FAR: Fixed-Points Addition & Relaxation Based Placement

Bo Hu
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-5678
hu@cornet.ece.ucsb.edu

Malgorzata Marek-Sadowska
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-2721
mms@ece.ucsb.edu

Abstract

In this paper we describe the Fixed-points Addition and Relaxation (FAR) based placement technique. Fixed point is a pseudo cell connected to a movable cell. By introducing fixed points, the placement can be maintained in a force equilibrium state and further transformed into another equilibrium state. By relaxing some of the previously introduced fixed points, we can partially or completely collapse the current placement in order to reposition the cells, or incrementally perturb the existing good solution to fulfill additional requirements. We apply the FAR-based approach to global placement for total wire length minimization, and to incremental placement for Buffer Site Generation (BSG). For global placement, our experimental results show that the FAR method achieves 54.4% CPU speedup and total wire length comparable to that achieved by the constant force based approach [1]. Experimental results indicate that to accommodate buffers in specific regions, FAR is able to perturb incrementally a given solution in a well-controlled way.

1. Categories and Subject Descriptors

J.6 Computer-Aided Engineering - Computer-aided design (CAD)

2. General Terms

Algorithms.

3. Introduction

Placement is a critical step in the physical design of VLSI circuits. The quality of a placement solution determines whether the desired objectives - such as area, timing, or congestion - can be achieved. Due to placement's importance in the design flow, placement algorithms have been researched continuously for over three decades. Numerous placement techniques have been proposed and proved effective in practice. These include simulated annealing [8](Timberwolf), min-cut [9][10], force-directed and quadratic placement[1][2][3]. Typically, a placement task is divided into two steps: global and detailed. In global placement, relative locations of the cells are determined but some overlapping and illegal cell positions are tolerated. In detailed placement, positions of cells are legalized and optimized. In a different category are incremental placers, which transform the existing placements to accommodate the incremental changes in the netlist or attempt to fulfill additional requirements. The quality of such placers is measured by how close the obtained solution is to the initial one.

In VLSI circuit design, to achieve timing closure and to handle signal integrity problems, buffer insertion is considered essential. Many papers postulate that buffers must be planned in the early design stages. In [4][5][7], buffer block planning methodologies were proposed to meet the buffer demand. But as indicated in [6], buffer block planning has two intrinsic drawbacks: (1) The planned buffer blocks might not be usable due to congestion. (2) Good buffer sites may not be found in the existing buffer blocks. To overcome these two drawbacks, we formulate a new buffer planning methodology called Buffer Site Generation (BSG). The goal is to transform a given placement incrementally, in order to introduce enough buffer sites at the locations required by the buffer insertion algorithm.

In this paper, we describe the Fixed-points Addition and Relaxation (FAR) placement technique. Fixed points are pseudo cells connected to the real cells in the netlist in order to keep the existing placement in a state of equilibrium. By introducing and relaxing the fixed points, the existing placement can be maintained in force equilibrium state and perturbed into another equilibrium state according to specific requirements. The FAR-based approach can be applied to global or incremental placement problems. We compare global placement formulations using fixed points and constant forces [1]. To test FAR in the incremental placement context, we applied it to solve the Buffer Site Generation (BSG) problem. Experimental results show that the FAR-based placer is very effective and efficient in solving both global and incremental placement problems.

Since our method can be viewed as a series of gradually resolving cell overlaps, the most relevant previous works are [1][3]. [1] proposed constant density-induced forces to maintain force equilibrium state and evenly distribute the cells. In [3], an Attractor-Repeller(AR) model was formulated to eliminate overlapping. AR has hard constraints on the distances (Target Distances) between cells and uses fixed dummy cells (attractors) to pull the cells out of the dense area. We will show that fixed points generalize the idea of constant forces and can be used to maintain and perturb the existing placement, as well as to constrain the movement of cells during incremental changes. In our formulation, the hard constraints on the distances between cells [3] can be removed.

