Diffusion-Based Placement Migration with Application on Legalization

Haoxing Ren, *Member, IEEE*, David Z. Pan, *Senior Member, IEEE*, Charles J. Alpert, *Fellow, IEEE*, Paul G. Villarrubia and Gi-Joon Nam, *Member, IEEE*

***Abstract*— Placement migration is the movement of cells within an existing placement to address a variety of post-placement design closure issues, such as timing, routing congestion, signal integrity, and heat distribution. To fix a design problem, one would like to perturb the design as little as possible while preserving the integrity of the original placement. This work presents a new diffusion-based placement method based on a discrete approximation to the closed-form solution of the continuous diffusion equation. It has the advantage of smooth spreading, which helps preserve neighborhood characteristics of the original placement. Applying this technique to placement legalization demonstrates significant improvements in wire length and timing compared to other commonly used techniques.**

I. Introduction

During placement and physical synthesis of VLSI circuits, one is commonly faced with tasks such as cell spreading, legalization of overlapping cells, and manipulation of the placement to address objectives like power and routing congestion. These tasks share a common theme of starting with an initial placement that is "good" and perturbing it so that it is improved in some way while still preserving the essential characteristics (cell ordering, wirelength, etc.) of the original placement. We call these sets of tasks "placement migration". Some specific examples of placement migration include the following:

- During physical synthesis, one may insert buffers and repower gates, thereby creating overlapping cells. The new instance needs to be legalized, but one wants to avoid moving any cell too far away from its original location.
- After placement, it may be necessary to make Engineering Change Orders (ECO) or insert decoupling capacitors which requires spreading to resolve induced overlaps.
- Post routing congestion analysis may identify several hot spots of congestion or crosstalk noise. Placement migration can locally spread out cells in these congested or noisy regions [1].
- A global analytic or force-directed placer may use placement migration to spread out the cells while attempting to preserve the ordering induced by the overlapping analytic solution.

This work presents a new technique for placement migration based on the physical process of diffusion. Diffusion is a well-understood physical process that moves elements (such as dopants) from a state with non-zero potential energy to a state of equilibrium. The process can be modeled by breaking down the movements into several small finite time steps, then moving each element the distance it would be expected to move during that time step. Our approach to placement migration follows this model; it moves each cell a small amount in a given time step according to its local density gradient. The more time steps the process is run, the closer the placement gets towards achieving equilibrium. The primary advantage to this approach is that it spreads the placement *smoothly* which is more likely to preserve the integrity of the original placement.

Among the various placement migration applications, legalization is perhaps the most straightforward. Therefore, the remainder of the paper will discuss diffusion in this context. The paper is organized as follows. Section II formulates the legalization problem and reviews previous techniques. Section III describes the mathematical formulation for diffusion in the continuous space. Section IV presents the numerical algorithm required to simulate the diffusion process. Section V gives the diffusion-based legalization algorithm. Section VI introduces a robust local diffusion algorithm which utilizes *local* diffusion and dynamic density update to achieve better quality results than the original diffusion algorithm and runs faster. It only "diffuse" cells as necessary to make a placement legal. Experimental results are shown in section VII, followed by the conclusion in section VIII. A preliminary version of this paper was presented in DAC 2005 [2].

II. Legalization Formulation and Overview

A placement is illegal if cells overlap or are not aligned with circuit rows. The term "legalization" describes the problem of taking an illegal placement and perturbing the layout so that it is legal. The objective of legalization is for this perturbation to be minimal in order to preserve the desired characteristics of the given illegal placement.

This work is supported in part by SRC under contract 2005-TJ-1321, IBM Faculty Award and IBM Degree Work Study Program.

H. Ren is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, email: haoxing@us.ibm.com

D. Z. Pan is with ECE Department, University of Texas at Austin, Austin, TX 78712, email: dpan@ece.utexas.edu

C. J. Alpert is with IBM Austin Research Laboratory, Austin, TX 78758, email: alpert@us.ibm.com

P. Villarrubia is with IBM Corporation, Austin, TX 78759, emails: pgvillar@us.ibm.com

G. Nam is with IBM Austin Research Laboratory, Austin, TX 78758, email: gnam@us.ibm.com

A. Formulation

A placement is close to legal if all that is required to legalize the placement is to snap cells to rows or perhaps perform minor cell sliding in order to fit the cells. Assume the chip layout is divided into small, equal sized bins (which can fit around 5-15 cells for example). Let d_{max} be the maximum allowed density of a bin, where commonly $d_{max} = 1$. The placement is considered close to legal if the area density of every bin is less than or equal to d_{max}. For all bins with density greater than d_{max}, cells must be migrated out of those bins into less dense ones. The goal of legalization is to reduce the density of each bin to no more than d_{max} while avoiding moving these cells far from their original locations and also to preserve the ordering induced by the original placement.

B. Existing Legalization Techniques

Existing legalization techniques include network flow [3], heuristic ripple cell movement [4] , dynamic programming [5], single row optimization [6] [7], packing [8] and other heuristics [9] [10] [11] [12] [13].

The network flow approach [3] models the bins as a minimum-cost network flow graph and then flows cells out of high-density bins into low-density bins. Its objective function is to minimize a weighted sum of squared cell movements. Like network flow, Mongrel [4] legalizes by moving cells out of bins that exceed their capacity. Mongrel iteratively computes a low cost path of movement from a source bin to a destination bin, then ripples cells from the source to the destination by only allowing cells to move by at most one bin. The approach used by [5] tries to legalize one row at a time while preserving the original cell order. If not all the cells fit in the row, it uses dynamic programming to decide which cells to preserve in the row and which ones to push into the next row. Similar to [5], the approach of [6] optimizes cell locations for a single row to optimize wirelength and cell perturbation. Another row based legalization is introduced in [7], which uses a shorted path algorithm to optimize either perturbation or wirelength. A simple and effective technique [8] is to sort all the cells based on their x coordinates then place one by one like packing. And other heuristics [9] [10] [11] [12] [13] employ greedy heuristics to legalize placement.

An incremental placement based approach [14] can be used for legalization of large placement changes as the result of buffer insertion and gate sizing. It uses a slicing tree to represent the original floorplan, adjusts the size of each slice to accommodate additional placement demands and finally applies a detailed placer that can preserve the whitespace distribution. Although this approach can optimize wirelength and preserve white space, as a detailed placement algorithm it could significantly change the placement relative order which might have unpredictably impact on existing timing.

There are a few spreading techniques that can also be used for legalization potentially. These techniques [15] [16] [17] were originally designed to spread out cells in congested regions during analytical placement. [15] uses grid-warping technique to move cells as they are tethered to a warping grid; [16] uses a similar technique as [4] to move cells from congested bins to their neighboring bins; [17] uses a cell shifting technique to reduce the maximum bin density.

Although all these techniques can be used to perturb the placement, the perturbation is rather discrete compared to the diffusion-based method that is more continuous and smooth. Another technique for smooth cell movement is proposed in [18] recently. Similar to diffusion, it uses bin density gradient to spread out cells globally. The difference is that it moves cells as if they are tethered to an expanding grid, while diffusion moves cells directly based on density. Although [18] also uses a computational geometry based method to further arrange cells locally while maintaining the local relative order, these two methods produce similar results as the experiment section will show.

C. Force-Directed Techniques

One may view force-directed global placement (e.g., [19]) as a legalization technique. The algorithm starts with an overlapping global placement. It then adds forces based on the density of the layout to spread out the placement, proceeding iteratively. At first glance diffusion (DIFF) and force-directed spreading (FORCE) share a common approach of using the existing density map to spread out cells until bin density constraints are satisfied. However, there are several key differences between these approaches.

- FORCE generates spreading forces from a *global* density distribution, while DIFF generates cell velocities from *local* densities.
- FORCE models the placement density as an *electric* field that acts on the cells; each cell has some attraction or repulsion to each region of the layout. On the other hand, DIFF physically models the placement density as a *diffusion* process in which cells move along their local density gradients.
- Because their abstract physical models are different, so are the solution techniques. FORCE solves *linear algebraic equations* generated by cell connections and spreading forces, while DIFF solves the *partial differential equation* generated by local neighborhood densities.
- FORCE requires cell *connection* and density information for spreading, while DIFF only needs *density* information. DIFF does not consider the circuit connectivity.

III. The Diffusion Process

Algorithms which model the physical process of diffusion are not common in the VLSI physical design area, though they do exist elsewhere in the semiconductor industry. For example, the dopant diffusion process on chip substrate is a well known diffusion process [20]. Intuitively, materials from highly concentrated areas flow into less concentrated areas. Diffusion is driven by the concentration gradient, which is the slope and steepness of the concentration difference at a given point. The increase in concentration in a cross section of unit area with time is simply the difference of the material flow into the cross section and the material flow out of it. Diffusion reaches an equilibrium state when the material concentration is uniformly distributed.

Mathematically, we can describe the relationship of material concentration with time and space using the following partial differential equation.

$$\frac{\partial d_{x,y}(t)}{\partial t} = D\nabla^2 d_{x,y}(t) \quad (1)$$

where $d_{x,y}(t)$ is the material concentration at position (x, y) at time t and D is the diffusivity which determines the speed of diffusion. For simplicity of presentation, assume $D = 1$ for the rest of the paper. Equation (1) states that the speed of density change is linear with respect to its second order gradient over the density space. This implies that elements migrate with increased speed when the local density gradient is higher. In the context of placement, cells will move quicker when their local density neighborhood has a steeper gradient.

When the region for diffusion is fixed (as in placement), the boundary conditions are defined as $\nabla d_{x_b,y_b}(t) = 0$ for coordinates (x_b, y_b) on the chip boundary. We also define coordinates over fixed blocks in the same way in order to prevent cells from diffusing on top of fixed blocks. This forces cells to diffuse around the blocks.

