to this cell will be ripped up and rerouted. This incremental updating strategy may not give the best possible routing. However, it is good enough for routing congestion estimation and will not lead to an unaffordable runtime.

Second, in the IPR framework, cell positions are gradually refined from stage to stage. Hence, we adapt our incremental routing approach to this increasing accuracy of placement. We apply routing techniques with increasing accuracy during the whole placement and routing process to balance the accuracy and runtime. As we will discuss in detail in Section III, the routing information is also becoming more and more accurate as placement approaches final solution.

C. IPR Flow

The proposed IPR framework follows the basic flow of *FastPlace*. It also has three stages: (1) global placement, (2) legalization, and (3) detailed placement. However, different from *FastPlace*, the objective is no longer HPWL, but good routability. Hence, although we still have these three stages in the flow, many new techniques are introduced due to the new objective. As we will see later, these techniques closely integrate routing to get accurate congestion information and directly optimize routing congestion in the placement process.

The flow of IPR is summarized in Figure 1. Notice that the new techniques for reducing routing congestion are *Steiner-WL based Iterative Local Refinement*, *Routability Driven Refinement*, *Routability-driven Global Swap* and *Routability-driven Local Swap*. In addition, global routing is closely integrated in these new techniques to obtain global routing and congestion map. Finally, the output is not only a placement with good routability but also the global routing solution over it.

Stage 1: Global Placement

1. Repeat
   a. Solve convex quadratic program
   b. Perform cell-shifting and add spreading force
2. Until the placement is roughly even
3. Repeat
   a. perform ***Steiner-WL based Iterative Local Refinement***
4. Until the placement is even and no significant improvement on Steiner-WL
5. Run *FastRoute* to get an initial global routing and congestion map
6. Repeat
   a. Perform ***Routability Driven Refinement*** to reduce routing congestion
   b. Run *FastRoute* to get updated routing and congestion map
7. Until no significant improvement on congestion

Stage 2: Legalization

8. Move standard cells to legal positions and remove overlaps, minimize the disturbance to the global placement

Stage 3: Detailed Placement

9. Run *FastRoute* to get an initial global routing and congestion map
10. Repeat
    a. Apply ***Routability Driven Global Swap*** to reduce routing congestion
    b. Updating routing and congestion map by rip-up and reroute
11. Until no significant improvement on congestion
12. Run *FastRoute* again to get global routing and congestion map
13. Repeat
    a. Apply ***Routability Driven Local Swap*** to reduce routing congestion
    b. Updating routing and congestion map by rip-up and reroute
14. Until no significant improvement on congestion
15. Run *FastRoute* to obtain the final global routing solution

Fig. 1. Flow of IPR

III. Integrated Placement and Routing Techniques

In this section, we will introduce the new techniques in this integrated placement and routing framework in detail.

A. Global Placement

As shown in Figure 1, the global placement stage has 3 phases: (1) quadratic placement (lines 1-2), (2) St-WL based Iterative Local Refinement (lines 3-4), and (3) Routability Driven Refinement (lines 5-7).

1) Quadratic placement: In this phase, our global placement follows FastPlace. Basically, it is a quadratic placement method. It tries to optimize the quadratic objective function which sums up the cost of all nets. It can be written in matrix notation as:

$$\Phi(x,y) = \frac{1}{2}x^T Q x + d_x^T x + \frac{1}{2}y^T Q y + d_y^T y + constant \qquad (1)$$

In this quadratic placement formulation, only two-point connections can be used to express wirelength. So a hybrid net model [7] is used to transform the multi-pin nets into two-point connections. Since equation (1) is separable into $\Phi(x,y) = \Phi(x) + \Phi(y)$, we can optimize $\Phi(x)$ and $\Phi(y)$ separately. Therefore, the objective function $\Phi(x)$ and $\Phi(y)$ can be minimized by solving the following system of linear equations:

$$\begin{aligned} Qx + d_x &= 0 \\ Qy + d_y &= 0 \end{aligned} \qquad (2)$$

Equation (2) gives the solution to the unconstrained problem of minimizing the quadratic function in equation (1). However, it does not

consider the overlap among cells. Therefore, the cells are not evenly distributed over the placement region. Therefore, we apply the cell shifting technique as in [7] to even out the placement by distributing the cells over the placement region while retaining their relative ordering obtained from the quadratic programming. During cell shifting, the placement region is binned and the utilization of each bin is computed. The cells are then spread depending on the utilization of their respective bins. The basic intuition behind cell shifting is to even out the utilization of adjacent bins. After each iteration of cell shifting, pseudo nets are added to the cells to prevent them from collapsing back to their previous positions. We iteratively solve the quadratic program and perform cell shifting until the placement is relatively even, as shown in lines 1-2 of Figure 1. The goal is to achieve an relatively evenly distributed initial placement for the later phases.

