

Diffusion-Based Placement Migration

Haoxing Ren
IBM Corp. &
Univ. of Texas at Austin
Austin, TX

haoxing@us.ibm.com

David Z. Pan
ECE Department
Univ. of Texas at Austin
Austin, TX

dpan@ece.utexas.edu

Charles J. Alpert
IBM Austin Research
Laboratory
Austin, TX

alpert@us.ibm.com

Paul Villarrubia
IBM Corp.
Austin, TX

pgvillar@us.ibm.com

ABSTRACT

Placement migration is the movement of cells in an existing placement to address a variety of post placement design closure issues. This work presents a new diffusion-based method that has the advantage of smooth spreading which preserves the integrity of the original placement. Furthermore, the mathematics of the diffusion process is a well studied field, with a wide body of formal methods and techniques available to draw upon. Our algorithm takes advantage of a discrete approximation to a *closed form* solution of the continuous diffusion problem formulation. This approach can address the problem of post placement optimization for objectives such as timing, routing congestion, signal integrity, and heat distribution. Our algorithm is also potentially useful as a generic spreading technique to be used in conjunction with analytic or force-directed placement algorithms. In this paper we use the diffusion algorithm to address the problem of placement legalization. Our experimental results show significant improvements in wire length and timing as compared to other commonly used techniques.

1. INTRODUCTION

During placement and physical synthesis of VLSI circuits, one is often faced with tasks such as cell spreading, legalization of overlapping cells, and manipulating the placement to address other physical 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 nature (cell ordering, wirelength, etc.) of the original placement. We call these sets of tasks “placement migration”. Some specific examples of placement migration include:

- During physical synthesis, one may insert buffers and re-power gates, creating overlapping cells. The instance needs to be legalized while preserving the original placement characteristics.
- After placement, it may be necessary to make Engineering Change Orders (ECO) or insert decoupling capacitors which requires spreading to resolve overlaps induced by these changes.
- Post placement routing analysis determines several hot spots of routing congestion or crosstalk noise. Placement migration can locally spread out cells in these congested or noisy regions.
- An analytic or force-directed global placer uses placement migration to spread out the optimal, yet overlapping, squared wirelength solution.

In this paper, we propose a new technique for placement migration based on the physical process of diffusion. Diffusion is a well-understood process that moves a physical elements (such as air molecules) from a state with non-zero potential energy to a state of equilibrium. The process can be modeled by taking several small finite time steps and moving each element the distance it would be expected to move in that time step. Our approach to placement migration does just that; 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 to achieving equilibrium. The main advantages to this approach are

- It spreads the placement *smoothly* which is more likely to preserve the integrity of the original placement as opposed to techniques which move or swap cells.
- It changes the placement incrementally and thus can be run only as long as necessary to achieve the desired objective.
- Its implementation is both simple and efficient.

Existing techniques that can be potentially used for placement migration are cell expansions [1] [2], flow based legalization [3], heuristic ripple cell movement [4], and single row optimization [5] [6]. Cell expansion [2] enforces spreading in global placement by expanding the cells by some amount, which forces more white space inside the regions where cells were artificially expanded. It still requires to run global placement, which is slow and non-incremental, and would potential change the design characteristic. In an effort to maintain the placement stability, [1] proposed to start placement from a later cut. However, it would still affect local placement relative order. The network flow approach [3] uses minimum cost flows and dynamic programming to minimize the weighted sum of (squared) movements. Although the flow solution can give the amount of cells that need to be moved between adjacent bins, it is not trivial to decide which cell to move and whether a cell just moved into a bin should be moved again or not. Therefore it would also change the local relative order during the flow realization phase. Mongrel [4] uses a greedy heuristic to move cells from overflowed bins to under capacity bins in a ripple fashion based on total wire length (TWL) gain computed for each move between adjacent bins. Although the ripple move helps maintain relative order, like flow based legalization, this approach can not directly determine the locations of cells when they are moved to another bin. Therefore the gain computation is somewhat unrealistic when lots of cells move away from their original locations, or when cells need to move more than one bin. This approach is more like a greedy realization phase with a simple network flow algorithm. The single row optimization techniques [5] [6] use dynamic programming to optimally place cells in a single circuit row. However, they can

not handle global placement changes that require cells to move between rows. Although all these techniques can be used to perturb the placement, the perturbation is rather discrete compared to the diffusion-based method that is much more *continuous*.