In diffusion a cell migrates from an initial location to its final equilibrium location via a non-direct route. This route can be captured by a velocity function that gives the velocity of a cell at every location in the circuit for a given time t. This velocity at certain position and time is determined by the local density gradient and the density itself. Intuitively, a sharp density gradient causes cells to move faster. For every potential (x, y) location, define a 2-dimensional velocity field $\mathbf{v}_{x,y} = (v^H_{x,y}, v^V_{x,y})$ of diffusion at time t as follows:

$$\begin{aligned} v^H_{x,y}(t) &= -\frac{\partial d_{x,y}(t)}{\partial x}/d_{x,y}(t) \\ v^V_{x,y}(t) &= -\frac{\partial d_{x,y}(t)}{\partial y}/d_{x,y}(t) \end{aligned} \quad (2)$$

Given this equation, and a starting location $(x(0), y(0))$ for a particular element, one can find the new location $(x(t), y(t))$ for the element at time t by integrating the velocity field:

$$\begin{aligned} x(t) &= x(0) + \int_0^t v^H_{x(t'),y(t')}(t')dt' \\ y(t) &= y(0) + \int_0^t v^V_{x(t'),y(t')}(t')dt' \end{aligned} \quad (3)$$

Equations (1), (2), (3) are sufficient to simulate the diffusion process. Given any particular element, one can now find the new location of the molecule at any point in time t. To apply this paradigm to placement, one needs to migrate from a continuous space to a discrete place since cells have various rectangular sizes and the placement image itself is discrete. The next section presents a technique to simulate diffusion specifically for placement.

IV. Diffusion Based Placement

One can discretize continuous coordinates by dividing the placement areas into equal sized bins indexed by (j, k). Assume the coordinate system is scaled so that the width and height of each bin is one. Then location (x, y) lies inside bin $(j, k) = (\lfloor x \rfloor, \lfloor y \rfloor)$. We can also discretize continuous time t as $n\Delta t$, where Δt is the size of the discrete time step.

A. Computing Bin Density

Instead of the continuous density $d_{x,y}$, we now can describe diffusion in the context of the density $d_{j,k}$ of bin (j, k). The initial density $d_{j,k}(0)$ of each bin (j, k) can be defined as $d_{j,k}(0) = \Sigma\hat{A}_i$ where $\hat{A}_i$ is the overlapping area of cell i and bin (j, k).

For simplicity, assume that if a fixed block overlaps a bin, it overlaps the bin completely. In these cases, the bin density is defined to be one, though boundary conditions prevent cells from diffusing on top of fixed blocks.

Assume that the density $d_{j,k}(n)$ has already been computed for time n. Now one needs to find how the density changes and cells movements for the next time step $n + 1$. We use the Forward Time Centered Space (FTCS) [21] scheme to discretize Equation (1). The new bin density is given by

$$\begin{aligned} d_{j,k}(n+1) &= d_{j,k}(n) \\ &+\frac{\Delta t}{2}(d_{j+1,k}(n) + d_{j-1,k}(n) - 2d_{j,k}(n)) \\ &+\frac{\Delta t}{2}(d_{j,k+1}(n) + d_{j,k-1}(n) - 2d_{j,k}(n)) \end{aligned} \quad (4)$$

The new density of a bin at time $n + 1$ depends only on its density and the density of its four neighbor bins. Note that one does not actually use the cell locations at time $n + 1$ to compute the density. The degree of migration out of (or into) the bin is proportional to its local gradient. For example, consider a density distribution at a given time n shown in Fig. 1 and assume $\Delta t = 0.2$. The density of bin $(1, 1)$ at time $n+1$ is given by:

$$\begin{aligned} d_{1,1}(n+1) &= d_{1,1}(n) + \frac{0.2}{2}(d_{2,1}(n) + d_{0,1}(n) - 2d_{1,1}(n)) \\ &+\frac{0.2}{2}(d_{1,2}(n) + d_{1,0}(n) - 2d_{1,1}(n)) = 0.98 \end{aligned}$$

B. Computing Cell Velocity

Just as Equation (1) can be discretized to compute placement bin density, Equation (2) can be discretized to compute the velocity for cells inside the bins. For now, assume that each cell in the bin is assigned the same velocity, the velocity for the bin, given by:

$$\begin{aligned} v^H_{j,k}(n) &= -\frac{d_{j+1,k}(n) - d_{j-1,k}(n)}{2d_{j,k}(n)} \\ v^V_{j,k}(n) &= -\frac{d_{j,k+1}(n) - d_{j,k-1}(n)}{2d_{j,k}(n)} \end{aligned} \quad (5)$$

The horizontal (vertical) velocity is proportional to the differences in density of the two neighboring horizontal (vertical) bins. For example, the velocity for bin $(1, 1)$ in Fig. 1 is given by:

$$\begin{aligned} v^H_{1,1} &= -\frac{d_{2,1} - d_{0,1}}{2d_{1,1}} = -\frac{0.4 - 1.4}{2(1.0)} = 0.5 \\ v^V_{1,1} &= -\frac{d_{1,2} - d_{1,0}}{2d_{1,1}} = -\frac{0.4 - 1.6}{2(1.0)} = 0.6 \end{aligned}$$

Similarly, densities for other bins are given by $\mathbf{v}_{1,2} = (0.5, 0)$, $\mathbf{v}_{2,1} = (0.25, -0.25)$ and $\mathbf{v}_{2,2} = (-0.125, 0.125)$.

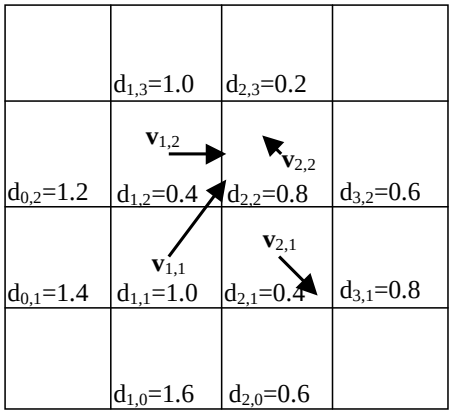

Fig. 1. Density and velocity distribution at time n.

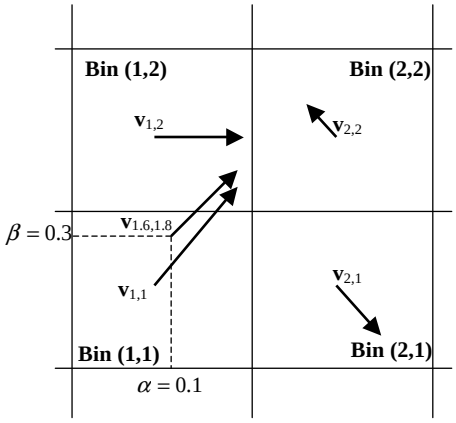

Fig. 2. Velocity interpolation within a bin.

Note that bin $(1, 2)$ has no vertical velocity component since the densities both above and below are equal to 1.0. To make sure that fixed cells and bins outside the boundary do not move, we enforce $v^V = 0$ at a horizontal boundary and $v^H = 0$ at a vertical boundary.

C. Cell Velocity Interpolation

Assuming that each cell in a bin has the same velocity fails to distinguish between the relative locations of cells within a bin. Further, two cells that are placed sided by side but in different bins can be assigned very different velocities, which could change their relative ordering. Since the goal of placement migration is to preserve the integrity of the original placement, this behavior cannot be permitted. To remedy this behavior, we apply velocity interpolation to generate a velocity for any given (x, y).

Let bin (p, q) be such that the four closest bin centers to (x, y) are (p, q), $(p + 1, q)$, $(p, q + 1)$ and $(p + 1, q + 1)$. Let $\alpha = x + 0.5 - \lfloor x + 0.5 \rfloor$ and $\beta = y + 0.5 - \lfloor y + 0.5 \rfloor$. If $\alpha = \beta = 0$, then (x, y) is located at the center of bin (p, q) and its velocity is given velocity $v_{p,q}$. As shown in Fig. 2, the bin velocity will be marked at the center of each bin. The velocity for a point inside of a bin is interpolated by the velocities at its four closest centers. The velocity for cell (x, y) (denoted by $(v^H_{x,y}, v^Y_{x,y})$) is given by

$$\begin{aligned} v^H_{x,y} &= v^H_{p,q} + \alpha(v^H_{p+1,q} - v^H_{p,q}) + \beta(v^H_{p,q+1} - v^H_{p,q}) \\ &\quad +\alpha\beta(v^H_{p,q} + v^H_{p+1,q+1} - v^H_{p+1,q} - v^H_{p,q+1}) \\ v^V_{x,y} &= v^V_{p,q} + \alpha(v^V_{p+1,q} - v^V_{p,q}) + \beta(v^V_{p,q+1} - v^V_{p,q}) \\ &\quad +\alpha\beta(v^V_{p,q} + v^V_{p+1,q+1} - v^V_{p+1,q} - v^V_{p,q+1}) \end{aligned} \quad (6)$$

Consider the example shown in Fig. 2, which is actually a close-up of Fig. 1. For an example location $(x = 1.6, y = 1.8)$, we have $\alpha = 0.1$ and $\beta = 0.3$. The velocity for this point is given by:

$$\begin{aligned} v^H_{1.6,1.8} &= v^H_{1,1} + 0.1(v^H_{2,1} - v^H_{1,1}) + 0.3(v^H_{1,2} - v^H_{1,1}) \\ &\quad +0.03(v^H_{1,1} + v^H_{2,2} - v^H_{2,1} - v^H_{1,2}) = 0.45625 \\ v^V_{1.6,1.8} &= v^V_{1,1} + 0.1(v^V_{2,1} - v^V_{1,1}) + 0.3(v^V_{1,2} - v^V_{1,1}) \\ &\quad +0.03(v^V_{1,1} + v^V_{2,2} - v^V_{2,1} - v^V_{1,2}) = 0.40175 \end{aligned}$$

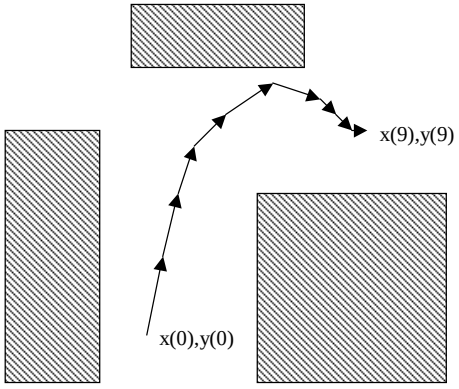

Fig. 3. A cell movement trajectory during diffusion.