2) St-WL based Iterative Local Refinement: After the initial placement is obtained, the original *FastPlace* apply HPWL based *Iterative Local Refinement* (*ILR*) to optimize HPWL and generate a more even placement. For the IPR algorithm, HPWL is no longer the objective. Instead, the goal is to achieve an even placement with less routing congestion. Hence, the *ILR* technique in [10] is modified to optimize the rectilinear Steiner tree wirelength. Similar to ILR technique in [10], we try 8 different directions to move a cell in order to reduce the Steiner-wirelength. The new technique is called *Steiner-WL based Iterative Local Refinement* (*St-WL ILR*) (lines 3-4 in Figure 1).

The reason to use Steiner-WL as the objective is that it has much higher fidelity with routed wirelength than HPWL. Since the placement is still very unstable and cells are moving frequently over a long distance, it is not necessary to consume a lot of runtime to run global routing to obtain accurate routing information for the very inaccurate placement at this point. Hence, we try to minimize the Steiner-WL in order to minimize the total routing demand, which is beneficial to the overall routing congestion. In this step, since we need to evaluate Steiner-WL all the time, the Steiner-WL algorithm has to be very fast. Hence, we employ the extremely fast rectilinear Steiner minimal tree algorithm FLUTE [8], [9].

3) Routability Driven Refinement: After *St-WL ILR*, we have an even and quite stable placement. We use it as a starting placement for the consideration of direct congestion reduction. *FastRoute* is invoked (line 5 in Figure 1) to generate the global routing solution and the congestion map over the current placement. From now on, accurate routing information is used to direct the placement optimization.

Based on the global routing and congestion map obtained by FastRoute, a new *Routability Driven Refinement* (*RDR*) technique is designed to optimize the routability directly. First, we divide the placement region into regular bin structure. The bin size is about the total area of 4 standard cells on average. For each cell, we try to move it into the 8 neighboring bins, as shown in Figure 2. We choose the same relative positions in the neighboring bins as it is located in the current bin to be the target positions. RDR is similar to the *ILR* technique in [10]. However, the criterion to move a cell is no longer the score based on bin cell utilization change and HPWL reduction. Now we measure the routing congestion before and after moving a cell to each of the 8 neighboring bins. Then based on the change in bin cell utilization and routing congestion reduction, we decide whether to move a cell to a certain bin or not. Note that we are not using any congestion estimator to predict the congestion variation, but applying rip-up and reroute to update routing solution and congestion map. Therefore, we have high confidence for each move to reduce the routing congestion.

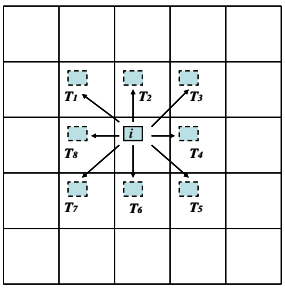

Fig. 2. Routability Driven Refinement

In below, we discuss how to update routing solution and congestion map when we move a cell. For any cell i, we tentatively move i to the 8 target positions. For each target position T_j ($j = 1, \ldots, 8$), we first need to rip-up the routing for the nets connecting to i. Then we move i to T_j, and reroute the nets connecting to i. We compute the routing congestion difference before and after moving i to T_j. At the same time, we need to control the bin cell utilization to maintain the even placement. Hence, we use a score which is a combination of routing congestion reduction and bin cell utilization change. We will move i to the best T_j which has the highest score. If all 8 target positions are worse than the current position, we do not move i.