The rest of the paper is organized as follows: In section 2, a traditional quadratic formulation is briefly reviewed. In section 3, the fixed points addition and relaxation concepts are detailed and related to quadratic placement. In sections 4 and 5, FAR based approaches are applied to global and incremental placement problems. Experimental results are given in section 6, followed by the conclusions in section 7.

4. Quadratic Placement Preliminary

Let n denote the number of movable cells in the circuit and (x_i, y_i) the coordinates of the center point of a cell i. A quadratic placement problem minimizes the cost function, which is a summation of the squares of interconnection lengths as shown in EQ1:

$$cf = \sum_{i,j} w_{ij}\left((x_i - x_j)^2 + (y_i - y_j)^2\right) \qquad \text{(EQ1)}$$

The circuit connectivity is modelled by a graph. Cells correspond to vertices, and nets are modeled as edges. Multi-pin nets are modeled by cliques. Each edge in the clique connecting cell i and j con-

ISPD'02, April 7-10, 2002, San Diego, California, USA.

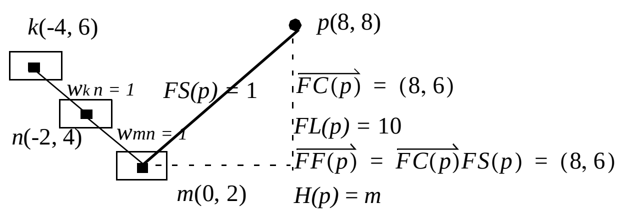

Fig. 1: An example of a fixed point and force equilibrium state

tributes two weighted quadratic terms, $w_{ij}(x_i - x_j)^2$ and $w_{ij}(y_i - y_j)^2$ in x and y direction respectively.

Let $\vec{p}$ denote the coordinate vector $(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$. EQ1 can be rewritten in a matrix form[1]:

$$cf = \frac{1}{2}\vec{p}^T C\vec{p} + \vec{d}^T\vec{p} + const \qquad \text{(EQ2)}$$

To minimize *cf*, we must solve a system of linear equations:

$$C\vec{p} + \vec{d} = 0 \qquad \text{(EQ3)}$$

Traditionally, quadratic formulation is also referred to as a force-directed approach. If we model movable cells and fixed input/output (IO) pads as objects, and nets as springs, the netlist forms a system of objects connected by the springs with different strengths (weights). Minimizing EQ1 is equivalent to putting the system in force equilibrium state. In this state, the force applied on each movable cell by all the connected springs is equal to zero.

It is well known that placement obtained by simply minimizing EQ1 results in excessive overlapping among cells. To eliminate overlapping and to evenly distribute the cells, an additional constant force based on density was introduced in [1]. There, placement was obtained by solving a sequence of modified linear equation systems as follows:

$$C\vec{p} + \vec{d} + \vec{e} = 0 \qquad \text{(EQ4)}$$

In EQ4, $\vec{e}$ is the vector of forces which are assumed to remain constant while solving EQ4. The force equilibrium state is maintained by the constant force $\vec{e}$ and connections among the cells and IO pads.

In [2], a partitioning technique and formation of additional constraints were proposed. In [3], fixed dummy cells in the sparse area were introduced to pull the cells out of the dense areas.

5. Fixed Points Addition and Relaxation

In this section, we begin with definitions relevant to fixed points and force equilibrium state, followed by the discussion of Fixed-points Addition and Relaxation (FAR) in the context of quadratic placement.

5.1 Fixed Points and Force Equilibrium State

Definition 1: A fixed point *p(x,y)* is a dimensionless pseudo cell positioned at *(x,y)* on the chip plane. We use *H(p)* to denote the cell connected to *p*.

Definition 2: A connection of a fixed point *p*, denoted by $\overrightarrow{FC(p)}$, is a directed edge from *H(p)* to *p*. We use *FL(p)* and *FS(p)* to denote the length and strength of $\overrightarrow{FC(p)}$ respectively.