D. Computing Cell Location

Since the velocity for each cell can be determined at time $n = \frac{t}{\Delta t}$, one can compute its new placement via a discretized form of Equation (3). It is easier to comprehend (and it is more useful) in its recursive form. Suppose we have already computed $(x(n), y(n))$. Using Taylor expansion gives compute $x(n + 1), y(n + 1)$ as:

$$\begin{aligned} x(n+1) &= x(n) + v^H_{x(n),y(n)} \cdot \Delta t \\ y(n+1) &= y(n) + v^V_{x(n),y(n)} \cdot \Delta t \end{aligned} \quad (7)$$

An example is shown in Fig. 3 in which a cell takes nine discrete time steps. Observe how the cell never overlaps a blockage and also how the magnitude of its movements becomes smaller toward the tail end of its path.

V. Diffusion-Based Legalization Algorithm

Now that we have presented the general diffusion paradigm, we show how to apply this technique to legalization. Recall that to legalize the design, we require each bin to have density $d_{j,k} \leq d_{max}$. Once this is achieved, local slide and spiral methods can be used to quickly and easily achieve a legal placement. Thus we are given an existing placement with locations (x_i, y_i) for each cell i, N placement bins, and a maximum bin density d_{max}.

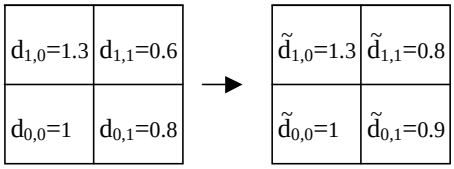

Fig. 4. The initial density map is modified to a new density map so that the average bin density is 1.0.

A. Density Map Manipulation

Since the diffusion process reaches equilibrium when each bin has the same density, we can expect the final density after diffusion to be the same as the average density $\Sigma d_{j,k}/N$. This may cause unnecessary spreading especially if the average density is well below the maximum density constraint. For example, once every bin is below the maximum density constraint, diffusion can cause additional spreading even though the requirements for legalization have been met. This spreading will no doubt further perturb the placement from its original state.

Therefore, before beginning diffusion we need to properly set the initial density values of those bins under the maximum density. To achieve this, we artificially increase the densities of those bins less than d_{max} so that the average density equals d_{max}.

One way to adjust $d_{j,k}$ is

$$\tilde{d}_{j,k} = \begin{cases} d_{max} - (d_{max} - d_{j,k})\frac{A_o}{A_s} & d_{j,k} < d_{max} \\ d_{j,k} & d_{j,k} \geq d_{max} \end{cases} \quad (8)$$

where A_o is the total area over d_{max} and A_s is total area less than d_{max}, which is the available space to hold the A_o after spreading. We can validate that $\frac{\Sigma \tilde{d}_{j,k}}{N} = d_{max}$.

Fig. 4 shows an example of the density manipulation for a 2×2 bins. In the left figure, there is one bin whose density 1.3 is over the maximum allowed density 1, two bins whose densities are lower than 1, and the other bin whose density is exactly 1. Therefore $A_o = d_{1,0} - 1 = 0.3$ and $A_s = (1 - d_{1,1}) + (1 - d_{0,1}) = 0.6$. If we adjust the density on those two bins under 1 with (8), we will get the density map shown on the right, $\frac{\tilde{d}_{0,0}+\tilde{d}_{0,1}+\tilde{d}_{1,0}+\tilde{d}_{1,1}}{4} = 1$.

$\tilde{d}_{j,k}$ will be used as the initial condition ($t = 0$) for the diffusion equation (4),

$$d_{j,k}(0) = \tilde{d}_{j,k}. \quad (9)$$

B. Macro and Chip Boundary Handling

At the boundary of the chip or a fixed macro, there is no diffusion between either side of the boundary. Therefore, we need to make the densities on both sides the same to assure the density gradient is zero when computing (4). On a horizontal boundary, we make $d_{j,k+1}(n) = d_{j,k-1}(n)$ if bin (j,k) is on the lower side of the boundary, or $d_{j,k-1}(n) = d_{j,k+1}(n)$ if on the upper side; while on a vertical boundary, $d_{j+1,k}(n) = d_{j-1,k}(n)$ if on the left side, or $d_{j-1,k}(n) = d_{j+1,k}(n)$ if on the right side.

For example, suppose $\Delta t = 0.2$ and bin $(4,3)$ to $(5,4)$ are fixed, the density value for time n is shown in Fig. 5. Bin $(3,4)$ is on the left vertical boundary of the fixed macro, while bin $(4,5)$ is on the upper horizontal boundary. When computing $d_{3,4}(n+1)$, we make $d_{4,4}(n) = d_{2,4}(n)$, thus (4) becomes:

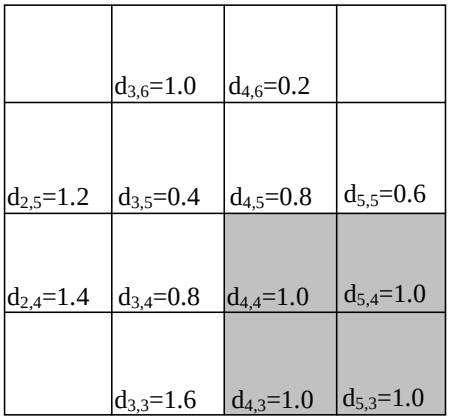

Fig. 5. Bins with fixed blocks are shaded to illustrate density computations under boundary conditions.

$$\begin{aligned} d_{3,4}(n+1) &= d_{3,4}(n) + \frac{0.2}{2}(d_{2,4}(n) + d_{2,4}(n) - 2d_{2,3}(n)) \\ &\quad + \frac{0.2}{2}(d_{3,5}(n) + d_{3,3}(n) - 2d_{3,4}(n)) = 0.96. \end{aligned}$$

Similarly, we make $d_{4,4}(n) = d_{4,6}(n)$ to compute $d_{4,5}(n+1)$,

$$\begin{aligned} d_{4,5}(n+1) &= d_{4,5}(n) + \frac{0.2}{2}(d_{3,5}(n) + d_{5,5}(n) - 2d_{4,5}(n)) \\ &\quad + \frac{0.2}{2}(d_{4,6}(n) + d_{4,6}(n) - 2d_{4,5}(n)) = 0.62. \end{aligned}$$

For bins inside of fixed macros, we do not update the density.

C. Algorithm

The algorithm begins by computing the initial bin density using the given placement, then manipulates the density map to avoid over spreading. Starting from time 0, it recursively computes bin density, bin velocity and cell locations for each time step n. It stops when the maximum bin density is less than d_{max}. The complete diffusion algorithm is given in Algorithm 1.

After diffusion, the placement should have a maximum density of d_{max} and is roughly legal. We need to run a final legalization step to put cells in a circuit row without overlap. Any legalizer can be used at this step. It will only take the legalizer a little effort to remove those overlaps. Here we use the IBM CPlace internal legalizer.

Fig. 6 shows an example of diffusion-based legalization in a small region surrounded by fixed blocks. The left figure shows the initial illegal placement. The right figure is the placement out of legalization. Cells are shaded to represent their relative order. We can observe that after diffusion the relative orders are not changed.

VI. Robust Local Diffusion

The diffusion based legalization algorithm introduced in previous sections provides a smooth global solution based on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Algorithm 1 Diffusion-based Legalization Algorithm

Inputs: cell locations (x_i, y_i), N bins, maximum density d_{max}

1: map cells onto bins and compute $d_{j,k}$ for each bin (j,k)
2: compute $\tilde{d}_{j,k}$ using (8), the average bin density is now d_{max}
3: $d_{j,k}(0) \leftarrow \tilde{d}_{j,k}$
4: $n \leftarrow 0$
5: **repeat**
6: compute $v^H_{j,k}(n), v^V_{j,k}(n)$ for each bin (j,k) using (5)
7: compute $x_i(n), y_i(n)$ for each cell i using (7) and velocity interpolation (6)
8: compute $d_{j,k}(n+1)$ for each bin (j,k) using (4)
9: $n \leftarrow n+1$
10: **until** $max(d_{j,k}(n)) \leq d_{max} + \Delta$

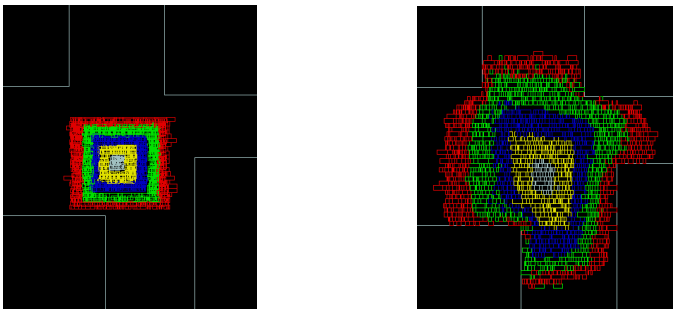

Fig. 6. Placements before and after diffusion-based legalization.

the diffusion process. Diffusion global legalization is tightly coupled with the following detailed legalization because the diffusion algorithm is continuous. This diffusion algorithm, however, could still move cells in legal regions which might result in unnecessary cell displacement, and the run time is a bit higher than other global algorithms as shown in section VII. Therefore a fast and robust diffusion algorithm which moves fewer cells is needed. Another major contribution of this paper is that we introduce a robust local diffusion algorithm, which utilizes a couple of fast and robust diffusion techniques that only "diffuse" cells as necessary to legalize placement, thus making fewer cell movements and using less run time than the diffusion algorithm described in previous section. The first technique is local diffusion, the second is dynamic density update. The reasoning and details of these techniques are given in this section.

A. Local Diffusion

The diffusion algorithm as shown in Algorithm 1 can be called *global* diffusion because it moves cells wherever there exists any density difference between adjacent bins. With initial density manipulation, we can limit the amount of spreading to be just enough to reduce densities of illegal bins (where bin densities are higher than the target density) to the target density. However, it still could move cells in legal bins (where bin densities are less than target density) unnecessarily because the spreading is global. The stopping criteria only check whether the maximum bin density is less than the target

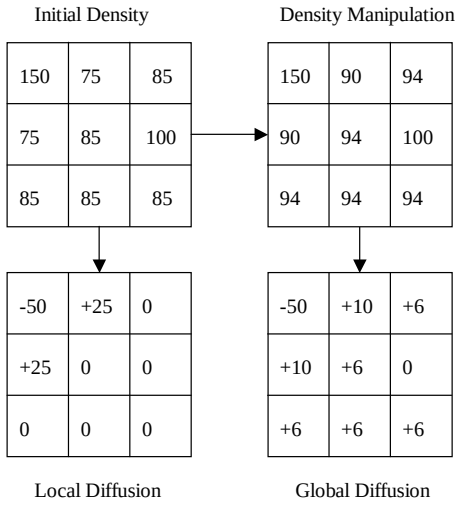

Fig. 7. Global vs. local diffusion example.

density, it does not check for each bin. Therefore, it could move cells in those bins whose densities are already less than the target density as long as there is a density difference between adjacent bins. In other words, cells all over the chip could end up moving, even if moving cells from illegal bins to their adjacent bins is enough to satisfy the target density. The ideal diffusion for legalization should be *local*, which only spreads out cells from illegal bins to nearby free space without moving cells in other legal bins unnecessarily.