One issue in the above procedure is the rip-up and reroute of the nets connecting to i. This is a very time consuming part. Because we need to run RDR over all the cells for many iterations, and in each iteration we need to compute the score for 8 target positions for each cell i, the nets connecting to i need to be ripped up and rerouted many times. In order to reduce the runtime, we make some trade-off between the accuracy and runtime. For each net n connecting to i, instead of rip-up and reroute the routing tree for the whole net n, we just rip-up the routing tree branch connecting to i (from a Steiner node to cell i in the tree of net n). As a result, only one branch for each net connecting to i is updated. In this approach, we assume the tree topology of n will not be changed after moving i. Since we only move i for a small distance (one bin size), there is no need to change the routing tree topology in most cases. However, we cannot make this assumption if the cells are moved by a longer distance. Therefore, after one iteration of RDR in which all cells are considered once, the routing solution will be fully reconstructed by FastRoute to maintain the accuracy of routing information (lines 6b). This includes initial congestion map generation, congestion-driven Steiner tree construction, and maze/pattern routing.

After the loop of *RDR* (line 7-8), an even placement with good routability is obtained. Now the cells are in good relative positions with overlaps among each other.

B. Legalization

For legalization, the main goal is to put all cells in legal positions and remove overlaps. At the same time, we want to maintain the cell positions from global placement to ensure the routability of the global placement solution will not be destroyed. Hence, we try to minimize the disturbance of the cell positions from to the global placement solution.

The legalization technique is similar to the one in *FastPlace*. The basic idea is to divide each standard cell row into several segments. Then the cell utilization of each segment is computed, and cells are iteratively move to neighboring segments in order to ensure all segments are within capacity. For placement with very high cell utilization (95% or more), very little whitespace is available and legalization becomes very hard. Hence, we add a row utilization control method to favor the move of cells from a standard cell row with over capacity to an adjacent row with lower capacity. In this way, we can quickly make all rows within capacity. Then a legalized placement can be obtained by arranging the cells in each row in order without overlap.

C. Detailed Placement

In the detailed placement stage, *FastRoute* is first employed to generate an updated global routing solution and congestion map. After that, *Routability-driven Global Swap (RGS)* and *Routability-driven Local Swap (RLS)* are performed repeatedly to refine the legalized placement and further reduce routing congestion.

1) Routability-driven Global Swap (RGS): In this technique, we move the cells globally to reduce the routing congestion. Similar to *Global Swap* in [11], for each cell i, we first find its optimal region. Then we consider each cell j in i's optimal region as a candidate and make the tentative swap. After measuring the benefit for each tentative swap, we pick the cell j which results in the most benefit and make the real swap for i and j. The major difference between *RGS* and *Global Swap* in [11] is that we no longer care for HPWL, but routing congestion. In original *Global Swap*, the benefit function is composed of two parts. One is the HPWL reduction due to the swap, and the other is the penalty part charged for the cell overlap caused by the swap. In *RGS*, we still have the overlap penalty part in the score function, but we use the routing congestion reduction to substitute the HPWL reduction. Next, we will focus on how to measure and update the routing congestion during the placement process.

Before any tentative swap, we measure the congestion score for current routing. For each candidate cell j, we perform the following operations. (1) Rip-up all the routing trees for all the nets connecting to i and j. (2) Swapping i with j, update the pin locations for the affected nets. (3) Reroute all the nets connecting to i and j for the new pin positions. (4) Measure the routing congestion score. (5) Recover the original routing trees and congestion map before considering another tentative swap. In this way, we can get the congestion score for each candidate cell j.

Combining the congestion score with the overlap penalty (as in [11]), the benefit for swapping each candidate cell j with i can be obtained. Based on this benefit function, we pick the best j and swap it with i. After the swap, we update the cell positions, rip-up and reroute the affected nets according to their new pin locations, and update the congestion map. This procedure involves a lot of routing task and is very time consuming. However, we are able to perform it due to the extremely fast global routing technique.

