

# Ripple: An Effective Routability-Driven Placer by Iterative Cell Movement

Xu He, Tao Huang, Linfu Xiao, Haitong Tian, Guxin Cui and Evangeline F.Y. Young  
Department of Computer Science and Engineering  
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong  
email: {xhe, thuang, lfxiao, httian, gxcui, fyyoung}@cse.cuhk.edu.hk

## ABSTRACT

In this paper, we describe a routability-driven placer called *Ripple*. Two major techniques called *cell inflation* and *net-based movement* are used in global placement followed by a rough legalization step to reduce congestion. Cell inflation is performed in the horizontal and the vertical directions alternatively. We propose a new method called net-based movement, in which a target position is calculated for each cell by considering the movement of a net as a whole instead of working on each cell individually. In detailed placement, we use a combination of two kinds of strategy: the traditional HPWL-driven approach and our new congestion-driven approach. Experimental results show that *Ripple* is very effective in improving routability. Comparing with our previous placer, which is the winner in the ISPD 2011 Contest, *Ripple* can further improve the overflow by 38% while reduce the runtime is reduced by 54%.

## 1. INTRODUCTION

Routability has become an important issue in VLSI physical design. Placement, which plays a fundamental and important role in physical design, can greatly affect the quality of the routing solution and is perhaps the most powerful tool in the congestion alleviation [1].

In previous placement algorithms, the major objective is to minimize wirelength which is often estimated by the half-perimeter wirelength (HPWL) model. Although minimizing HPWL can reduce the average routing demand, the routing demand may be distributed unevenly and as a result, some nets are difficult to be routed or even unroutable at the end. Moreover, hundreds of large macros which occupy several metal layers usually exist in modern circuits. The existence of such routing blockages have made this routability-driven placement problem even more challenging.

There are some previous works on this routability-driven placement problem. Several of them try to distribute the routing demand evenly by allocating whitespace in the congested regions explicitly or implicitly [14][8][2][10][4]. However, the congestion maps in the works [2][4] may be inaccurate during their top-down partitioning-based global placement, since all the cells are still far from being legalized. Therefore, it is difficult to calculate the amount of whitespace needed based on the inaccurate congestion maps. The works [14][8][10] adjust the whitespace distribution after global placement, but this is not flexible enough since the global placement result cannot be changed too much. Some previous works try to change the objective function in analytical placement [6][5][3][11] to address the routability issue. However, the function modeling the routing demand information directly is usually nonlinear [6][5][3]. This kind of nonlinear analytical

placement is often based on a multilevel framework for efficiency, but accurate congestion estimations of those intermediate-level layouts in the multilevel framework will be hard to obtain. The work [11] uses an additional move force in their quadratic function to move cells to regions with lower routing demand, but this approach lags behind in solution quality [7].

In this paper, our routability-driven placer *Ripple* is based on a lower-upper-bound framework described in [7] during global placement. Since *Ripple* is a flat placer and congestion estimation will always be done after a rough legalization step, the estimation obtained is much more accurate. In the lower-bound computation, the objective is to minimize the HPWL using Bound2Bound (B2B) net model [12], in order to reduce the total routing demand and wirelength. In the upper-bound computation, after congestion estimation, we will use two major techniques, called *cell inflation* and *net-based movement* followed by rough legalization to distribute the congestion measures evenly. Since the cell inflation step will enlarge the cell areas in the congested regions, the cells will be placed further apart after the rough legalization step. The net-based movement method will try to move a whole net directly according to the congestion estimation, and is an effective method to be used with cell inflation. For detailed placement, we use a combination of two strategies: the traditional HPWL-driven approach, and a new congestion-driven approach that considers the displacement between the cells and their target positions computed to reduce congestion.

We summarize our major contributions as follows:

- Our cell inflation method considers the inflation ratio and the cell spreading scope as a whole in the horizontal and vertical directions separately. We find that by doing cell inflation in alternate directions iteratively, the total congestion can be reduced more effectively.
- We devise an approach called net-based movement to calculate the target positions for the cells by considering congestion and the movement of a whole net directly.
- In detailed placement, we propose a congestion-driven approach. Besides minimizing HPWL, we also consider the displacement between the cells and their target positions calculated to optimize congestion.

The remainder of this paper is organized as follows. In Section 2, we give the problem formulation. Section 3 is an overview of the whole process of our placer. Section 4 explains the congestion estimation method to compute congestion maps quickly. Section 5 gives details of the cell inflation and net-based movement steps. Section 6 describes the detailed placement phase.