Diffusion examples for both global and local diffusion are shown in Fig. 7. Suppose there are a total of 9 bins, and that the upper left bin has a density of 150%, while the remaining 8 bins are lesser or equal to 100% density. The diffusion process using initial density manipulation will move cells in almost every bin. For example, the density of the bin at the lower right hand corner increases 6%, which comes from cells moved into this bin. However, as shown in the local diffusion figure, there is a local solution that only affects the adjacent bins of the overflowed bin. The total amount of movement in this local solution is much less than the global solution.

We propose to use a diffusion window to implement local diffusion, which only spreads out cells in a window around bins with higher density than the target density. The procedure of identifying diffusion windows is described in Algorithm 2.

This procedure first sets the move type of each bin as fixed. Then for each bin (j,k), it computes the average density $d'_{j,k}$ of bins within a distance of W_1 to bin (j,k). W_1 can range from 1 to 10 bins. If the average density is larger than the targeted density d_{max}, then it marks those bins which are W_2 ($W_2 >= W_1$) distance away from bin (j,k) as movable. Conceptually, this procedure first analyzes the average bin density with the analysis window W_1, then it allows diffusion to work on the diffusion window W_2 around those bins with average density greater than the target density. This is because the final legalization does not need all the bins' densities less than 1. As long as the average local density is less than 1, the detailed legalization algorithm can make them legal quite easily. Therefore we only use average bin density of a window

Algorithm 2 Identify Local Diffusion Window

Inputs: max density d_{max}, analysis window size W_1 and diffusion window size W_2

```
for all bin (j, k) do
    initialize bin (j, k) as fixed
for all bin (j, k) do
    d'_{j,k} = 0; N = 0
    for all bin (j', k') within a distance of W_1 to (j, k) do
        d'_{j,k} = d'_{j,k} + d_{j',k'}
        N = N + 1
    d'_{j,k} = d'_{j,k} / N
    if d'_{j,k} greater than d_max then
        for all bin (j', k') within a distance of W_2 to (j, k) do
            mark bin (j', k') as movable
```

to measure whether we need to diffuse, and mark those bins far away from overflowed region as fixed. If bins are marked as fixed, the diffusion process will not move any cells in those bins as described in [2].

The window sizes W_1 and W_2 determine how much spreading the diffusion process would produce and the speed of spreading. In section VII we will examine the impact of the window size to legalization performance.

After a certain period of diffusion steps, Algorithm 2 should be invoked again to correctly reflect the density distribution at that time. Note that local diffusion does not require the initial density manipulation step because the window identification process guarantees the minimum spreading.

B. Dynamic Density Update

Although we can use (4) to compute bin densities during the diffusion process, the computed densities are not exactly the same as the real placement densities. The equations of the physical diffusion process (4) (5) (7) assume continuous density distribution. However, the real standard cell distribution does not always satisfy this condition. First, cells are not equally distributed inside a bin. For example, it is possible that all the cells inside of a bin are on the right side of the bin. As shown in Fig. 8, the left bin contains 100% cells that are clustered on the right side of the bin. The target density is 50%. the diffusion process would move all the cells in that bin to the adjacent bin on the right. This would make the final density of the right bin 100% and the left bin 0%, which is different than the computed density equilibrium of 50% on both bins. Second, cells have different sizes, and moving one cell might pertub the design more than moving another. Third, the error of integral equation (7) may accumulate. Therefore, it is necessary to update the density based on the real cell placement when the error exceeds a certain threshold.

Since errors are associated with diffusion simulation, we could update the density map for every simulation step N_U. We choose to update density whenever we need to identify a local diffusion window. The impact of density update period on the legalization performance is studied in section VII.

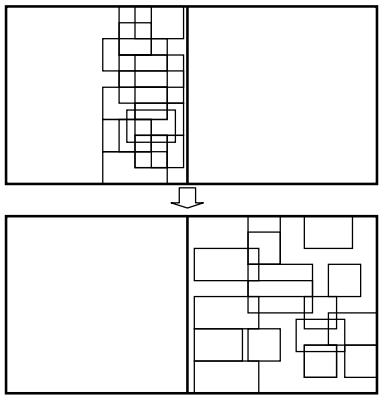

Fig. 8. Density error caused by irregular cell distribution.

C. Robust Local Diffusion Algorithm

In this section, we put together the complete legalization flow based on the robust local diffusion algorithm. As shown in Algorithm 3, we iteratively identify local diffusion window (Algorithm 2) and run diffusion Algorithm 1 to perform local diffusion. When we can no longer reduce the total bin density overflow, we put cells on circuit row and eliminate all overlaps to finalize the placement.

Algorithm 3 Robust Local Diffusion-based Legalization

```
repeat
    identify local diffusion regions (Algorithm 2)
    run diffusion for N_U steps (Algorithm 1)
    update bin density
until no improvement in overflow
final legalization
```

VII. Experimental Results

This section reports experimental results for the global diffusion-based legalizer ($DIFF(G)$) and robust local diffusion-based legalizer ($DIFF(L)$) by comparing it to a greedy legalizer ($GREED$) which uses slide-and-spiral techniques to place cells onto their nearest legal locations and to a network flow legalizer ($FLOW$) which uses a min-cost flow algorithm to direct cell movements. $FLOW$ is similar to [3]: first, cells are roughly spread out by the min-cost flow algorithm, then they are moved to their final positions such that all overlaps are removed. $GREED$ sorts all the cells and places them sequentially. It first tries to place a cell at the original location. If that location is occupied, it performs a spiral search starting from the original location. During a spiral search, it could slide other placed cells a little bit in order to fit in. All four legalizers are implemented in C and run on an IBM P690 server. The timing results are reported by IBM Einstimer.

We use seven industrial circuits for comparison. The sizes of circuits range from 64K cells to over a million cells. All the circuits are legally placed initially. To simulate the behavior of repowering in physical synthesis we inflate cells and create overlaps that need to be resolved. The circuit sizes and amount

TABLE I
DESIGN SIZES AND INFLATIONS

testcases	number of cells	size(mm)	Inflation(%)
ckt1	64K	1.9 x 1.9	23.1
ckt2	72K	2.3 x 2.3	32.4
ckt3	159K	5.3 x 5.3	47.2
ckt4	216K	9.0 x 9.0	40.4
ckt5	307K	11.9 x 11.9	25.4
ckt6	440K	10.0 x 10.0	42.2
ckt7	1076K	13.0 x 13.0	18.9

of inflations are reported in Table I. The inflations are reported as the percentage with respect to the total moveable cell areas.

A. Legalization Performance

Tables II, III, and IV show the TWL, worst slack and FOM[1] results of the four legalizers. Since we inflate cells, the new placement has longer wire length, worse slack and FOM than those of the original placements. The absolute Δ and relative Δ rows of each table report the average improvements of $DIFF(G)$ and $DIFF(L)$ over the best result of $FLOW$ and $GREED$. The relative Δ measures the relative improvement of the results before and after legalization. For example, on $ckt1$, $GREED$ increases the TWL by $13.23 - 11.48 = 1.75$, while $DIFF(L)$ only increases the TWL by $12.11 - 11.48 = 0.63$, therefore the relative Δ is $(1.75 - 0.63)/1.75 = 64\%$. We can observe that both $DIFF(G)$ and $DIFF(L)$ achieve smaller TWL than $FLOW$ or $GREED$. The average relative improvement of TWL over seven circuits is 17% for $DIFF(G)$ and 35% for $DIFF(L)$. The average improvement on wiring congestion after global routing is 27% and 35% for $DIFF(G)$ and $DIFF(L)$, respectively (the detailed numbers are not shown to save space). The slack and FOM degradations of $DIFF(G)$ and $DIFF(L)$ are also much less than those of $FLOW$ and $GREED$. The average improvements of slack over the best of $FLOW$ and $GREED$ are 48% and 63% by $DIFF(G)$ and $DIFF(L)$, respectively. And the average improvements of FOM over the best of $FLOW$ and $GREED$ are 36% and 62%. These results are actually conservative because in $ckt5$, $ckt6$ and $ckt7$, the inflation did not cause a larger amount of overlaps due to sparse initial placements. We have also compared the four legalizers on industry circuits generated by physical synthesis without legalization, and the results consistently show that $DIFF$ results in better timing, TWL and congestion for those circuits.

Table V reports the runtimes for four legalizers. Although the runtime of $DIFF(G)$ is about 2X of $FLOW$, the $DIFF(L)$ is in the ballpark of the other two legalizers. On average, $DIFF(L)$ actually runs slightly faster than those two.

We also test $FLOW$ and $DIFF(G)$ with different cell overlapping distributions, i.e., through distributed or concentrated inflations. Table VI shows the results of $DIFF(G)$ (referred as $D(G)$) and $FLOW$ on the same circuit $ckt1$ but with different inflation distributions. The inflations are centralized (C), or evenly distributed (D). The amounts of inflations are 23% and 18% for centralized and distributed cases, respectively. The localized inflation mimics a hot-spot that needs to be spread out. The distributed inflation is more like legalization after physical synthesis. The Δ row reports the difference of C and D. Both $DIFF(G)$ and $FLOW$ get worse results on concentrated inflation distribution than those on distributed inflation distribution, although the amount of inflations are actually smaller for concentrated case. However, we can observe that $DIFF(G)$ is less sensitive to the inflation distribution than $FLOW$. The TWL degradation of $DIFF(G)$ is only $0.16m$ compared to $1.06m$ of the

[1] FOM is a metric that measures the amount of work needed for a designer to close timing; basically it is weighted area under the timing histogram of the paths with negative slack [22].