We also need to mention forced directed technique proposed by Eisenmann and Johannes [7] in the context of global placement. Force directed approach generates spreading forces from global density distribution and uses it in the quadratic optimization formulation. This spreading technique models the density map as an electric field whereby every region of the density map has some attraction or repulsion to every cell in the design. The diffusion based approach generates cell velocities from local density distribution and use integral equations of velocities to directly compute cell location. There are several major differences of force directed approach (FORCE) and diffusion based approach (DIFF):

- FORCE determines the cell coordinates by solving the linear system generated by cell connections and spreading forces, while DIFF computes cell location by solving the integral equation of the diffusion velocities.
- FORCE computes spreading forces from the global density distribution, while DIFF only requires *local* density information to compute velocities.
- FORCE can not achieve density equalization without the help of iterations, while DIFF is a closed-form formulation, which can spread out the design without even recompute the real cell density. The density-time relationship can be solved by the diffusion equation alone.
- Electric force model can not be used in diffusion process, but diffusion velocity actually can be used as spreading force for force directed algorithm. It satisfies all the four requirements for the spreading force [7].
- It is hard to apply the force-directed approach to placement migration, which does not start from scratch but from an existing placement.

Among all the placement migration applications, the most straightforward one is legalization. Therefore we will use legalization to describe the detail of diffusion method. The rest of the paper is organized as follows. Section 2 describes the formulation of the diffusion-based placement migration in the context of legalization. Section 3 gives the numerical method to simulate the diffusion process. Section 4 describes the proposed diffusion-based legalization algorithm and its implementation details. The experimental results are shown in section 5, followed by the conclusion in section 6.

2. PROBLEM FORMULATION

In this section, we first give the formulation of placement migration using the example of legalization, then introduce the diffusion process for placement migration.

2.1 Placement Migration for Legalization

Suppose we divide the chip area into N equal sized bins. If the chip has a width of W and height of H , the density $d_{j,k}$ of each bin (j, k) can be defined as:

$$d_{j,k} = \frac{N \sum \hat{A}_i}{WH} \quad (1)$$

where \hat{A}_i is the overlapping area of cell i and bin (j, k) . For simplicity, we assume the fixed macros either totally occupy a bin or not, therefore the density for a bin on a fixed macro is always 1.

If all the bins have density less than or equal to 1, we can easily produce a legal placement by simply snap cells into the closest legal location. It is difficult when the bin density is larger than 1, which means cells need to move out of their own bins. Those cells would end up far away from their original locations and affect the relative order. They will also increase the density of other bins and force cells in those bins to move, causing a domino effect. Therefore the objective of placement migration for legalization is to generate a placement that will make the density of each bin under 1, in other words, the objective is to achieve a maximum density of $d_{max} = 1$. Meanwhile, the new placement should be similar to the original illegal placement in order to keep the original design characteristics. Cells should gradually move between adjacent bins during the migration process instead of jumping to bins far away. Therefore the placement mitigation for legalization should smoothly move cells from bins over d_{max} to those under d_{max} .

The problem of placement migration for legalization can be described as: Given an existing placement (x_i, y_i) for each cell i , how to gradually move cells to produce a new placement (x_i^*, y_i^*) such that the maximum density $d_{j,k}$ is less than or equal to d_{max} .

This process is similar to the diffusion process, which moves material from high concentration area to less concentrated area. Naturally, we can formulate the placement migration process under the diffusion law, which is given in next section.

2.2 Diffusion Process

Although diffusion process has been heard little in VLSI physical design area, it is not unfamiliar at all for the semiconductor industry. The dopant diffusion process on chip substrate is a well known diffusion process [8]. Intuitively, materials from highly concentrated areas would 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. And 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. The final equilibrium of diffusion is an equal concentration distribution.