Table 1: Major information for global router

| Parameter         | Description   |
|-------------------|---|
| LayerNum          | Number of layers  |
| VTrack( $i$ )     | Number of vertical or horizontal tracks on layer $i$ in a tile ( $i = 1, \dots, LayerNum$ ) |
| HTrack( $i$ )     | Number of vertical or horizontal tracks on layer $i$ in a tile ( $i = 1, \dots, LayerNum$ ) |
| TileSizeX/Y       | Width/Height of a tile  |
| BlockPorosity     | Porosity of the routing blockages: zero implies complete blockage                           |
| BlockLayer( $i$ ) | The set of blocked layers of routing blockage $i$   |

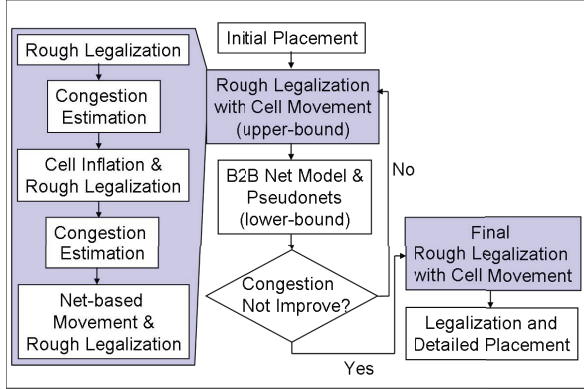


Figure 1: Flow of our placement algorithm

Experimental results and a conclusion will be given in Section 7 and Section 8.

## 2. PROBLEM FORMULATION

As defined in the ISPD Contest 2011 [13], given a netlist  $N = (E, V)$  with nets  $E$  and nodes  $V$ , a routability-driven placer is required to calculate the positions of the movable nodes (cells) to minimize the total routing overflow and wirelength when the placement is routed by a given router within a limited amount of time. The major objective is to minimize the overflow obtained by the router. The routing information is given as well. We list the major parameters in Table 1.

## 3. OVERVIEW OF OUR ALGORITHM

Our placer is developed based on the lower-upper-bound framework in [7]. The two bound computations are invoked alternatively. Figure 1 shows the whole process of our placer. In the upper-bound computation phase, both cell inflation and net-based movement will be used after congestion estimation and followed by a rough legalization. Notice that our congestion estimation is always performed after a rough legalization step, so it will be more accurate and useful. We use cell inflation to reduce congestion in the horizontal and vertical directions alternatively and use net-based movement to reduce congestion in the direction with higher overflow. Both the cell inflation and net-based movement steps are incorporated into the flow after a certain number of iterations ( $5^{th}$  and  $15^{th}$  for cell inflation and net-based movement respectively in our implementation). The total number of iterations in the global placement phase is around 32 to 40, regardless of the benchmark size. We found that with even more iterations will only place the cells more closely and will cause routability problem.

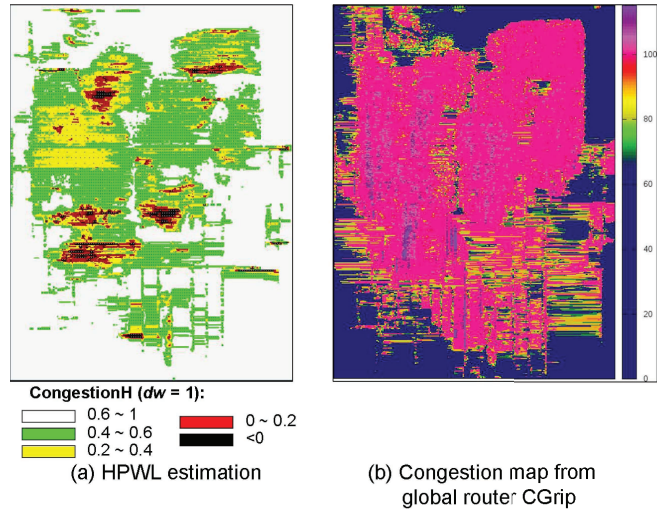


Figure 2: The congestion estimation of HPWL is too optimistic. Both congestion maps are for the horizontal direction (superblue18).

For detailed placement, we use a combination of two kinds of strategies. The first one is the traditional HPWL-driven approach [9]. Although only focusing on HPWL can reduce the routing demand, it may make the placement less routable in some cases. The other one is congestion-driven, in which we try to reduce the displacement between the cells and their target positions calculated to optimize congestion. At the end, we will return the one with less overflow.