TABLE II
TWL COMPARISON OF FOUR LEGALIZERS (M)

testcases	Base	GREED	FLOW	$DIFF(G)$	$DIFF(L)$
ckt1	11.48	13.23	13.40	12.46	12.11
ckt2	15.06	17.03	17.33	16.65	16.17
ckt3	47.10	52.47	52.65	51.76	50.72
ckt4	51.37	59.02	58.67	56.85	55.71
ckt5	150.8	159.0	159.2	158.7	157.4
ckt6	166.6	175.6	175.4	174.8	173.1
ckt7	367.7	382.7	382.5	381.7	379.8
absolute Δ				1.9%	3.6%
relative Δ				16.8%	35.0%

TABLE III
WORST SLACK COMPARISON OF FOUR LEGALIZERS (NS)

testcases	Base	GREED	FLOW	$DIFF(G)$	$DIFF(L)$
ckt1	-0.571	-1.266	-1.497	-0.921	-0.900
ckt2	0.275	-0.287	-0.789	-0.065	0.001
ckt3	-0.265	-1.155	-1.121	-0.265	-0.265
ckt4	-1.592	-3.569	-3.447	-2.373	-2.151
ckt5	-0.623	-6.072	-3.640	-2.047	-1.97
ckt6	-0.387	-3.450	-3.562	-3.305	-3.103
ckt7	-0.796	-1.601	-1.274	-0.796	-0.796
absolute Δ				42.5%	48.1%
relative Δ				48.0%	62.9%

TABLE IV
FOM COMPARISON OF FOUR LEGALIZERS (NS)

testcases	Base	GREED	FLOW	$DIFF(G)$	$DIFF(L)$
ckt1	-3188	-4942	-8441	-3883	-2552
ckt2	0	-247	-620	-319	-169
ckt3	-446	-1073	-1054	-524	-511
ckt4	-557	-1321	-1068	-862	-753
ckt5	-144	-4827	-4871	-3069	-2677
ckt6	-15286	-24694	-24154	-22936	-22543
ckt7	-2583	-5391	-7631	-4157	-3919
absolute Δ				18.0%	34.2%
relative Δ				36.3%	62.2%

TABLE V
CPU TIME COMPARISON OF FOUR LEGALIZERS (S)

testcases	GREED	FLOW	$DIFF(G)$	$DIFF(L)$
ckt1	161	55	107	51
ckt2	41	24	74	34
ckt3	228	197	290	125
ckt4	320	313	581	153
ckt5	414	584	841	323
ckt6	619	626	1231	910
ckt7	2102	1768	4681	2034
Average	1	0.86	1.68	0.77

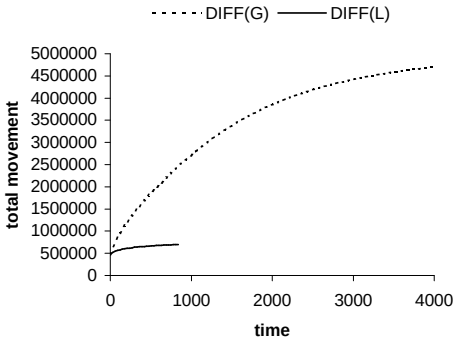

Fig. 9. Total cell movement during diffusion on ckt1.

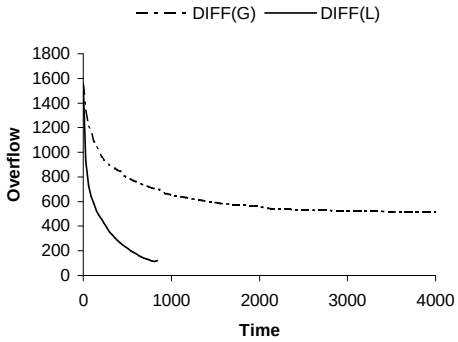

Fig. 10. Total density overflow during diffusion on ckt1.

$FLOW$. The slack and FOM degradation of $DIFF(G)$ is also significantly lower than those of $FLOW$. This indicates that diffusion-based legalization can handle hot-spot situation better than the network flow based method.

TABLE VI
INFLATION DISTRIBUTION EFFECT ON LEGALIZATION

type(%)	TWL (m)		Slack (ns)		FOM(ns)	
	FLOW	D(G)	FLOW	D(G)	FLOW	D(G)
D(23)	13.40	12.46	-1.497	-0.921	-8441	-3883
C(18)	14.46	12.62	-1.976	-1.253	-11822	-4361
Δ	1.06	0.16	-0.479	-0.332	-3381	-478

B. Movement and Overflow Analysis for Diffusion

Bin density overflow and cell movements are two most important measurements for diffusion legalization algorithm. As described in the previous section, detailed legalization can still do a good job if the local average density is less than targeted density. Therefore, for density overflow, we only measure the local average density overflow, i.e. $max(d'_{j,k} - d_{max}, 0)$. For cell movement, we are interested in total cell movement and maximum cell movement.

Fig. 9 compares the total cell movement during the diffusion process for $DIFF(G)$ and $DIFF(L)$ on $ckt1$. We can observe that local diffusion makes much less cell movement than global diffusion.

Fig. 10 compares the total bin density overflow during the diffusion process for $DIFF(G)$ and $DIFF(L)$ on $ckt1$. We can observe the density update significantly reduces the density overflow. Density update also helps diffusion run faster because it reduces overflow quickly.

Tables VII and VIII compare the density overflow and cell movement between $DIFF(G)$ and $DIFF(L)$. Both maximum and total numbers are reported. Clearly, $DIFF(L)$ has much less density overflow and cell movements across the board. The average cell movement reduction of $DIFF(L)$ over $DIFF(G)$ is 70% and the average density overflow reduction is 58%. Note that the density overflow of both global and local diffusions are not zero. For global diffusion, although we check the maximum bin density as the stopping criteria, the density it measures is not the real density as we mentioned in section VI-A. For local diffusion, the overflow measured at each density update point is affected by the density update period N_U. It does not converge nicely as global diffusion, which always reduce the computed overflow (not the real overflow) during the diffusion process. One might see overflow increase a bit then continue decrease when maximum density is close to the target density. From that point, it usually takes a long time to converge. Therefore, for runtime reason, we stop local diffusion whenever the overflow starts to increase, which is a good indication of the majority of diffusion is done. This issue is also discussed in section VII-C.

TABLE VII
DENSITY OVERFLOW COMPARISON

testcases	DIFF(G)		DIFF(L)	
	max	total	max	total
ckt1	11.82	527.92	0.08	120.48
ckt2	11.91	2105.30	2.60	1264.42
ckt3	11.98	3738.26	2.63	660.57
ckt4	22.44	2869.68	5.14	772.24
ckt5	23.28	13415.87	4.88	9510.61
ckt6	35.74	35506.25	11.54	21089.78
ckt7	11.42	8182.58	3.77	2966.06
Improvement			78%	58%

TABLE VIII
CELL MOVEMENT COMPARISON

testcases	DIFF(G)		DIFF(L)	
	max	total$_{(10^6)}$	max	total$_{(10^6)}$
ckt1	284.41	4.6706	148.47	0.71
ckt2	598.98	9.2568	463.45	3.7434
ckt3	536.92	5.8905	656.82	2.3571
ckt4	538.65	6.604	215.39	2.0497
ckt5	622.08	3.537	324.52	1.05
ckt6	498.4	15.129	341.6	4.8348
ckt7	349.6	11.2671	550.56	2.2869
Improvement			19%	70%

C. Parameter Characterization

In this section, we show the performance tradeoff of different local diffusion parameters, including bin size B, window size W_1 and W_2, and density updating period N_U.

Fig. 11 plots the total movements and worst negative slack (WNS) result of legalization with different bin size B for $ckt2$ (the area of a bin is B^2). We can observe that when bin size is smaller than twice the height of circuit row (12), the result is worse. This is because when bin size is smaller than the cell,

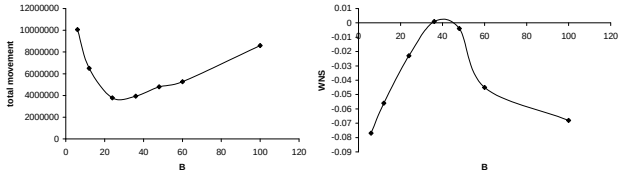

Fig. 11. Total cell movements and worst negative slack with different bin size (B) on ckt2.

we can no longer formulate the cell movement as molecular movement. We can also see that when bin size is too large, the result also degrades. This is because that the larger bin size could produce unnecessary spreading thus make timing worse. Therefore the sweet spot for bin size is between 20 and 40, which is about 2 to 4 circuit row high.

Fig. 12 and Fig. 13 show the total movement and WNS result with different local diffusion window size W_1 and W_2 (as used in Algorithm 2) for $ckt2$. Fig. 12 assumes $W_1 = W_2$, and Fig. 13 uses $W_1 = 2$. We can observe that larger W_1 and W_2 cause more spreading. Larger W_2 makes the spreading faster. For legalization, overspread usually means worse timing and wire length. Therefore we choose to use the smaller W_1 and W_2.

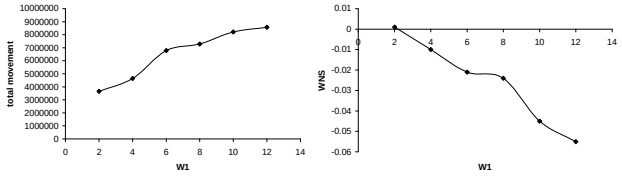

Fig. 12. Total cell movements and worst negative slack with W_1 (assuming $W_1 = W_2$) on ckt2.

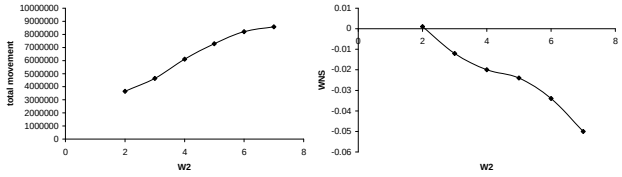

Fig. 13. Total cell movements and worst negative wlack with W_2 (assuming $W_1 = 2$) on ckt2.