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

$$\frac{\partial d}{\partial t} = D \nabla^2 d \quad (2)$$

where d is the material concentration, D is the diffusivity which determines the speed of diffusion (For the rest of the work, we set D to 1 for the simplicity of presentation). It states that the speed of density change is linear to its second order gradient over space.

In the context of placement, material concentration can be defined as the placement density $d_{x,y}(t)$. Note that, unlike bin density $d_{j,k}$ which is defined on discrete bin coordinates (j, k) , $d_{x,y}$ is defined on continuous space coordinates (x, y) .

The boundary conditions are $\nabla d_{x_b, y_b}(t) = 0$ for coordinates (x_b, y_b) on the chip boundaries or boundaries of fixed macros, because cells can not move out of boundary.

In diffusion a cell migrates from an initial location point to its final equilibrium location via a very 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, sharp density gradient or sparse density will cause cells to move faster. We can define a

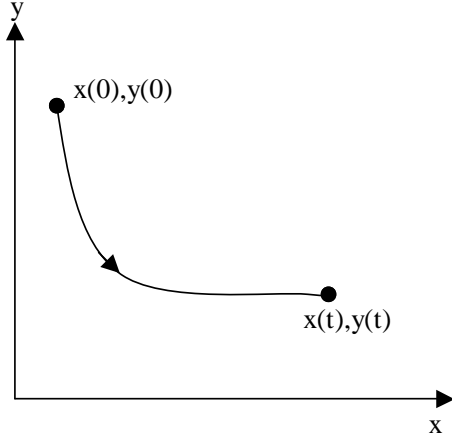


Figure 1: Cell Trajectory in the Velocity Field.

velocity field $\mathbf{v}_{x,y} = (v_{x,x,y}, v_{y,x,y})$ of diffusion at time t , which can be computed as:

$$\begin{aligned} v_{x,x,y}(t) &= -\frac{\partial d}{\partial x} \\ v_{y,x,y}(t) &= -\frac{\partial d}{\partial y} \end{aligned} \quad (3)$$

Therefore, as shown in Fig 1, starting from a initial location $(x(0), y(0))$, the cell location $(x(t), y(t))$ at time t can be calculated by integrating the velocity field thus:

$$\begin{aligned} x(t) &= x(0) + \int_0^t v_{x,x(t'),y(t')}(t') dt' \\ y(t) &= y(0) + \int_0^t v_{y,x(t'),y(t')}(t') dt' \end{aligned} \quad (4)$$

With (2), (4) and (3), we can incrementally change a placement based on the continuous density distribution. However, the continuous form of these equations can not be directly used in placement migration, which has a discrete bin structure. Therefore, in next section, we will give a discrete approximation for these equations.

3. DIFFUSION DISCRETIZATION

The discretization of continuous coordinates is similar to divide the chip image into bins. The width B_w and height B_h of a bin are the steps on horizontal and vertical directions, respectively. We can also discretize continuous time t into discrete time n where $t = n\Delta t$. In the rest of the paper, we use bin coordinates (j, k) instead of the continuous coordinates (x, y) (For the simplicity of the presentation, we assume $B_w = B_h = 1$), and discrete time step n instead of continuous time t .

Assume that the density $d_{j,k}(n)$ has already been computed for time n . Next we need to find how the density changes and cells move during the next time step $n + 1$. Suppose we use Forward Time Centered Space (FTCS) [9] scheme to discretize (2) (assuming $D = 1$):

$$\begin{aligned} d_{j,k}(n+1) &= d_{j,k}(n) + \frac{\Delta t}{2}(d_{j+1,k}(n) + d_{j-1,k}(n) - 2d_{j,k}(n)) \\ &\quad + \frac{\Delta t}{2}(d_{j,k+1}(n) + d_{j,k-1}(n) - 2d_{j,k}(n)) \end{aligned} \quad (5)$$

Starting from $d_{j,k}(0)$, we can compute density $d_{j,k}(n)$ at each bin (j, k) for any time n using (6). For example, let us consider