## 4. ROUTING CONGESTION ESTIMATION

Our congestion estimation is obtained by extending and generalizing RUDY [11]. RUDY uses the HPWL estimation directly which is indeed much shorter than the actual wirelength. Therefore, it will often under-estimate the real routing congestion a lot as shown in Figure 2(a) as compared with Figure 2(b) that is obtained from the global router CGrip. To cope with this, instead of using HPWL directly, multi-pin nets are first decomposed into two-pin nets using the rectilinear minimum spanning tree (RMST) method. Shared edges between subnets of the same net will be handled in such a way to avoid double counting. The routing demand is computed in the horizontal and vertical directions separately. We will discuss below the computation of the horizontal demand  $DemandH$ , while the vertical demand  $DemandV$  can be obtained similarly.

The layout is divided into uniform routing tiles whose width and height are  $TileSizeX$  and  $TileSizeY$  respectively. Consider a two-pin net  $p$  connecting from  $(x_1, y_1)$  to  $(x_2, y_2)$ , the horizontal wirelength is the width of the bounding box. The  $DemandH$  due to  $p$  for each tile overlapping with the bounding box of  $p$  is computed as follows:

$$\begin{aligned}
 DemandH &= \frac{ConnInterArea}{BoundingBoxArea} \times WireAreaH \\
 WireAreaH &= (\max\{x_1, x_2\} - \min\{x_1, x_2\}) \times WireSpaceH \\
 WireSpaceH &= \frac{TileSizeY}{TotalTracksH} \\
 TotalTracksH &= \sum_{i=1}^{LayerNum} HTrack(i)
 \end{aligned} \tag{1}$$

where  $BoundingBoxArea$  is the bounding box area of this con-

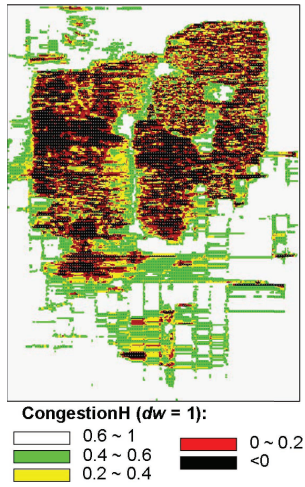


Figure 3: The congestion estimation maps of the horizontal direction (superblue18).

nection,  $ConnInterArea$  is the intersection area between the tile and the bounding box,  $WireSpaceH$  is the area of one unit of wire in the horizontal direction,  $WireAreaH$  is the horizontal wire area of this net which can be obtained by multiplying the horizontal wirelength by  $WireSpaceH$  and  $TotalTracksH$  is the total number of horizontal tracks (counting all metal layers) in the tile.

The supply of a tile in the horizontal direction  $SupplyH$  can be computed as follows:

$$\begin{aligned} SupplyH &= TileSizeX \times TileSizeY - BlockageH \\ BlockageH &= \sum_{i \in Btile} BInterArea(i) \times BRatioH(i) \\ BRatioH(i) &= \frac{\sum_{j \in BlockLayer(i)} HTrack(j) \times (1 - BlockPorosity)}{TotalTracksH} \end{aligned} \quad (2)$$

where  $Btile$  is the set of routing blockages overlapped with the tile,  $BInterArea(i)$  is the intersection area between routing blockage  $i$  and the tile,  $BRatioH(i)$  is the ratio of horizontal tracks being occupied by the routing blockage  $i$  and  $BlockageH$  is the routing resource occupied by all the routing blockages overlapped with the tile in the horizontal direction. The level of horizontal routing congestion  $CongestionH$  of the tile can then be computed as:

$$CongestionH = \frac{SupplyH - DemandH \times dw}{SupplyH} \quad (3)$$

where  $dw$  (demand weight,  $dw > 0$ ) is a variable defined to control the number of tiles to be inflated. The less the value of  $CongestionH$ , the more congested a tile is. If  $CongestionH < 0$ , the tile is said to be over congested.

As mentioned above, we will handle the shared edges of the subnets of the same net carefully in order not to double count some congestion estimation mistakenly. As illustrated by the example in Figure 4(a), using RMST, this three-pin net will be decomposed into two 2-pin subnets which are  $a$  to  $b$  and  $a$  to  $c$ . Both subnets will increase the routing demand of tile 1 and the demand of this net on tile 1 will be counted twice. In order to estimate more accurately, the  $demandH$  of this tile due to this net will be equal to the maximum  $demandH$  of those subnets. Figure 3 shows the congestion map in the horizontal direction for superblue18 obtained by our method. We can see that more

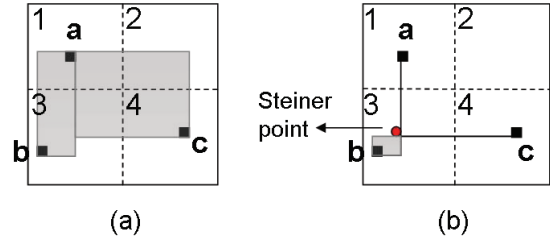


Figure 4: An example to show the RMST model will have larger density estimation than RSMT model.

accurate estimation can be obtained by comparing Figure 3 with the simple HPWL approach shown in Figure 2 (a).