Definition 3: A force introduced by the fixed point *p*, denoted by $\overrightarrow{FF(p)}$, is the attracting force applied on *H(p)*. $\overrightarrow{FF(p)} = \overrightarrow{FC(p)}FS(p)$. If more than one fixed point is connected to a cell *i*, $\overrightarrow{FF(i)}$ is used to denote the total force introduced by all these fixed points.

Figure 1 gives an example. The fixed point *p*(8,8) is connected to its host cell *m*(0, 2). The length of *p* is 10. Suppose the strength of *p* is 1 then the attracting force $\overrightarrow{FF(p)}$ is equal to (8, 6). Since *p* is the only fixed point connected to *m*, $\overrightarrow{FF(m)}$ is also equal to (8,6).

Definition 4: An intrinsic connection $\overrightarrow{IC}$ is a connection between real cells. We use $\overrightarrow{IC(i, j)}$ to denote the intrinsic connection from a cell (or IO pad) *i* to a cell (or IO pad) *j*. Each $\overrightarrow{IC(i, j)}$ has a weight w_{ij} associated with it.

Definition 5: An intrinsic force is the force arising from to an intrinsic connection $\overrightarrow{IC}$. We use $\overrightarrow{IF(i)}$ to denote all the intrinsic forces applied on the cell *i* by *i*'s incident intrinsic connections, and $\overrightarrow{IF(i, j)}$ denotes the intrinsic force caused by the connection $\overrightarrow{IC(i, j)}$. $\overrightarrow{IF(i, j)} = \overrightarrow{IC(i, j)}w_{ij}$. For example, in figure 1, *m* has only one intrinsic connection $\overrightarrow{IC(m, n)}$ with the cell *n*. $\overrightarrow{IF(m)} = \overrightarrow{IF(m, n)} = \overrightarrow{IC(m, n)}w_{mn}$ = (-2, 2).

Definition 6: A constant force $\overrightarrow{CF(i)}$ is the force externally applied on cell *i*. $\overrightarrow{CF(i)}$ does not depend on the location of the cell *i.*

Definition 7: A cell *i* is in a force equilibrium state if $\overrightarrow{FF(i)} + \overrightarrow{IF(i)} + \overrightarrow{CF(i)} = 0$, otherwise, the cell *i* is in disequilibrium. A placement is in force equilibrium state if and only if all the movable cells are in equilibrium state.

In figure 1, the cell *n*(-2,4) is in force equilibrium state because $\overrightarrow{FF(n)} + \overrightarrow{IF(n)} + \overrightarrow{CF(n)} = 0 + \overrightarrow{F(n, k)} + \overrightarrow{F(n, m)} + 0 = 0$.

5.2 Fixed Points Addition

Fixed points are added to the existing placement for the following purposes:

(1) to achieve the force equilibrium state;

(2) to perturb the placement towards a specific direction;

(3) to constrain the cells with different flexibility to remain around their neighborhoods.

We present the following two theorems related to the point (1) above:

Theorem 1: Any given initial placement with fixed IO pads and movable standard cells can be transformed into a force equilibrium state by adding one fixed point to each movable cell.

Theorem 2: Any initial placement can be transformed to a force equilibrium state in an infinite number of ways.

proof: *we omit the proof due to the space limitation.*

The fixed points used to keep a current placement in force equilibrium state are called the *Controlling Fixed Points (cfp)*. To transform a placement into force equilibrium state, theorem 1 tells us that we need only to add one controlling fixed point per cell. Theorem 2 says that we have the flexibility of choosing different combinations of lengths and strengths for each fixed point as long as the attracting force cancels out the intrinsic force induced by the connections.

Once all the cells are in the force equilibrium state, more fixed points can be added to perturb the current placement. These fixed points are called *Perturbing Fixed Points (pfp)*. The addition of new fixed points will result in unbalanced forces acting on cells and thus will cause redistribution of cells towards the new equilibrium state.

The third category of fixed points are the *Constraining Fixed Points (cnfp)*. These points are used to constrain the movement of cells. Suppose that initially a cell *i* is in a force equilibrium state.