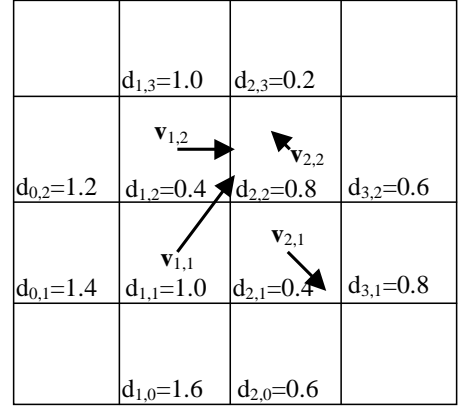


Figure 2: Density and Velocity Distribution at Time n .

a density distribution at a given time n shown in Fig 2, assuming $\Delta t = 0.2$. The density of bin (1, 2) at time $n + 1$ can be computed as:

$$\begin{aligned} d_{1,2}(n+1) &= d_{1,2}(n) + \frac{0.2}{2}(d_{2,2}(n) + d_{0,2}(n) - 2d_{1,2}(n)) \\ &\quad + \frac{0.2}{2}(d_{1,3}(n) + d_{1,1}(n) - 2d_{1,2}(n)) = 0.64 \end{aligned}$$

After we get $d_{j,k}(n)$, we can then compute the velocity of each bin by discretizing (3).

$$\begin{aligned} v_{x_{j,k}}(n) &= -\frac{d_{j+1,k}(n) - d_{j-1,k}(n)}{2d_{j,k}(n)} \\ v_{y_{j,k}}(n) &= -\frac{d_{j,k+1}(n) - d_{j,k-1}(n)}{2d_{j,k}(n)} \end{aligned} \quad (6)$$

The velocity for bin (1, 1) in Fig 2 can be computed as:

$$\begin{aligned} v_{x_{1,1}} &= -\frac{d_{2,1} - d_{0,1}}{2d_{1,1}} = 0.5 \\ v_{y_{1,1}} &= -\frac{d_{1,2} - d_{1,0}}{2d_{1,1}} = 0.6 \end{aligned}$$

Similarly we can get $\mathbf{v}_{2,1} = (0.25, -0.25)$, $\mathbf{v}_{2,2} = (-0.125, 0.125)$, and $\mathbf{v}_{1,2} = (0.5, 0)$. Keep in mind that at horizontal boundary $v_y = 0$, and at vertical boundary $v_x = 0$.

To get the cell placement, we need to compute the integral equation (4). First we use the discretized form $(x(n), y(n))$ instead of $(x(t), y(t))$, where $n = \frac{t}{\Delta t}$, to represent the location of a cell. Then (4) can be transformed into a recursive form. Suppose we have already computed $(x(n-1), y(n-1))$, we can simply use Taylor expansion to compute $x(n), y(n)$.

$$\begin{aligned} x(n) &= x(n-1) + v_{x_{x(n-1),y(n-1)}}(n-1)\Delta t \\ y(n) &= y(n-1) + v_{y_{x(n-1),y(n-1)}}(n-1)\Delta t \end{aligned} \quad (7)$$

3.1 Velocity Interpolation

One problem with the proposed approach is that every cell within a bin has the same velocity and will thus get the same displacement. Meanwhile a cell just across the bin boundary will get a totally different velocity. So before we assign the velocity of a bin to a velocity of a cell, we use interpolation. As shown in Fig. 3, the bin velocity will be marked at the lower left corner of each bin. The velocity for a point inside of a bin is interpolated by the velocities at the four corners of this bin. Given a cell at (x, y) which is inside

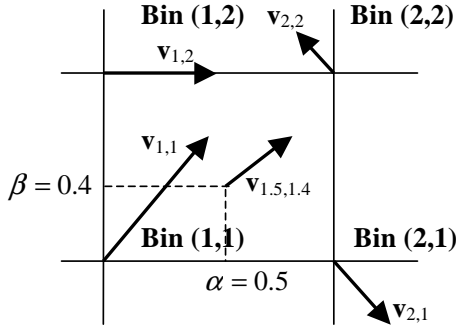


Figure 3: Velocity Interpolation inside Bin.