	IPR			ROOSTER + FR2.0			R-FastPlace + FR2.0			FastPlace + FR2.0		
	Overflow	R-WL	Time(s)	Overflow	R-WL	Time(s)	Overflow	R-WL	Time(s)	Overflow	R-WL	Time(s)
ibm01h	0	0.61	77	0	0.59	273	0	0.61	55	0	0.62	19
ibm02h	0	1.80	396	463	1.89	696	546	1.87	225	1309	1.90	37
ibm07h	369	3.73	945	736	3.92	1469	1125	3.78	464	3934	3.92	114
ibm08h	2	4.24	1275	97	4.37	2150	78	4.36	806	703	4.47	166
ibm09h	0	3.41	1028	2	3.35	1665	4	3.41	578	9	3.58	163
ibm10h	0	6.32	1685	11	6.59	2505	25	6.42	1270	31	6.57	200
ibm11h	1	4.96	1395	25	5.02	2139	22	4.98	978	42	5.28	178
ibm12h	1152	8.80	2105	1046	9.38	2901	2136	9.00	1718	5107	9.48	401
Total	1524	33.87	8906	2380	35.12	13798	3936	34.43	6094	11135	35.82	1279
Norm*	**1**	**1**	**1**	**1.562**	**1.036**	**1.550**	**2.580**	**1.016**	**0.682**	**7.306**	**1.058**	**0.143**

TABLE I
COMPARISON BETWEEN DIFFERENT FLOWS. (*) RESULTS IN THE LAST ROW ARE NORMALIZED TO IPR.

Note that in this approach, every move is intended for directly reducing routing congestion. This is achieved by keeping the routing solution and congestion map accurate. And it is the reason why we spend so much runtime on updating the routing solution.

In Figure 1, we apply *Routability-driven Global Swap* iteratively (lines 10-11). We run several iterations. In each iteration, we scan through all the cells to identify beneficial swaps. After *RGS*, the routing congestion is significantly reduced.

2) Routability-driven Local Swap (RLS): *Routability-driven Global Swap* is very effective in reducing routing congestion. However, because of a lot of rip-up and reroute operations, we cannot afford to run many iterations of it. Hence, we develop a *Routability-driven Local Swap (RLS)* technique to efficiently reduce routing congestion by moving cells locally.

The main idea for *RLS* is as follows. For each cell i, we just look at the cells directly adjacent to i, i.e., four neighbor cells of i. Then, we consider swapping i with them one by one. We also pick the one resulting the best benefit and swap it with i, similar to *RGS*. But in this technique, we only consider the swaps without creating any overlap. Otherwise, the swap will be neglected. Therefore, the benefit function is just the routing congestion reduction.

We have mentioned that the most time consuming part in *RGS* is to rip-up and reroute the nets being affected. In order to speed up *RLS*, we only update the routing tree branches of the nets being affected by the swap instead of updating the whole routing trees. (Note that a similar method is employed in the *RDR*.) Here, the basic assumption is that the routing tree topologies will not be affected by the swap. Since we only move cells very locally, the routing tree topologies will remain the same in most cases. For *RGS*, this assumption will not hold. Another reason for *RLS* to be much faster than *RGS* is that less candidate cells are considered for each cell. In *RLS*, only four candidate cells will be considered. However, in *RGS*, there could be hundreds of candidate cells.

IV. EXPERIMENTAL RESULTS

In this section, we present our experimental results. All experiments are performed on a Linux workstation with a 3.0 GHz Intel Pentium 4 CPU and 2GB memory.

We ran experiments on the IBMv2 suite of benchmarks. For the easy cases of the suite, most previous placers have already obtained 0 overflow. Therefore, we only show the experimental results on the hard cases in the suite. For the global routing grid, we use the same grid as defined in the LEF/DEF files. The routing grid size is 660×560. The original capacity of routing grid is 20 routing tracks for both horizontal direction and vertical direction. However, we use the capacity 16 for vertical direction and 14 for horizontal direction. The reason for this reduced routing capacity is to make the routing harder. Also, because part of Metal1 layer will be blocked by some routing inside standard cell, we decrease the horizontal routing capacity by 2 more tracks.

We ran four different flows to get placement and global routing results.

1) Integrated placement and routing approach (IPR)
2) ROOSTER followed by FastRoute2.0 (ROOSTER + FR2.0)
3) Routability-driven FastPlace followed by FastRoute2.0 (R-FastPlace + FR2.0)
4) FastPlace followed by FastRoute2.0 (FastPlace + FR2.0)

For Routability-driven FastPlace, we follow the framework of IPR. However, instead of applying global routing to get accurate routability information, we simply construct rectilinear Steiner minimal Steiner trees (RSMT) to break multi-pin nets into 2-pin nets and use probabilistic congestion model to get the congestion map. This is similar to many traditional routability-driven placement algorithms.