## 5. CELL MOVEMENT

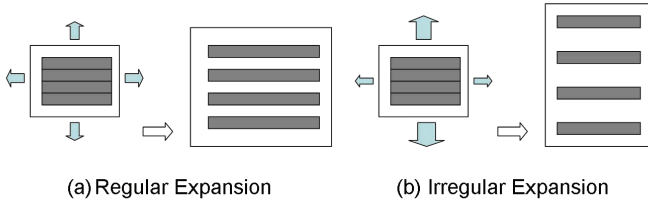
### 5.1 Cell Inflation and Spreading

In cell inflation, we need to address three important issues. Where should cell inflation be used? What is the inflation ratio? How to spread the cells? Unlike previous works [2][4][10], we consider these three issues as a whole for the horizontal and vertical directions separately. By applying our cell inflation in the horizontal and vertical directions alternatively for several times (4 times for each direction in our implementation) in each iteration of global placement, the routing densities in both directions will be alleviated effectively. The following discussion is based on the horizontal direction while the vertical direction can be handled similarly.

The cells are inflated according to the values of  $CongestionH$  of the tiles (Section 4). In previous works, if a tile's routing supply is less than its routing demand (i.e.,  $CongestionH < 0$ ), the cells in this tile will be inflated. However, if almost all the tiles are congested, inflating the cells in all the tiles is helpless to distribute the routing demand evenly all over the chip. Therefore, we adjust the variable  $dw$  in equation 3 to control the number of tiles to be inflated. If too many tiles are over-congested, the value of  $dw$  can be reduced to decrease the weighted routing demand in equation 3. The number of congested tiles whose  $CongestionH$  is smaller than zero will thus decrease. Besides, we use three levels of inflation ratio as shown in equation 4 where  $a_1 = 0.3, b_1 = 0.4, c_1 = 0.6$  and  $a_2 = 0, b_2 = -0.2, c_2 = -0.4$ .

$$InflationRatio = \begin{cases} a_1 & \text{if } CongestionH < a_2 \\ b_1 & \text{if } CongestionH < b_2 \\ c_1 & \text{if } CongestionH < c_2 \end{cases} \quad (4)$$

After cell inflation, we will spread the cells in those tiles whose cell density is larger than a certain threshold  $\gamma$  ( $0 \leq \gamma \leq 1$ ). A *modified* rough legalization will be performed to reduce cell density. In a regular rough legalization step, the clusters of cells with cell density larger than  $\gamma$  will be expanded in all four directions (left, down, right, up) until the cell density of the region is below  $\gamma$ . In this modified legalization step, if we focus on reducing the congestion in the horizontal direction, the expansion will go more in the vertical direction (up and down). In that case, the horizontal demand will be distributed more sparsely in a larger vertical scope. As shown in Figure 5, if the congested region is expand more in the vertical direction, the horizontal congestion will be distributed more sparsely. After applying cell



**Figure 5: An example to show that the irregular expansion strategy is more effective in reducing congestion in one direction.**

inflation in the horizontal and vertical directions alternatively, the routing densities in both directions will be alleviated effectively. All the cells will then be changed back to their original sizes after the last rough legalization of the iteration.

We also limit the total inflation area of the cells for each cell inflation process. A threshold  $\delta$  is set to limit the total inflation area. The tiles with  $CongestionH < 0$  will be sorted in an increasing order of their  $CongestionH$  value, and the cells in the most congested tiles are inflated first until the total inflation area reaches  $\delta$ . In our implementation, we set  $\delta$  as 35% of the total available area.

## 5.2 Net-Based Movement

In cell inflation, cells over congested regions will be expanded and spread to the surrounding. However this approach does not identify target positions for the cells constructively, and it works on each cell individually without considering how one whole net should be moved according to the congestion. To solve these problems, we devise another effective method called net-based movement which will compute a target position for each cell and will consider the movement of a net as a whole.

Our method includes two steps: (1) move the nets' bounding boxes away from the congested regions, since it is the nets instead of the cells using the routing resource directly; (2) move the cells on the nets according to the target positions of the nets. The following discussion is on reducing the congestion measures in the horizontal direction while the vertical direction can be handled similarly.