of bin (j, k) , where $j < x < j + 1$, $k < y < k + 1$, we can compute $vx_{x,y}$ and $vy_{x,y}$ using following interpolation:

$$\begin{aligned}
 vx_{x,y} &= vx_{j,k} + \alpha(vx_{j+1,k} - vx_{j,k}) + \beta(vx_{j,k+1} - vx_{j,k}) \\
 &\quad + \alpha\beta(vx_{j,k} + vx_{j+1,k+1} - vx_{j+1,k} - vx_{j,k+1}) \\
 vy_{x,y} &= vy_{j,k} + \alpha(vy_{j+1,k} - vy_{j,k}) + \beta(vy_{j,k+1} - vy_{j,k}) \\
 &\quad + \alpha\beta(vy_{j,k} + vy_{j+1,k+1} - vy_{j+1,k} - vy_{j,k+1}) \quad (8)
 \end{aligned}$$

where $\alpha = x - j$ and $\beta = y - k$.

For the example shown in Fig 3, which is the zoom-in picture of bin (1, 1) in Fig 2. We calculate the velocity at $(x = 1.5, y = 1.4)$ with $\alpha = 0.5$, $\beta = 0.4$,

$$\begin{aligned}
 vx_{1.5,1.4} &= vx_{1,1} + 0.5(vx_{2,1} - vx_{1,1}) + 0.4(vx_{1,2} - vx_{1,1}) \\
 &\quad + 0.2(vx_{1,1} + vx_{2,2} - vx_{2,1} - vx_{1,2}) = 0.3 \\
 vy_{1.5,1.4} &= vy_{1,1} + 0.5(vy_{2,1} - vy_{1,1}) + 0.4(vy_{1,2} - vy_{1,1}) \\
 &\quad + 0.2(vy_{1,1} + vy_{2,2} - vy_{2,1} - vy_{1,2}) = 0.13
 \end{aligned}$$

4. ALGORITHM

In this section, we first give the diffusion-based legalization algorithm, then describe its implementation details such as how to manipulate initial density to satisfy the maximum density constraint, and how to deal with boundary conditions. The complexity analysis is given at the end.

4.1 Diffusion-Based Legalization Algorithm

The input of the diffusion-based legalization algorithm are locations (x_i, y_i) of each cell i , maximum bin density d_{max} , bin number N and diffusion time T . It first computes the initial bin density using the given placement, then manipulates the density map to avoid over spreading, which will be explained in section 4.2. Starting from time 0, it recursively compute bin density, bin velocity and cell locations for each time step n . It stops after T iterations or 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 max density of d_{max} and is roughly legal. We need to run a final legalization step to put cells onto circuit rows 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 4 shows an example of diffusion-based legalization in a small region surrounded by fixed blocks. The left picture shows the initial illegal placement. The right picture is the placement out of legalization. Cells are colored to represent their relative order. We can see after diffusion, the relative orders are not changed.

Algorithm 1 Diffusion-based Legalization Algorithm

Inputs: cell locations (x_i, y_i) , max density d_{max} , bin number N , diffusion time T

- 1: map cells onto bins and compute $d_{j,k}$ for each bin (j, k)
- 2: compute $\tilde{d}_{j,k}$ using (11), 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 $vx_{j,k}(n), vy_{j,k}(n)$ for each bin (j, k) using (6)
- 7: compute $x_i(n), y_i(n)$ for each cell i using (7) and velocity interpolation (8)
- 8: compute $d_{j,k}(n+1)$ for each bin (j, k) using (6)
- 9: $n \leftarrow n + 1$
- 10: **until** $n = T$ OR $\max(d_{j,k}(n)) \leq d_{max} + \Delta$

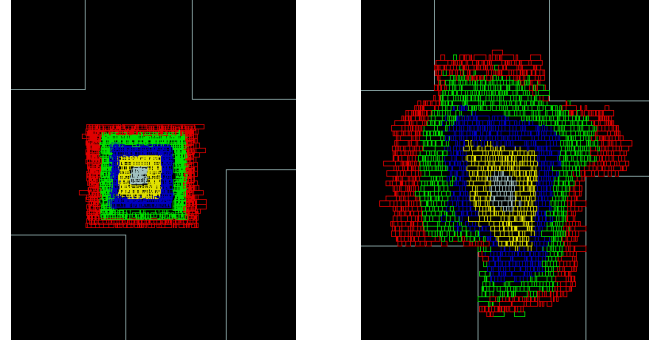


Figure 4: Diffusion-based Legalization Example.