Table IX shows the total movement, TWL, WNS and runtime results with different density updating period N_U for $ckt2$. One would imagine that frequently updating the density map could result in a better result (i.e. less movement, better timing), however, our experiment shows this is not the case. Longer updating period N_U results in a little better wire length and similar or even better timing. The reason is following: shorter update period means the density is updated when the cells make smaller moves, therefore the overflow reduction is also smaller. It is easier for the additional overflow introduced by the uneven cell distribution (as explained in section VI-B) to override the smaller overflow reduction. Therefore the chance that cell relative order being violated is higher. More disruption to the relative order will increase the wire length and degrade timing. This phenomenon also tells us that pure optimization for total cell movement does not always optimize timing or wirelength. Furthermore, the runtime of diffusion with shorter period is also higher because it needs to recompute density for more iterations. Therefore, we choose to use longer density updating periods ($N_U = 30$) in our legalizer.

TABLE IX
LEGALIZATION PERFORMANCE WITH DIFFERENT DENSITY UPDATING PERIOD N_U FOR $ckt2$

N_U	movement(10^6)	TWL(10^6)	WNS	CPU(s)
1	3.337	11.575	-0.001	136.6
5	3.409	11.481	-0.002	60.7
10	3.477	11.437	-0.004	37.4
15	3.483	11.423	-0.002	36.5
20	3.826	11.331	-0.006	33.3
25	3.493	11.414	-0.005	32.3
30	3.743	11.339	0.001	34.0
40	3.684	11.341	-0.004	28.2

D. ISPD Benchmarks

Previous sections use proprietary industry circuits to evaluate the performance of the diffusion legalization. In this section, we run a set of experiment on publicly available ISPD 2004 benchmark [17] [23]. First we take the 18 benchmark circuits and place them with Capo [24]. Then we inflate cells to create overlaps. We create two sets of overlapped circuits: $RANDOM$ and $CENTER$. The $RANDOM$ set randomly picks cells to inflate, while the $CENTER$ set only inflates cells in the center of the chip. Therefore $RANDOM$ has a random overlap distribution while $CENTER$ has a centralized overlap distribution. Both inflation procedures only inflate 10% of cells. For each cell, both procedures inflate the cell width by 60%. The resulting overlap percentages and other design characteristics are reported in Table X.

We compare the diffusion legalization (local diffusion) with Capo legalization ($Capo$) [25], FengShui 5.1 ($FengShui$) [26] detailed placement and the latest computational geometry

TABLE X
TESTCASES WIRELENGTHS AND OVERLAPS

testcase	objs	TWL (tracks)	CENTER (%)	RANDOM (%)
ibm01	12506	1.805E+06	5.531	4.941
ibm02	19342	3.947E+06	7.375	5.243
ibm03	22853	5.008E+06	5.003	5.162
ibm04	27220	6.146E+06	5.529	4.746
ibm05	28146	1.002E+07	5.379	4.943
ibm06	32332	5.553E+06	6.356	5.045
ibm07	45639	9.209E+06	5.878	5.124
ibm08	51023	9.665E+06	5.692	5.236
ibm09	53110	1.043E+07	5.675	4.986
ibm10	68685	1.899E+07	5.932	5.044
ibm11	70152	1.534E+07	5.719	5.206
ibm12	70439	2.388E+07	5.1	5.078
ibm13	83709	1.834E+07	5.817	5.359
ibm14	147088	3.445E+07	5.667	5.046
ibm15	161187	4.225E+07	5.792	5.39
ibm16	182980	4.702E+07	5.821	5.326
ibm17	184752	6.730E+07	6.33	5.185
ibm18	210341	5.767E+07	5.543	5.255

placement migration (GEM) [18]. It is not clear to us what legalization algorithm $Capo$ uses, we guess they are greedy heuristics. $FengShui$ uses dynamic programming algorithm [5]. GEM uses a global bin-stretching spreading algorithm followed by detailed legalization. All experiments run on a Linux server with 4 AMD 2.0GHz Opteron CPUs. The wirelength, movement and runtime statistics of legalizations for $CENTER$ testcases are reported in Tables XI, XII and XIII. Statistics for $RANDOM$ testcases are reported in Tables XIV, XV and XVI. The wirelengths are scaled to the original wirelength before legalization. The movement reports include maximum movement, average movement, average square movement, and total number of cells moved. The best result among all legalizers are highlighted in tables.

Clearly, diffusion introduces much less cell movement and wirelength degradation, in particular for $CENTER$, which is consistent with the result of Table VI. On both $CENTER$ and $RANDOM$ testcases, diffusion produces the best maximum movement and average square movement, while $Capo$ produces best number of movements and GEM the best linear movement. The average maximum movements $DIFF$ makes are only about 20% of those made by $Capo$, $FengShui$ and GEM. On $CENTER$ set, the advantage of wirelength of diffusion over other legalizers is clear. On $RANDOM$ testcases, diffusion is as good as $Fenshui$ and slightly better than $Capo$ and GEM. Note that diffusion is designed for placement migration applications, where lots of placement changes need to be made. Legalization for $RANDOM$ set is not a typical application for placement migration. Therefore diffusion does not show much more benefit than other legalization tools on $RANDOM$. However, if we have timing information for these testcases, diffusion could have achieved the best timing result because the maximum movement and average square movement from diffusion are the smallest among all compared legalizers.

The runtime of diffusion legalization is comparable to $Capo$ and $Fengshui$.

One could argue that the legalization engines of Capo and FengShui were not designed to handle overly congested overlaps like $CENTER$ testcases. There are not many legalization algorithms proposed to handle placement migration problems as we defined in the introduction section. To the best of our knowledge, only the recently proposed computation geometry migration algorithm [18] does similar spreading as diffusion.

Diffusion still gives better results than GEM on both $CENTER$ and $RANDOM$ sets although the gap is smaller than those to $Capo$ and $FengShui$. For example, $DIFF$ achieved an average total wirelength of 1.08 on $CENTER$ compared to 1.15 by GEM. And the maximum movement and average square movement of $DIFF$ are also smaller. The biggest advantage of GEM is its speed. Among these four legalizers, GEM runs fastest. Since GEM is not the focus of this paper, here we only give a brief analysis of why GEM is much faster than diffusion. The main reason is that diffusion needs to use smaller time stamp than GEM to make the computation stable. Diffusion algorithm solves the diffusion equation with Forward Time Centered Space (FTCS) [21]. To make it stable, we have to choose a very small time interval such that the stability condition $\frac{|v|\Delta t}{\Delta x} \leq 1$ is satisfied. Therefore it requires more iterations. Second, diffusion uses a much smaller grid size than GEM in our implementations, which means there are more grids to compute but better accuracy of spreading. Last but not least important, as implemented in an existing industry placement tool, diffusion has more runtime overhead than a brand new academic legalizer GEM.

TABLE XI
WIRELENGTH AFTER LEGALIZATIONS ON $CENTER$ TESTCASES
(SCALED TO ORIGINAL WIRELENGTH)

testcase	Capo	FengShui	DIFF(L)	GEM
ibm01	1.30	1.22	**1.10**	1.17
ibm02	1.29	1.28	**1.08**	1.14
ibm03	1.26	1.16	**1.07**	1.15
ibm04	1.19	1.09	**1.06**	1.08
ibm05	1.15	1.11	**1.05**	1.12
ibm06	1.31	1.30	**1.09**	1.14
ibm07	1.28	1.21	**1.08**	1.10
ibm08	1.26	1.14	**1.08**	1.11
ibm09	1.32	1.28	**1.09**	1.14
ibm10	1.21	1.18	**1.08**	1.17
ibm11	1.28	1.24	**1.08**	1.12
ibm12	1.19	1.12	**1.06**	1.09
ibm13	1.44	1.26	**1.09**	1.22
ibm14	1.35	1.22	**1.08**	1.11
ibm15	1.49	1.30	**1.11**	1.21
ibm16	1.45	1.32	**1.10**	1.29
ibm17	1.44	1.29	**1.08**	1.19
ibm18	1.38	1.29	**1.09**	1.14
Average	1.31	1.22	**1.08**	1.15

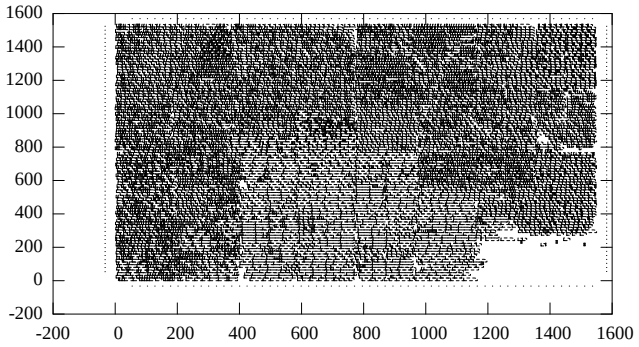

Fig. 14. Placement of $ibm01$.

To further reveal the difference of these legalizers, we plot the movement pictures during legalization for $ibm01$ $CENTER$, which are shown in Figs. 15 - 18. The original placement picture of $ibm01$ is shown in Fig. 14. To make cell movement easy to visualize, we only show the movements over 50 tracks in these figures. The arrows illustrate cell movements during legalization. We can observe that diffusion smoothly moves cells out of the center overlapping region, while $Capo$ and $FengShui$ both make large moves that would change the cell relative order completely. We can observe that GEM makes similar movement as diffusion: both move cells gradually from center to the low-right corners. This is because GEM also uses the density gradient to guide bin stretching, which is similar to diffusion using density gradient to guide cell movement. Note that Fig. 18 also shows some movements