We can add a fixed point p with strength *FS(p)* at i's force center. At this moment, the cell i is still in equilibrium state, because the added fixed point does not apply any extra force. Now we perturb the current placement. As a result, the cell i will be relocated from its original position. But the movement of the cell i depends on the magnitude of *FS(p)*. The larger *FS(p)* is, the harder for the cell i to be moved from its original position. In the extreme case, if *FS(p)* is infinitely large, the cell i is fixed and will not be allowed to deviate from its original position. So we can choose fixed points with different strengths and apply them to movable cells to achieve proper mobility requirements.

5.3 Fixed Points Relaxation

Fixed point forces can be relaxed (adjusted) for different purposes:

(1) To improve the timing or congestion objectives, the existing placement can be partially or completely collapsed;

(2) To satisfy the cell mobility requirements.

The fixed points relaxation can be considered as a kind of perturbation. Instead of introducing new perturbing fixed points, we can adjust the existing fixed points. For example, controlling fixed points can be relaxed to expand or collapse the current placement; and constraining fixed points can be adjusted to satisfy different mobility requirements for different cells. Relaxation is equivalent to adjustment in our context. A fixed point p is relaxed when its strength *FS(p)* changes while its location $\overrightarrow{FC(p)}$ remains unchanged. The strength can increase or decrease depending on the specific relaxation requirement.

6. FAR Based Global Placement

In this section, we first compare the fixed points to the constant forces in the context of quadratic placement and prove that the fixed points are a generalization of the constant forces. FAR-based global placement will be discussed after the comparison.

6.1 Fixed Points vs. Constant Forces

The constant force *CF* introduced in [1] can be used to maintain a force equilibrium state and to perturb the current placement. We compare the fixed points to the constant forces in the following categories:

(1) Controllability. Fixed points control the placement better than constant forces do. Using fixed points guarantees that in the force equilibrium state all the movable cells will be located in the bounding box formed by the fixed points and the fixed IO pads. But for a constant force, the final location of a cell can be anywhere, possibly very far away from chip boundary. The phenomenon will be demonstrated in experimental results section.

(2) Determination. Since a fixed point is directly related to a position on the chip, it is much easier to determine a good fixed point than an appropriate constant force. For example, if we want to move a cell i to a new location(x, y), we need only to put a fixed point p at (x, y) and connect it to the cell i. Then we choose an appropriate strength *FS(p)* to reflect how much we want to attract i to (x, y). The larger the *FS(p)* is, the closer i is to (x, y) in the equilibrium state. To determine an appropriate direction and magnitude of a constant force is not trivial.

(3) Flexibility. The following theorem characterizes the flexibility of the fixed points as related to the constant forces.

Theorem 3: Fixed point is a generalization of a constant force.

Proof: A fixed point p is able to mimic the constant force $\overrightarrow{CF(i)}$ applied on a cell i by using an infinitely large length *FL(p)* $= \left|\overrightarrow{FC(i)}\right|$ and infinitely small strength *FS(p)* while making $\overrightarrow{FF(p)} = \overrightarrow{FC(p)}FS(p) = \overrightarrow{CF(i)}$. Since *FL(p)* is infinitely large, the movement of a cell i within the chip's boundary has no effect on $\overrightarrow{FC(p)}$, thus $\overrightarrow{FF(p)} = \overrightarrow{FC(p)}FS(p) = \overrightarrow{CF(i)}$ does not depend on the position of a cell i and remains constant. #

6.2 Applying FAR in Global Placement

As in [1], we will call the process of entering a force equilibrium state from a disequilibrium state, a transformation. In our work, we divide one transformation into two stages. The first stage is to transform the present placement, which either was obtained from a quadratic solver or was an initial placement, to a force equilibrium state by introducing one controlling fixed point per cell. The controlling fixed point p for a cell i can be derived from the intrinsic force $\overrightarrow{IF(i)}$. After we decide the strength *FS(p)*, we can obtain the corresponding $\overrightarrow{FC(i)} = -\overrightarrow{IF(i)}/FS(p)$ and $\vec{p} = \vec{i} + \overrightarrow{FC(i)})$ where $\vec{p}$ and $\vec{i}$ are the coordinate vectors of p and i respectively. In the experiments, *FS(p)* is inversely proportional to chip size. In the words, the bigger chip size is, the more freedom cells are given to move. The reason for introducing one controlling fixed point instead of keeping all the previously added fixed points is to minimally disrupt the original cost function. One fixed point introduces two extra quadratic terms, and a large number of fixed points might dominate the original cost function.