4.2 Initial Density Generation

Since the diffusion process will generate an equal density placement when it reaches the equilibrium, we can expect the the final density after diffusion is average density $\frac{\sum d_{j,k}}{N}$ of the initial densities, which means not only the cells in bins above 1 will expand, those in bins above average will expand as well. This is unnecessary and would cause the new placement far different than the original placement. Therefore, to satisfy the maximum density constraints while not overly expand bins, we have to change the density values of those bins under the maximum density. Suppose we want to achieve the maximum density d_{max} for the equilibrium, the total area A_o that need to spread out of bins over d_{max} are:

$$A_o = \sum \max(d_{j,k} - d_{max}, 0) \quad (9)$$

The total slack A_s that can be used to hold A_o is:

$$A_s = \sum \max(d_{max} - d_{j,k}, 0) \quad (10)$$

If we can change $d_{j,k}$ for those bins under d_{max} to make $A_s = A_o$, then at the equilibrium only the overlaps A_o will move to A_s , and the densities of all the bins will be under d_{max} . One way to adjust $d_{j,k}$ is

$$\tilde{d}_{j,k} = \begin{cases} d_{max} - (d_{max} - d_{j,k}) \frac{A_o}{A_s} & d_{j,k} < d_{max} \\ d_{j,k} & d_{j,k} \geq d_{max} \end{cases} \quad (11)$$

We can validate that the new $\tilde{A}_s = \sum \max(d_{max} - \tilde{d}_{j,k}, 0) = A_o$. Fig 5 shows an example of the density manipulation for a 2×2 bins. In the left figure, there are 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 $A_o = 0.3$ and $A_s = 0.6$. If we adjust the density on those two bins

$d_{1,0}=1.3$	$d_{1,1}=0.6$	→	$d_{1,0}=1.3$	$d_{1,1}=0.8$
$d_{0,0}=1$	$d_{0,1}=0.8$		$d_{0,0}=1$	$d_{0,1}=0.9$

Figure 5: Initial Density.

		$d_{3,6}=1.0$	$d_{4,6}=0.2$	
$d_{2,5}=1.2$	$d_{3,5}=0.4$	$d_{4,5}=0.8$	$d_{5,5}=0.6$	
$d_{2,4}=1.4$	$d_{3,4}=0.8$	$d_{4,4}=1.0$	$d_{5,4}=1.0$	
		$d_{3,3}=1.6$	$d_{4,3}=1.0$	$d_{5,3}=1.0$

Figure 6: Boundary Condition.

under 1 with (11), we will get the density map shown on the right: $\tilde{A}_s = A_o$.

$\tilde{d}_{j,k}$ will be used as the initial condition ($t = 0$) for the diffusion equation (6)

$$d_{j,k}(0) = \tilde{d}_{j,k} \quad (12)$$

4.3 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

$$\nabla d_{j,k}(n) = 0 \quad (13)$$

where (j, k) are bins on the boundaries of the chip and fixed macros. Therefore, we need to make the densities on both sides the same to assure the density gradient is zero when computing (6). On a horizontal boundary, we make $d_{j,k+1}(n) = d_{j,k-1}(n)$, while on a vertical boundary, $d_{j+1,k}(n) = d_{j-1,k}(n)$. 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. (6). When computing $d_{3,4}(n+1)$, we replace $d_{4,4}(n)$ with $d_{2,4}(n)$ to satisfy the vertical boundary conditions.