In order to move a net away from regions with high horizontal congestions, we can change the  $x$  or  $y$  coordinate of its bounding box center. We choose to change the  $y$ -coordinate only for horizontal congestion reduction. It is because if the  $x$ -coordinate is not changed, the width of the bounding box of each net is not modified and the horizontal demand will not be worsened. This strategy can reduce the horizontal congestion in a more stable manner. We use  $NSet(i)$  to denote the set of cells on net  $i$ . Let the number of cells in both net  $i$  and net  $j$  be  $Share(i, j) = |NSet(i) \cap NSet(j)|$ . Nets  $i$  and  $j$  are said to be associated with each other if  $Share(i, j) > 0$ . The larger the value of  $Share(i, j)$ , the closer is the relationship between  $i$  and  $j$ .

This method is outlined in Algorithm 1. On line 1-4,  $CenterY(i)$  and  $CH(i)$  are calculated for net  $i = 1, 2, \dots, n$ , where  $n$  is the number of nets,  $CenterY(i)$  is the original  $y$ -coordinate of the center of net  $i$  and  $CH(i)$  is the average  $CongestionH$  of all the tiles overlapped by the bounding box of net  $i$ . Line 7-17 shows that if  $CH(i) < 0$ , net  $i$  will be moved nearer to those associated nets  $j$  where  $CH(j) > CH(i)$ . We first find the  $BestCH$  which is the largest  $CH(j)$  for all  $j$  such that  $Share(i, j) > 0$  as shown in line 8-13. If  $BestCH < 0$ , net  $i$  will not be moved; otherwise,

we will move net  $i$  according to equation 5:

$$NewCenterY(i) = CenterY(i) + \sum_{\substack{Share(i,j) > 0 \\ \&CH(j) > CH(i)}} Move(i, j)$$

$$Move(i, j) = (CenterY(j) - CenterY(i)) \times \frac{CH(j) - CH(i)}{BestCH - CH(i)} \times \frac{Share(i, j)}{|NSet(i)|} \quad (5)$$

After getting the new centers  $NewCenterY(i)$  of all the nets, each movable cell  $q = 1, 2, \dots, m$  will be moved in the  $y$ -direction according to the new centers of the nets they are on. Suppose the set of associated nets of cell  $q$  is  $Q$ . For each net  $i \in Q$ , we calculate a candidate position  $CellY(i)$  for  $q$  according to the new center  $NewCenterY(i)$  by keeping the same relative position between the cell  $q$  and the center of the net's bounding box (line 23-24). Now, cell  $q$  has a set of candidate positions according to the set of associated nets it is on. The range of all candidate positions is  $[MinY, MaxY]$ , where  $MinY = \min_{i \in Q} \{CellY(i)\}$  and  $MaxY = \max_{i \in Q} \{CellY(i)\}$  (line 25-29). Let  $MidY = (MinY + MaxY)/2$ . We will then move  $q$  with the smallest displacement into the range  $[CellY(i_1), CellY(i_2)]$ , where  $CellY(i_1)$  and  $CellY(i_2)$  are the two candidate positions closest to  $MidY$  and  $CellY(i_1) \leq MidY \leq CellY(i_2)$  (line 31-33).

---

### Algorithm 1 Net-Based Movement

---

```

1: for net  $i=1$  to  $n$  do
2:   Compute  $CenterY(i)$  of the bounding box of net  $i$ 
3:   Compute the average horizontal congestion  $CH(i)$ 
4: end for
5: for net  $i=1$  to  $n$  do
6:    $NewCenterY(i) = CenterY(i)$ 
7:   if  $CH(i) < 0$  then
8:      $BestCH = CH(i)$ 
9:     for each net  $j$  where  $Share(i, j) > 0$  do
10:      if  $BestCH < CH(j)$  then
11:         $BestCH = CH(j)$ 
12:      end if
13:    end for
14:    if  $BestCH > 0$  and  $BestCH \neq CH(i)$  then
15:      Update  $NewCenterY(i)$  using equation 5
16:    end if
17:  end if
18: end for
19: for cell  $q=1$  to  $m$  do
20:    $MinY = +\infty, MaxY = 0$ 
21:   Let  $Q$  be the set of associated nets of cell  $q$ 
22:   for each connected net  $i \in Q$  do
23:      $OffsetY = CellCoordY(q) - CenterY(i)$ 
24:      $CellY(i) = OffsetY + NewCenterY(i)$ 
25:     if  $MinY > CellY(i)$  then
26:        $MinY = CellY(i)$ 
27:     else if  $MaxY < CellY(i)$  then
28:        $MaxY = CellY(i)$ 
29:     end if
30:   end for
31:    $MidY = (MinY + MaxY)/2$ 
32:   Find  $i_1, i_2 \in Q$ , such that  $CellY(i_1)$  and  $CellY(i_2)$  are
   closest to  $MidY$ , and  $CellY(i_1) \leq MidY \leq CellY(i_2)$ 
33:   Move  $CellCoordY(q)$  to the closest position in the range
    $[CellY(i_1), CellY(i_2)]$ 
34: end for