The second stage is to add perturbing fixed points in order to eliminate overlapping and to finish the transformation with a quadratic solver. We apply the same density concept as in [1]. Let w_i and h_i denote the width and height of a cell i, and W and H denote the width and height of the chip. In [1], a density at a location*(x,y)*, *D(x, y)*, has been defined as follows:

$$D(x, y) = N(x, y) - ave, \qquad \text{(EQ5)}$$

$$ave = \frac{\sum w_i \cdot h_i}{W \cdot H}$$

In EQ5, *N(x, y)* is the total number of cells which cover the point *(x,y)*, and *ave* is the quotient of the total cell area and the placement area.

In [1], the following constant density-induced force has been introduced:

$$\overrightarrow{CF(x, y)} = \frac{k}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} D(x', y')\frac{\vec{r} - \vec{r}'}{|\vec{r} - \vec{r}'|^2}dx'dy' \qquad \text{(EQ6)}$$

In EQ6, $\vec{r}$ and $\vec{r}'$ denote position vectors (x, y) and (x', y') respectively. For a cell i located at (x, y), $\overrightarrow{CF(i)} = \overrightarrow{CF(x, y)}$. In [1], the forces acting on each cell are scaled such that the maximum $\overrightarrow{CF(i)}$ is equal to the force of a net whose length is $K \cdot (W + H)$. K is a user-defined constant controlling the overall strength of $\overrightarrow{CF(i)}$. In our current FAR implementation, we use scaled $\overrightarrow{CF(i)}$ as a perturbing force to eliminate overlapping. In other words, we install perturbing fixed points at infinite locations such that each force $\overrightarrow{CF(i)}$ is independent of the position of the cell *i*.

Global placement begins with the matrix C and vector d in EQ1 determined from the netlist structure. In consecutive transformations, C and d are updated according to the introduced controlling fixed points and perturbing forces. We use BiConjugate Gradient Stabilized and preconditioning method to solve the quadratic programs. Transformations are carried out until the stopping criterion is fulfilled. In our implementation, we terminate the placement iterations when the cells are evenly distributed.

7. FAR Based Incremental Placement

In this section, we first discuss the buffer site generation problem and then apply our FAR-based approach to solve it.

7.1 Buffer Sites Generation(BSG) Problem

Suppose that after obtaining the initial placement results we discover that buffers have to be inserted on some nets. If there is no empty space to place the buffers at appropriate locations, we have to modify the existing placement such that the buffers can be inserted in positions that are valid for the purpose they serve. The BSG problem is to transform incrementally the existing placement to introduce enough buffer sites at appropriate locations as required by the buffer insertion algorithm. The minimal placement transformation does not necessarily constrain all the cell movements in the same degree. Instead, different cells might have different mobility requirements. For example, timing-critical cells could be perturbed as little as possible. If the topology of a net is to remain intact, we can limit the mobility of the cells incident to it. In our BSG implementation, we use the cell mobility distance to denote cell *i's* maximum allowable deviation from its original position.

The incremental placer imposes a global bin structure on a chip. For each bin *b*, we define:

(1) *A(b)* is the chip area covered by *b*.

(2) *CA(b)* is the total cell area in *b*.

(3) *CCA(b)* is the total chip area covered by all the cells in *b*. The difference between *CA(b)* and *CCA(b)* is that *CA(b)* is the summation of the cell areas while *CCA(b)* denotes the total chip area. It's possible for *CA(b)* to exceed *A(b)*. But the maximum *CCA(b)* is *A(b).*

(4) *COR(b)* is the cell occupation ratio defined as *CCA(b)/CA(b). COR(b)* reflects how evenly the cells are distributed in *b*.