TABLE XII

MOVEMENTS OF LEGALIZATION ON $CENTER$ TESTCASES

testcase	Capo				FengShui				DIFF(L)				GEM			
	max	avg	avg^2	#mov	max	avg	avg^2	#mov	max	avg	avg^2	#mov	max	avg	avg^2	#mov
ibm01	1043	51	12005	**11498**	1558	84	31985	12183	**245**	**48**	**3492**	12428	608	50	3959	12482
ibm02	1204	62	16442	**17537**	1626	134	54221	18745	**289**	**56**	**5389**	18975	1198	58	6499	18646
ibm03	906	52	7796	**18141**	816	92	22188	21722	**182**	**48**	**3323**	22305	1042	49	4861	22568
ibm04	1231	97	22572	**18876**	480	78	12902	25337	**205**	53	**4218**	24453	717	**51**	4389	25220
ibm05	1449	76	22071	**23609**	1019	98	28397	26529	**306**	**64**	**6322**	28026	1473	65	11231	28029
ibm06	1261	101	23577	**26299**	1725	110	32195	31038	**256**	60	**5470**	32185	900	**58**	5959	32089
ibm07	1231	108	31486	**27927**	1967	103	33128	41649	**438**	68	7821	40293	1147	**60**	**6862**	40533
ibm08	1696	93	25606	**40277**	1010	112	37829	45651	**357**	**61**	**5605**	49484	1427	58	6316	49012
ibm09	1467	99	26024	**39381**	2350	126	56857	50838	**312**	67	**7076**	50525	1831	**63**	7529	49943
ibm10	1799	122	40515	**46088**	1503	140	52936	63235	**512**	**92**	**12036**	68160	2797	**92**	16049	68468
ibm11	1478	148	46586	**37390**	2324	124	48361	66649	**385**	75	**9034**	66212	1554	**73**	9640	64291
ibm12	1872	146	49411	**47520**	1380	133	46861	66780	**299**	89	**11049**	69935	2381	**82**	11484	69144
ibm13	2196	126	52085	**67664**	1638	148	68111	81193	**359**	86	**11270**	82281	2443	**82**	14485	79829
ibm14	2999	160	69547	**114808**	3873	214	139152	137135	**440**	118	19676	136043	2432	**98**	**17422**	136665
ibm15	2714	162	81970	**141677**	4395	219	170982	157040	**699**	125	24230	153685	3987	**102**	**22014**	152596
ibm16	3464	187	114484	**160008**	2606	307	257357	168576	**852**	152	**35502**	182707	4420	**110**	41019	176756
ibm17	3701	235	150333	**149376**	2414	325	286307	171093	**1112**	157	**38187**	174864	4101	**122**	42401	171494
ibm18	3294	193	102511	**152276**	2871	244	200385	188118	**576**	126	**23814**	190913	2982	**115**	30926	184771
Average	1945	123	49723	**63353**	1975	155	87786	76306	**435**	86	**12973**	77971	2080	**77**	14614	76808

TABLE XV

MOVEMENTS OF LEGALIZATION ON $RANDOM$ TESTCASES

testcase	Capo				FengShui				DIFF(L)				GEM			
	max	avg	avg^2	#mov	max	avg	avg^2	#mov	max	avg	avg^2	#mov	max	avg	avg^2	#mov
ibm01	966	**17**	**927**	**11483**	1345	28	3759	12201	**226**	26	1219	12289	477	24	1263	12155
ibm02	1187	**21**	1963	**18021**	547	30	3169	18893	**210**	32	1824	19083	340	23	**1051**	18816
ibm03	732	25	1691	**21032**	446	36	2609	22537	**232**	25	**1216**	22531	1261	**22**	3452	21690
ibm04	368	**19**	906	**24042**	265	24	1043	26303	**184**	**19**	**639**	26554	783	**19**	749	25774
ibm05	1584	**20**	**1969**	**26812**	689	25	2396	27513	**332**	33	2373	27688	2663	25	11307	27039
ibm06	672	**20**	1281	**30447**	287	25	1137	31680	**147**	29	1193	31989	824	22	**1013**	31570
ibm07	1412	22	2109	**40840**	1442	25	2249	44475	**292**	37	2009	45135	1467	**20**	**1264**	43254
ibm08	963	15	**538**	**44716**	252	19	653	48932	**222**	25	1080	49788	1394	**14**	1009	47831
ibm09	1094	24	1713	**47663**	1528	33	4060	51853	**237**	31	1605	52396	799	**22**	**958**	51672
ibm10	3333	28	1896	**63758**	601	38	2587	67233	**289**	29	**1674**	67693	3318	**26**	5566	67024
ibm11	2135	35	8157	**65392**	2271	52	23278	69198	**308**	38	2476	69448	1589	**28**	**1613**	69590
ibm12	1346	35	4875	**66249**	1041	36	6104	69399	**383**	51	3618	70226	2557	**27**	**3154**	68259
ibm13	2358	33	6944	**78284**	1271	49	9880	82614	**321**	43	**2776**	83030	2857	**24**	5892	80796
ibm14	986	22	1088	**131484**	350	23	1013	142901	**391**	48	3494	146017	1573	**19**	**976**	142468
ibm15	2530	39	7980	**155340**	2998	48	17976	159200	**563**	78	8922	160686	3616	**31**	**4701**	158719
ibm16	4121	43	16338	**177418**	1868	51	16563	181148	**714**	83	**10253**	182516	5026	**33**	21696	178098
ibm17	1541	33	4321	**171166**	872	38	5710	181276	**666**	62	6312	183338	3851	**26**	**4308**	180395
ibm18	4390	27	1906	**193880**	1129	29	2775	205753	**439**	29	1640	207380	1644	**19**	**1067**	203793
Average	1762	27	3700	**76002**	1067	34	5942	80173	**342**	40	**3018**	80988	2002	**24**	3947	79386

TABLE XIII

CPU TIME (S) OF LEGALIZATION ON $CENTER$ TESTCASES

testcase	Capo	FengShui	DIFF(L)	GEM
ibm01	5	9	8	4
ibm02	9	19	17	4
ibm03	8	20	15	3
ibm04	9	24	15	3
ibm05	29	29	34	7
ibm06	11	34	39	6
ibm07	34	52	46	9
ibm08	53	67	57	13
ibm09	66	65	64	14
ibm10	105	90	80	25
ibm11	52	93	87	21
ibm12	185	97	137	20
ibm13	152	125	279	41
ibm14	387	277	488	68
ibm15	747	339	855	82
ibm16	947	389	1453	104
ibm17	1149	401	1297	105
ibm18	664	482	435	113
Average	256	145	300	36

outside the main flow area, which Fig. 15 does not have. These movements are introduced by the detailed legalization because there are still minor density overflow in those areas even after the main global bin stretching phase.

VIII. CONCLUSIONS

The incremental nature of design optimization demands smooth placement migration techniques. They must be capable of spreading cells to satisfy design constraints such as image space, routing congestion, signal integrity and heat distribution, while keeping the original relative order. To address these tasks, we proposed a diffusion-based placement method. This method takes advantages of smooth movement of the diffusion model. Our experimental results have demonstrated significant improvements on timing and wire length over conventional methods. The future works include applying diffusion technique to other design closure objectives, such

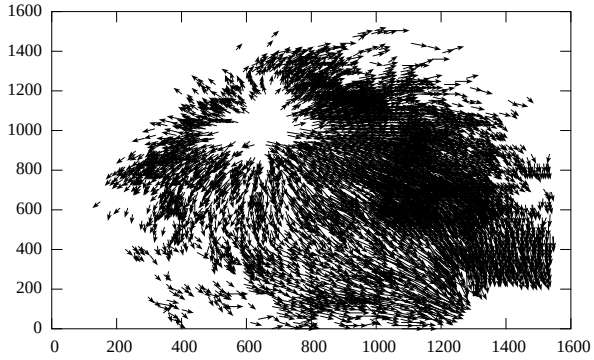

Fig. 15. Cell movement (over 50 tracks) during diffusion legalization on $ibm01$ with $CENTER$ overlap.

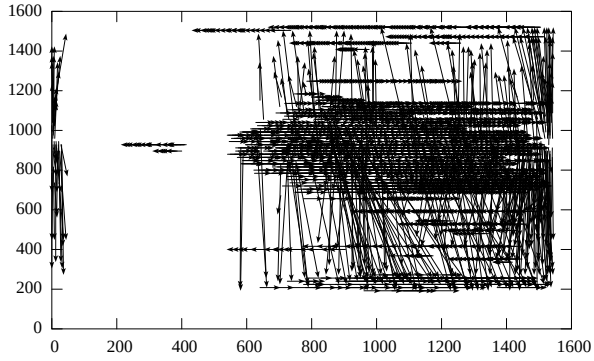

Fig. 16. Cell movement (over 50 tracks) during Capo legalization on $ibm01$ with $CENTER$ overlap.

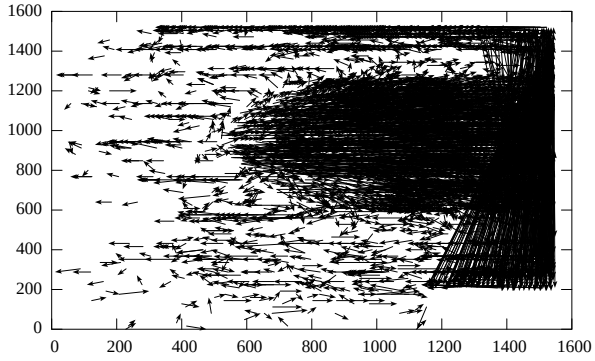

Fig. 17. Cell movement (over 50 tracks) during FengShui detailed placement on $ibm01$ with $CENTER$ overlap.

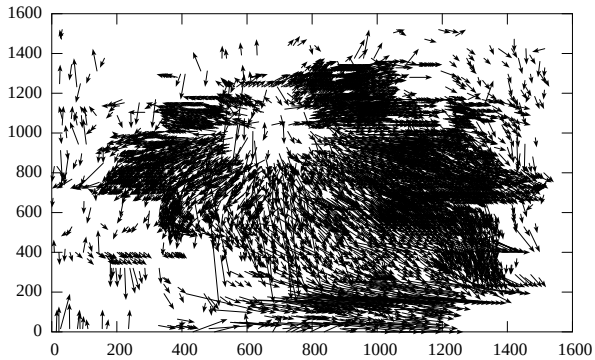

Fig. 18. Cell movement (over 50 tracks) during computational geometry based legalization on $ibm01$ with $CENTER$ overlap.

as routing congestion mitigation and power ground noise reduction.

REFERENCES

[1] H. Ren, D. Z. Pan, and P. Villarrubia, "True crosstalk aware incremental placement with noise map," in *Proc. Int. Conf. on Computer Aided Design*, pp. 616–619, 2004.

[2] H. Ren, D. Z. Pan, C. J. Alpert, and P. Villarrubia, "Diffusion-based placement migration," in *Proc. Design Automation Conf.*, pp. 515–520, 2005.

[3] U. Brenner, A. Pauli, and J. Vygen, "Almost optimum placement legalization by minimum cost flow and dynamic programming," in *Proc. Int. Symp. on Physical Design*, pp. 2–9, 2004.

[4] S. W. Hur and J. Lilis, "Mongrel: hybrid techniques for standard cell placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 165–170, 2000.

[5] A. Agnihotri, M. C. Yildiz, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden, "Fractional cut: improved recursive bisection placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 307–310, 2003.