```

---

After changing the positions of the cells, we will call the modified rough legalization step as described in section 5.1.

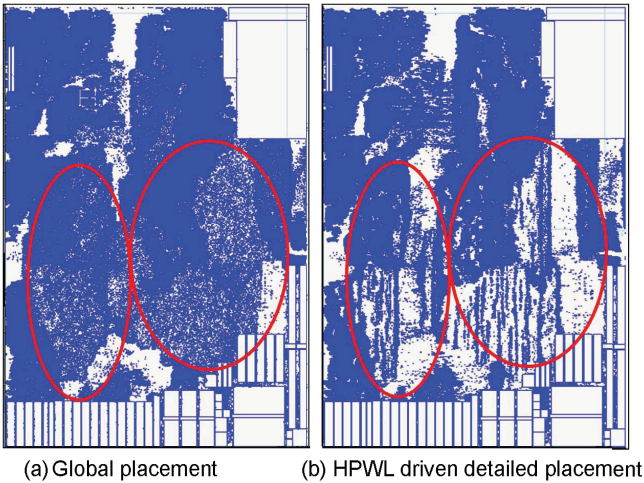


Figure 6: The cells of superblue12 are placed densely in HPWL-driven detailed placement. The regions highlighted in red rings have high routing congestion.

## 6. DETAILED PLACEMENT

We use FastPlace-DP [9] for our detailed placement. However, FastPlace-DP is HPWL-driven and it may increase congestion and reduce routability. As shown in Figure 6(a), after global placement, the cells are located quite sparsely in the potentially over-congested regions. However, FastPlace-DP will try to move cells together (Figure 6(b)) and will very likely increase the overflow of the global routing step. Therefore, besides using the HPWL-driven approach, we will also use a congestion-driven approach. At the end, we will return the result with less overflow.

In the congestion-driven detailed placement, the displacement between the cell and global placement is considered. We have two steps in the congestion-driven approach. In the first step, we make use of the result from global placement. When swapping cells to minimize HPWL, the swapping window is around the target position which is the position obtained by the global placement. If the HPWL is not improved after a number of swapping iterations, this step is finished. In the second step, the target position is calculated by the net-based movement method. When swapping cells to minimize HPWL, the swapping window is around the target position. After several swapping iterations without congestion improvement, the detailed placement will be finished.

## 7. EXPERIMENTAL RESULTS

Our implementation is written in C++ and compiled with g++ 4.1.2. All the benchmarks are run on an Intel Core 2 Duo Linux workstation with 2.8GHz and 4 GB RAM, using one CPU core. We use the global router CGrip with the same configuration as in the ISPD 2011 contest to evaluate our placement result.

### 7.1 Analysis of Our Placement Method

The global placement phase mainly consists of five parts: Conjugate Gradient (CG), rough legalization, congestion estimation, cell inflation and net-based movement. About 42.9%, 28.5% and 25.4% of the total runtime of the global placement phase is spent on congestion estimation, CG and rough legalization respective-

Table 2: Runtime for global and detailed placement

| Benchmark   | Global (s) | Detailed (s) |            |
|-------------|------------|--------------|------------|
|             |            | HPWL         | Congestion |
| superblue18 | 2558       | 146          | 1707       |
| superblue4  | 4752       | 332          | 1682       |
| superblue5  | 4626       | 405          | 1783       |
| superblue1  | 6525       | 544          | 2669       |
| superblue2  | 6182       | 362          | 2416       |
| superblue15 | 8797       | 277          | 1454       |
| superblue10 | 8321       | 420          | 3573       |
| superblue12 | 7652       | 991          | 2666       |
| Avg.        | -          | 1.00         | 5.16       |

Table 3: Overflow of detailed placement

| Benchmark   | HPWL    | Congestion |
|-------------|---------|------------|
| superblue18 | 60650   | 38176      |
| superblue4  | 75540   | 182962     |
| superblue5  | 128054  | 347462     |
| superblue1  | 90      | 410        |
| superblue2  | 936070  | 730544     |
| superblue15 | 60222   | 153566     |
| superblue10 | 646300  | 722588     |
| superblue12 | 1104162 | 547954     |
| Avg.        | 1.11    | 1.00       |

ly. The processes of cell inflation and net-based movement take only 0.4% and 2.7% of the total runtime of the global placement phase respectively.