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

Similarly, we replace $u(4, 4, n)$ with $u(4, 6, n)$ when computing $u(4, 5, n+1)$ to satisfy the horizontal boundary conditions:

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

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

4.4 Complexity Analysis

The complexity of computing the density and velocity for each bin at any given time is $O(N)$, where N is the total number of bins. Since the diffusion usually take $O(N)$ time to close to equilibrium, we need to compute $O(N)$ timing points¹, therefore the complexity of updating density and velocity matrixes is $O(N^2)$. If there are K cells in the design, we need to compute $O(N)$ time steps to get their final locations. The complexity of computing the cells locations are $O(KN)$. Therefore, the overall complexity is $O(N^2 + KN)$. We can see that N can not be too large for the runtime consideration. On the other hand, we want to choose a large N , such that the bin size will be small enough to reflect the detailed density information.

One way to solve this conflict is to run the diffusion process on subsets of the chip image. Since the area of each subset is smaller than the chip area, we can use a smaller N and still get a small bin size. If we divide the chip into S subsets, the overall complexity is $O(S(\frac{N}{S})^2 + K\frac{N}{S})$. In addition to the runtime advantage, using subset can help to make the local diffusion. Since the diffusion is limited in a subset, it will not move cells to other subsets which could happen if the diffusion is to be performed globally. The downside of this approach is that the diffusion in one subset can not migrate into the other subset even if it needs to, which might result in suboptimal solution. Therefore we need to be able to dynamically resize the subset to make sure the available area is bigger than the overlaps.

5. EXPERIMENTAL RESULTS

In this section, we report the experimental results of diffusion-based legalizer (*DIFF*). We evaluate its merit by comparing it with other legalizers, i.e. a greedy legalizer (*GREED*) which uses slide-and-spiral techniques to place cells onto their nearest legal locations, and a network flow legalizer (*FLOW*) which uses min-cost flow algorithm to direct cell movements. *FLOW* is similar to [3], which includes two steps: first cells are roughly spread out by the min-cost flow algorithm, then, in a second step they are moved to their final positions such that all overlaps are removed. *GREED* sorts all the cells and place 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 three legalizers are implemented in *C* and run on a IBM P690 server. The timing result are reported by IBM Einstimer.

Table 1: 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

We use 7 industrial circuits for comparison. The sizes of circuits range from 64K cells to over a million cells. All the circuits were legally placed initially. To simulate the behavior of repowering in physical synthesis we inflate cells which creates overlaps that need to be resolved. This technique can also be used to reduce

¹If we choose a bigger Δ for the max density constraint in Algorithm 1, it can be significantly less than $O(N)$.

routing congestion using diffusion to resolve overlap removal from inflating cells in congested regions. The circuit sizes and amount of inflations generated are reported in Table 1 (the inflations are reported as the percentage of inflation to the total moveable cell areas).

Table 2: TWL Comparison of Three Legalizers (m)

testcases	Base	GREED	FLOW	DIFF	%improv
ckt1	11.48	13.23	13.40	12.46	44
ckt2	15.06	17.03	17.33	16.65	19
ckt3	47.10	52.47	52.65	51.76	13
ckt4	51.37	59.02	58.67	56.85	25
ckt5	150.8	159.0	159.2	158.7	3
ckt6	166.6	175.6	175.4	174.8	8
ckt7	367.7	382.7	382.5	381.7	5
Average					17

Table 3: Worst Slack Comparison of Three Legalizers (ns)

testcases	Base	GREED	FLOW	DIFF	%improv
ckt1	-0.571	-1.266	-1.497	-0.921	50
ckt2	0.275	-0.287	-0.789	-0.065	40
ckt3	-0.265	-1.155	-1.121	-0.265	100
ckt4	-1.592	-3.569	-3.447	-2.373	58
ckt5	-0.623	-6.072	-3.640	-2.047	52
ckt6	-0.387	-3.450	-3.562	-3.305	5
ckt7	-0.796	-1.601	-1.274	-0.796	100
Average					45

Table 2, 3, and 4 show the *TWL*, worst slack and *FOM*² results of the three legalizers. Since we inflate cells, the new placement have longer wire length, worst slack and *FOM* than those of the original placements. The *%improv* column of each table reports the improvement of *DIFF* over the best result of *FLOW* and *GREED*. We can see that *DIFF* achieves significantly smaller *TWL* than *FLOW* or *GREED*. The average improvement of *TWL* over seven circuits are 17%. The average improvement on wiring congestion is 27% (detailed number for each circuit not shown due to space limit). The slack and *FOM* degradation of *DIFF* is also significantly less than those of *FLOW* and *GREED*. The average improvement of *DIFF* over the best of *FLOW* and *GREED* is 45% of slack and 36% for *FOM*. 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 three legalizers on industry circuits generated by physical synthesis with overlapping mode, the results show that *DIFF* gives better timing, *TWL* and congestion for those circuits. Due to page limit, those result are not shown.