Except for IPR, which includes the global routing inside the flow, we need to first generate the placement and then run global routing on the placement for the other three flows. We apply FastRoute2.0 [12] on the placement generated by ROOSTER, FastPlace and R-FastPlace to obtain the global routing solutions.

The results are summarized in Table I. We measure the total overflow, global routing wirelength and runtime for the four different flows. From the results we can see that our integrated placement and routing approach is the best among the four. Compared with ROOSTER+FR2.0 flow, IPR reduces overflow by 36%, global routing wirelength by 3.6%, and runtime by 36%. Considering that ROOSTER is so far the best routability-driven placement algorithm, our integrated placement and routing approach is able to generate high-quality placement and routing solutions. For R-FastPlace, it has the same framework as IPR but uses a probabilistic congestion model. Comparison with FastPlace shows that the simple congestion-driven approach is effective in improving routability. However, by comparing with IPR, although they follow the same framework, IPR can generate much better solutions in terms of routability. This justifies the importance of accurate routing information in placement process. For FastPlace, it is a typical wirelength driven placement algorithm. It is obvious that the placement generated by it is poor in terms of both routability and global routing wirelength. This clearly shows that wirelength driven placement algorithm is not able to generate placement solutions with good routability.

V. CONCLUSIONS

In this paper, we propose a new integrated placement and routing algorithm to solve the placement routability problem. Experimental results show that the proposed integrated framework together with several routability driven techniques can efficiently generate high-quality placement and routing solutions.

Although we only focus on routability issue in this work, this framework can be extended to consider timing, buffering, and signal integrity because the routing information becomes available during the placement process. This paper suggests that the integrated placement routing approach has great potential to tackle other critical placement and routing problems.

REFERENCES

[1] X. Yang, B. K. Choi and M. Sarrafzadeh. Routability Driven White Space Allocation for Fixed-die Standard-cell Placement. In *Proc. Intl. Symp. on Physical Design*, pp. 42-49, 2002.

[2] C. Li, M. Xie, C. K. Koh, J. Cong and P. H. Madden. Routability-driven Placement and White Space Allocation. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pp. 394-401, 2004.

[3] N. Selvakkumaran, P. Parakh and G. Karypis. Perimeter-degree: A Priori Metric for Directly Measuring and Homogenizing Interconnection Complexity in Multilevel Placement. In *Proc. Intl. Workshop on System-Level Interconnect Prediction*, pp. 53-59, 2003.

[4] D. Jariwala and J. Lillis. On Interactions Between Routing and Detailed Placement. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pp. 387-393, 2004.

[5] J. A. Roy, J. F. Lu and I. L. Markov. Seeing the Forest and the Trees: Steiner Wirelength Optimization in Placement. In *Proc. Intl. Symp. on Physical Design*, pp. 78-85, 2006.

[6] M. Pan and C. Chu. FastRoute: A step to integrate global routing into placement. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, 2006.

[7] N. Viswanathan and Chris Chu. FastPlace: Efficient Analytical Placement using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model. In *Proc. Intl. Symp. on Physical Design*, pp. 26-33, 2004.

[8] Chris Chu. FLUTE: Fast Lookup Table Based Wirelength Estimation Technique. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, 696-701, 2004.

[9] Chris Chu and Yiu-Chung Wong. Fast and Accurate Rectilinear Steiner Minimal Tree Algorithm for VLSI Design. In *Proc. Intl. Symp. on Physical Design*, pp. 28-35, 2005.

[10] N. Viswanathan, M. Pan and Chris Chu. FastPlace 2.0: An Efficient Analytical Placer for Mixed-Mode Designs. In *Asia and South Pacific Design Autom. Conf.*, pp. 195-200, 2006.

[11] M. Pan, N. Viswanathan and Chris Chu. An Efficient and Effective Detailed Placement Algorithm. In *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design*, pp. 48-55, 2005.

[12] M. Pan and C. Chu. FastRoute 2.0: A High-quality and Efficient Global Router. To appear. *Asia and South Pacific Design Autom. Conf.*, 2007.