Figure 7 shows the lower and upper bounds for HPWL, the inflation area and the sum of congestion estimations in both the horizontal and the vertical directions at each iteration for the benchmark superblue18. Since cell inflation is used after the 5th iteration, the HPWL upper-bound is dramatically increased at the 5th iteration, and then begin to decrease again. The congestion value is reduced greatly in the first 25 iterations, and then it is improved slowly. When the congestion is decreased iteratively, the inflation area is reduced gradually as well.

Table 2 shows the runtime for the global and detailed placement in seconds. We list the runtime of both HPWL-driven (HPWL) detailed placement, i.e., FastPlace-DP [9], and our congestion-driven (Congestion) detailed placement (Section 6).

We also use CGrip to calculate the overflow of the placement results from the two detailed placement approaches, as listed in Table 3. We can see that the HPWL-driven detailed placement can get a little bit smaller overflow in some benchmarks. That is because based on the result of global placement, reducing HPWL can reduce the routing demand. However, the total overflow of the HPWL-driven approach is 11% more than that of the congestion-driven approach. In superblue12 and superblue2, the congestion-driven detailed placement has much smaller overflow. It is because the HPWL-driven detailed placement changes the positions of the cells too much (as shown in Figure 6), making the congestion increase dramatically. Therefore, in order to combine the advantages of both methods, we use HPWL-driven detailed placement first, then we will check whether the displacement between the detailed placement result and the global placement result exceeds a certain threshold. If the displacement is too large, we will call our congestion-driven detailed placement and return the result with less overflow.

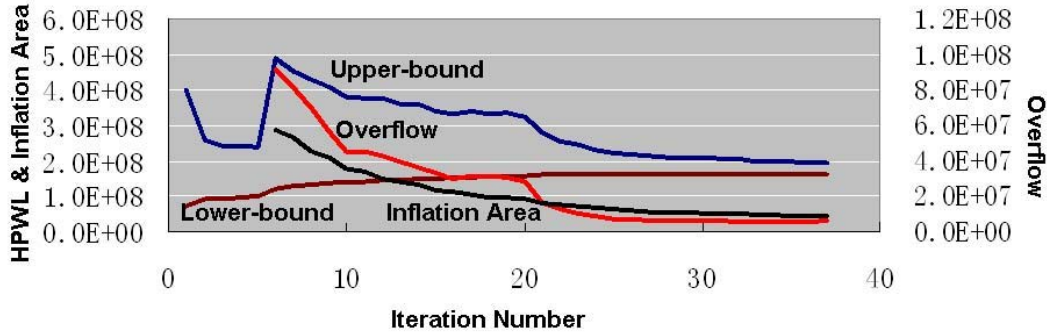


Figure 7: Lower-upper-bound HPWL, inflation area and overflow at each iterations (superblue18)

Table 4: Comparisons with the results of the ISPD 2011 Contest

| Benchmark   | RippleContest2011 |                   | mPL11        |                   | SimPLR        |                   | RADIANT       |                   | Ripple        |                   |
|-------------|-------------------|-------------------|--------------|-------------------|---------------|-------------------|---------------|-------------------|---------------|-------------------|
|             | Overflow          | Runtime (seconds) | Overflow     | Runtime (seconds) | Overflow      | Runtime (seconds) | Overflow      | Runtime (seconds) | Overflow      | Runtime (seconds) |
| superblue18 | 460910            | 11226             | <b>25134</b> | 2413              | 50064         | 2791              | 364820        | 3718              | 38176         | 6662              |
| superblue4  | 85526             | 7631              | 122842       | 2560              | 420738        | 3083              | 226092        | 3019              | <b>75540</b>  | 5094              |
| superblue5  | 147176            | 10885             | 368940       | 3818              | 191704        | 5649              | 613390        | 6170              | <b>128054</b> | 9641              |
| superblue1  | 108               | 9915              | 34660        | 3573              | <b>0</b>      | 5686              | 122720        | 6895              | 90            | 7079              |
| superblue2  | 797898            | 16887             | 1403738      | 4318              | 1659322       | 7913              | 1454226       | 8611              | <b>730544</b> | 15958             |
| superblue15 | 143580            | 17831             | 140208       | 5138              | 283056        | 7697              | 695684        | 12580             | <b>60222</b>  | 9083              |
| superblue10 | 1010058           | 22319             | 1159416      | 5046              | 1311688       | 12989             | <b>616424</b> | 12873             | 646300        | 8751              |
| superblue12 | 437676            | 21390             | 2201266      | 6971              | <b>415404</b> | 10046             | 3051096       | 19089             | 547954        | 14186             |
| Avg.        | 1.38              | 3.49              | 2.45         | 1.0               | 1.95          | 1.65              | 3.21          | 2.16              | 1.00          | 2.26              |