[6] A. B. Kahng, P. Tucker, and A. Zelikovsky, "Optimization of linear placements for wirelength minimization with free sites," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 18–21, 1999.

[7] A. B. Kahng, I. L. Markov, and S. Reda, "On legalization of row-based placements," in *Proceedings 14th Great Lakes Symposium on VLSI*, pp. 214–219, 2004.

[8] D. Hill, "Method and system for high speed detailed placement," in *US Patent 6370763*, 2001.

[9] M. Sarrafadeh and M. Wang, "Nrg: global and detailed placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 532–537, 1997.

[10] C. Sechen, *VLSI Placement and Global Routing Using Simulated Annealing*. Kluwer Academic Publishers, 1998.

[11] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: Standard-cell placement tool for large industry circuits," in *Proc. Int. Conf. on Computer Aided Design*, pp. 260–263, 2000.

[12] D. F. Wong, H. W. Leong, and C. L. Liu, *Simulated Annealing for VLSI Design*. Kluwer Academic Publishers, 1988.

[13] K. Doll, F. M. Johannes, and K. Antreich, "Deterministic placement improvement with hill-climbing capabilities," in *IFIP International Conference on VLSI*, pp. 91–100, 1991.

[14] C. Li, C.-K. Koh, and P. H. Madden, "Floorplan management: Incremental placement for gate sizing and buffer insertion," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 349–354, 2005.

[15] X. Zhong, J. D. Ma, S. M. Fowler, and R. A. Rutenbar, "Large-scale placement by grid-warping," in *Proc. Design Automation Conf.*, pp. 351–356, 2004.

[16] S. Hur, T. Cao, K. Rajapopal, Y. Parasuram, A. Chowdhary, V. Tiourin, and B. Halpin, "Force directed mongrel with physical net constraints," in *Proc. Design Automation Conf.*, pp. 214–219, 2003.

[17] N. Viswanathan and C. C. N. Chu, "Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model," in *Proc. Int. Symp. on Physical Design*, pp. 26–33, 2004.

[18] T. Luo, H. Ren, C. J. Alpert, and D. Z. Pan, "Computational geometry based placement migration," in *Proc. Int. Conf. on Computer Aided Design*, pp. 41–47, 2005.

[19] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. Design Automation Conf.*, pp. 269–274, 1998.

[20] J. D. Plummer, M. D. Deal, and P. B. Griffin, *Silicon VLSI Technology: Fundamentals, Practice, and Modeling*. Prentice Hall, 2003.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C++*. Cambridge University Press, 2002.

[22] H. Ren, D. Z. Pan, and D. Kung, "Sensitivity guided net weighting for placement driven synthesis," in *Proc. Int. Symp. on Physical Design*, pp. 10–17, Apr. 2004.

[23] N. Viswanathan and C. Chu, "ISPD04 IBM standard cell benchmarks with pads," *http://www.public.iastate.edu/∼nataraj/ISPD04_Bench.html*, 2004.

[24] J. Roy, D. Papa, A.-N. Ng, and I. Markov, "Satisfying whitespace requirements in top-down placement," in *Proc. Int. Symp. on Physical Design*, pp. 206–208, 2006.

TABLE XIV

WIRELENGTH AFTER LEGALIZATIONS ON $RANDOM$ TESTCASES (SCALED TO ORIGINAL WIRELENGTH)

testcase	Capo	FengShui	DIFF(L)	GEM
ibm01	1.11	1.10	**1.09**	1.12
ibm02	1.08	**1.06**	**1.06**	**1.06**
ibm03	1.11	1.07	**1.06**	1.13
ibm04	1.06	**1.03**	1.05	1.05
ibm05	1.04	**1.02**	1.04	1.08
ibm06	1.09	**1.05**	1.07	1.07
ibm07	1.09	**1.04**	1.06	1.08
ibm08	1.05	**1.02**	1.06	1.06
ibm09	1.11	**1.07**	**1.07**	**1.07**
ibm10	1.08	**1.04**	1.06	1.12
ibm11	1.19	1.13	**1.07**	1.08
ibm12	1.09	**1.03**	1.05	1.08
ibm13	1.19	1.09	**1.08**	1.18
ibm14	1.06	**1.02**	1.06	1.06
ibm15	1.16	**1.10**	**1.10**	1.12
ibm16	1.15	**1.07**	1.09	1.26
ibm17	1.08	**1.04**	1.05	1.07
ibm18	1.09	**1.03**	1.07	1.06
Average	1.10	**1.06**	1.07	1.10

TABLE XVI

CPU TIME (S) OF LEGALIZATION ON $RANDOM$ TESTCASES

testcase	Capo	FengShui	DIFF(L)	GEM
ibm01	1	9	8	2
ibm02	4	18	10	2
ibm03	2	20	14	3
ibm04	1	24	14	2
ibm05	8	30	35	6
ibm06	4	34	40	4
ibm07	8	51	48	5
ibm08	5	67	59	7
ibm09	14	63	52	9
ibm10	27	89	53	19
ibm11	26	93	84	14
ibm12	54	93	315	15
ibm13	118	124	265	36
ibm14	24	267	509	25
ibm15	114	327	640	80
ibm16	533	373	880	99
ibm17	226	375	607	60
ibm18	104	460	256	43
Average	71	140	216	24

[25] "Capo tool suite," *http://vlsicad.eecs.umich.edu/BK/PlaceUtils/*.
[26] "Fengshui tool suite," *http://vlsicad.cs.binghamton.edu/*.

Haoxing Ren (S'99-M'00) received the B.S. and M.Eng. degrees in electrical engineering from Shanghai Jiao Tong University, China, in 1996 and 1999, respectively, the M.S. degree in computer engineering from Rensselaer Polytechnic Institute, Troy, NY, in 2000, and the Ph.D degree in computer engineering from the University of Texas, Austin, in 2006.

Dr. Ren currently works as a Research Staff Member at IBM T. J. Watson Research Center, where he is primarily working on the physical synthesis field. He was with design automation department in IBM System and Technology Group from 2000 to 2006, where he has developed placement and timing optimization softwares for ASIC and microprocessor designs. His research interests include placement, physical design and logic synthesis of VLSI systems.

Dr. Ren has received the IBM Outstanding Technical Achievement Award in 2007.

David Z.Pan (S'97-M'00-SM'06) received his Ph.D. degree in computer science from University of California at Los Angeles (UCLA) in 2000. He was as a Research Staff Member at IBM T. J. Watson Research Center from 2000 to 2003. He is currently an Assistant Professor at the Department of Electrical and Computer Engineering, the University of Texas at Austin. His research interests include nanometer physical synthesis, design for manufacturing, vertical integration design and technology, and CAD for emerging technologies.

He has published over 65 technical papers and holds 5 U.S. patents. He is an Associate Editor for IEEE Transactions on CAD (TCAD), IEEE Transactions on VLSI Systems (TVLSI), and IEEE Transactions on CAS-II (TCAS-II). He is in the Design Technology Working Group of International Technology Roadmap for Semiconductor (ITRS). He has served in the program committees of major VLSI/CAD conferences, including ICCAD, ISPD, DATE, and ASPDAC. He is the Program Committee Chair of ISPD 2007, IEEE/CANDE Workshop Chair of 2007, CAD track Co-Chair of ISCAS 2006 and 2007, and Design of Reliable Circuits and Systems Track Chair of ISQED 2007. He is a member of the ACM/SIGDA Technical Committee on Physical Design.

Dr. Pan has received the Best Paper in Session Award from SRC Techcon 1998, IBM Research Fellowship in 1999, Dimitris Chorafas Foundation Award in 2000, SRC Inventor Recognition Award in 2000, Outstanding Ph.D. Award from UCLA in 2001, IBM Research Bravo Award in 2003, IBM Faculty Award from 2004 to 2006, ACM/SIGDA Outstanding New Faculty Award in 2005, Best Paper Award Nominations at DAC 2006 and ASPDAC 2006, and the 2nd prize of ISPD 2007 Global Routing Contest (3D category).

Charles J. Alpert (S'92-M'96-SM'02-F'06) received the B.S. degree in math and computational sciences and the B.A. degree in history from Stanford University, Stanford, CA, in 1991 and the Ph.D. degree in computer science from the University of California, Los Angeles (UCLA), in 1996.

He currently works as a Research Staff Member at the IBM Austin Research Laboratory, Austin, TX, where he serves as the technical lead for the design tools group. He has over 80 conference and journal publications. His research centers upon innovation in physical synthesis optimization.

Dr. Alpert has thrice received the Best Paper Award from the ACM/IEEE Design Automation Conference. He has served as the general chair and the technical program chair for the Tau Workshop on Timing Issues in the Specification and Synthesis of Digital Systems and the International Symposium on Physical Design. He also serves as an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN. For his work in mentoring SRC funded research, he received the Mahboob Khan Mentor Aware in 2001.

Paul G. Villarrubia received the B.S degree in electrical engineering from Louisiana State University, Baton Rouge, in 1981 and the M.S. degree from the Univeristy of Texas, Austin, in 1988.

He is currently a Senior Technical Staff Member at IBM, Austin, where he leads the development of placement and timing closure tools. He has worked at IBM in the areas of physical design of microprocessors, physical design tools development, and tools development for ASIC timing closure. Interests include placement, synthesis, buffering, signal integrity and extraction. He has 21 patents, 20 publications.

Mr. Villarrubia has one DAC best paper award. He was a member of the 2005 ICCAD TPC, and an invited speaker at the 2002 and 2004 ISPD conferences.

Gi-Joon Nam (S'99-M'01) received the B.S. degree in computer engineering from Seoul National University, Seoul, Korea, and the M.S. and Ph.D degrees in computer science and engineering from the University of Michigan, Ann Arbor.

Since 2001, he has been with the international Business Machines Corporation Austin Research Laboratory, Austin, TX, where he is primarily working on the physical design space, particularly placement and timing closure flow. His general interests includes computer-aided design algorithms, combinatorial optimizations, very large scale integration system designs, and computer architecture.

Dr. Nam has been serving on the technical program committee for the International Symposium on Physical Design (ISPD), the International Conference on Computer Design (ICCD), the Asia and South Pacific Design Automation Conference (ASPDAC) and the International System-on-Chip Conference (SOCC). He was also the organizer of ISPD 2005/2006 placement contest.