Table 5 reports the runtimes for three legalizers. The runtime of *DIFF* is about 2X of *FLOW*. It takes a little more than an hour to legalize a 1M cells circuit, which is still acceptable. To reduce the runtime, we can use a bigger subset size, smaller *N*, or shorter *T* for diffusion. We can also easily reduce the runtime by tuning our C/C++ implementation (*GREED* and *FLOW* are heavily tuned codes).

We also test *FLOW* and *DIFF* with different inflation distributions to see whether *DIFF* is better on distributed inflations

²*FOM* is a metric that measure 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.

Table 4: FOM Comparison of Three Legalizers (ns)

testcases	Base	GREED	FLOW	DIFF	%improv
ckt1	-3188	-4942	-8441	-3883	60
ckt2	0	-247	-620	-319	-29
ckt3	-446	-1073	-1054	-524	87
ckt4	-557	-1321	-1068	-862	40
ckt5	-144	-4827	-4871	-3069	38
ckt6	-15286	-24694	-24154	-22936	14
ckt7	-2583	-5391	-7631	-4157	44
Average					36

Table 5: CPU Time Comparison of Three Legalizers (s)

testcases	GREED	FLOW	DIFF
ckt1	161	55	107
ckt2	41	24	74
ckt3	228	197	290
ckt4	320	313	581
ckt5	414	584	841
ckt6	619	626	1231
ckt7	2102	1768	4681

or concentrated inflations. Table 6 shows the results of *DIFF* and *FLOW* on the same circuit *ckt1* but with different inflation distributions. The inflations are centralized (*C*), or distributed almost evenly (*D*). The amounts of inflations are 23% and 18% for centralized and distributed cases, respectively. The centralized inflation mimics a hot-spot that need to be spread out. The distributed inflation is more like legalization after physical synthesis. Δ column reports the difference of *C* and *D*. Both *DIFF* 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 see that *DIFF* is less sensitive to the inflation distribution than *FLOW*. The *TWL* degradation of *DIFF* is only 0.16m compared to 1.06m of the *FLOW*. The slack and *FOM* degradation of *DIFF* 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 6: Inflation Distribution Effect on Legalization

type(%)	TWL (m)		Slack (s)		FOM(s)	
	FLOW	DIFF	FLOW	DIFF	FLOW	DIFF
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

6. CONCLUSIONS

The incremental nature of design optimization demands smooth placement mitigation techniques. They must be capable of spreading cells to satisfy design constrains 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 method. This method inherits the characteristics of local movement and incrementality of a physical diffusion process. And the similarities between the physical process of diffusion which move material from high concentration area to low concentration area, and the placement migrations such as legalization which move cells from overlapping area to under occupied

area, congestion mitigation which move cells from congested area to non-congested area, etc. make the diffusion method very attractive. The experiment result on legalization problem has demonstrated very significant improvements on timing and wire length over conventional methods.

7. 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] U. Brenner and A. Rohe, "An effective congestion driven placement framework," in *Proc. Int. Symp. on Physical Design*, pp. 6–11, 2002.
- [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. Lillis, "Mongrel: hybrid techniques for standard cell placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 165–170, 2000.
- [5] 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.
- [6] U. Brenner and J. Vygen, "Faster optimal single-row placement with fixed ordering," in *Proc. Design, Automation and Test in Europe*, pp. 117–121, 2000.
- [7] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. Design Automation Conf.*, pp. 269–274, 1998.
- [8] J. D. Plummer, M. D. Deal, and P. B. Griffin, *Silicon VLSI Technology: Fundamentals, Practice, and Modeling*. Prentice Hall, 2003.
- [9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C++*. Cambridge University Press, 2002.