## 7.2 Comparisons with the Results of the ISPD 2011 Contest

Table 4 lists the overflow result of the placements obtained by the ISPD 2011 contest participants and *Ripple*. We run the router CGrip on the released placement files on our workstation, and the overflow is reported by the contest evaluation script. We can see that the total overflow is reduced by 38% compared with our previous version, the first place in the ISPD 2011 Contest (RippleContest2011 in Table 4), and we can get the minimum overflow in most benchmarks. Although our overflow is not the best in some benchmarks, our results are very close to the minimum ones. The total runtime of our method is improved by 54% compared with RippleContest2011. The benchmark superblue1, superblue4, superblue10 and superblue15 finished after HPWL-driven detailed placement since the displacement between the detailed placement result and the global placement result is small. Other benchmarks use both kinds of detailed placement, and the global router CGrip is used to evaluate the results from the two detailed placement approaches. The runtime of the global router is included in the total runtime.

## 8. CONCLUSION

Our routability-driven placer is based on the lower-upper-bound framework for the global placement step. The lower-bound computation is to reduce the total routing demand by minimizing HPWL. The upper-bound computation use *cell inflation* and *net-based movement* to reduce congestion. For detailed placement, we use a combination of two strategies: the traditional HPWL-driven approach and our congestion-driven approach. Comparing with our previous placer which is the winner in the ISPD 2011 Contest, *Ripple* can further improve

the overflow by 38% while the runtime is reduced by 54%.

## 9. REFERENCES

- [1] C. Alpert, Z. Li, M. Moffitt, G. Nam, J. Roy, and G. Tellez. What makes a design difficult to route. In *ISPD*, pages 7–12, 2010.
- [2] U. Brenner and A. Rohe. An effective congestion-driven placement framework. *TCAD*, 22(4):387–394, 2003.
- [3] Y. Chuang, G. Nam, C. Alpert, Y. Chang, J. Roy, and N. Viswanathan. Design-hierarchy aware mixed-size placement for routability optimization. In *ICCAD*, pages 663–668, 2010.
- [4] W. Hou, H. Yu, X. Hong, Y. Cai, W. Wu, J. Gu, and W. Kao. A new congestion-driven placement algorithm based on cell inflation. In *ASP-DAC*, pages 605–608, 2001.
- [5] Z. Jiang, B. Su, and Y. Chang. Routability-driven analytical placement by net overlapping removal for large-scale mixed-size designs. In *DAC*, pages 167–172, 2008.
- [6] A. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. *TCAD*, 24(5):734–747, 2005.
- [7] M. Kim, D. Lee, and I. Markov. SimPL: An Effective Placement Algorithm. In *ICCAD*, pages 649–656, 2010.
- [8] C. Li, M. Xie, C. Koh, J. Cong, and P. Madden. Routability-driven placement and white space allocation. *TCAD*, 26(5):858–871, 2007.
- [9] M. Pan, N. Viswanathan, and C. Chu. An efficient and effective detailed placement algorithm. In *ICCAD*, pages 48–55, 2005.
- [10] J. Roy, N. Viswanathan, G. Nam, C. Alpert, and I. Markov. CRISP: congestion reduction by iterated spreading during placement. In *ICCAD*, pages 357–362, 2009.
- [11] P. Spindler and F. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *DATE*, pages 1226–1231, 2007.
- [12] P. Spindler, U. Schlichtmann, and F. Johannes. Kraftwerk2: A fast force-directed quadratic placement approach using an accurate net model. *TCAD*, 27(8):1398–1411, 2008.
- [13] N. Viswanathan, C. Alpert, C. Sze, Z. Li, G. Nam, and J. Roy. The ISPD-2011 routability-driven placement contest and benchmark suite. In *ISPD*, pages 141–146, 2011.
- [14] X. Yang, B. Choi, and M. Sarrafzadeh. Routability-driven white space allocation for fixed-die standard-cell placement. *TCAD*, 22(4):410–419, 2003.