(5) *BS(b)* is the buffer site supply in *b*. It denotes the buffer-usable blank space in terms of number of buffers in *b*.

(6) *BD(b)* is the buffer site demand in *b*. It is the result of buffer insertion.

For each cell *i*, we have the following definitions:

(1) *M(i)* denotes the cell's *i* mobility distance, which expresses the maximum allowable deviation from its original position during incremental placement.

(2) $\overrightarrow{m(i)}$ is the actual deviation after incremental placement. Let *(x,y)* and (x_0, y_0) denote the cell *i*'s present and initial locations respectively. $\overrightarrow{m(i)} = (x - x_0, y - y_0)$. $m_x(i)$ and $m_y(i)$ denote the projections of $\overrightarrow{m(i)}$ on *x*-axis and *y*-axis, respectively.

Based on the definitions above, Buffer Site Generation (BSG) problem is formulated as follows:

Given a placement *P*, incrementally transform *P* to *P'* such that $SD = \sum(|m_x(i)| + |m_y(i)|)$ is minimized while for each bin *b* in *P'*, the following constraints are satisfied:

(1) *BS(b) >= BD(b).*

(2) *CA(b) < A(b).*

(3) *COR(b) > cor_constant.*

and for each *i* in *P'*:

(4) $|m_x(i)| \le M(i)$ and $|m_y(i)| \le M(i)$.

7.2 Applying FAR to BSG

Our FAR-based approach provides an efficient way to solve the BSG problem. The added controlling and perturbing fixed points can be used to minimally perturb the existing placement while keeping the global placement structure. The constraining fixed point relaxation can be exploited to reflect a cell's mobility requirement.

First, for each cell *i* we introduce one constraining fixed point *cnfp(i)* at *i*'s initial position and with an initial strength *FS0(cnfp(i)). FS0(cnfp(i))* is determined by EQ7:

$$FS_0(cnfp(i)) = \alpha\left(\sum_{j \in N(i)} M(j)w_{ij}\right) / (M(i)) \qquad \text{(EQ7)}$$

where *N(i)* denotes the set of *i*'s adjacent cells and w_{ij} is the weight on an intrinsic connection $\overrightarrow{IC(i,j)}$. *S(cnfp(i))* captures the maximum force possible generated by the movement of *i's* neighboring cells. *FS0(cnfp(i))* depends on how strongly i's neighbor cells might be perturbed, which is measured by $\sum_{j \in N(i)} M(j)w_{ij}$, as well as *i*'s mobility distance *M(i)*. The stronger the expected perturbation is, the larger *FS0(cnfp(i))* is. On the other hand, the larger the mobility of a cell *i* is, the smaller its *FS0(cnfp(i))*. α is a user-defined parameter to control the overall constraining strength. The location of *cnfp(i)* is fixed throughout the incremental placement while *FS(cnfp(i))* is dynamically adjusted to penalize those cells which move beyond their mobility distance. In the current implementation, we scale *FS(cnfp(i))* by a factor *_cnfp_scale* each time when *i* exceeds its mobility distance.

As in the global placement, we have one controlling fixed point for each cell to guard the global placement structure, and we use the density induced forces $\overrightarrow{CF(i)}$ as perturbing fixed points. To derive $\overrightarrow{CF(i)}$, we modify EQ5 and take into account the buffer demand in each global bin *b* as expressed in EQ8:

$$D(x, y)' = D(x, y) + \frac{BD(b)}{A(b)}, for((x, y) \ni b) \qquad \text{(EQ8)}$$

After scaling $\overrightarrow{CF(i)}$ as in the global placement, we further refine $\overrightarrow{CF(i)}$ to make it consistent with *i*'s mobility requirement. If a cell *i* is already beyond its mobility distance, $\overrightarrow{CF(i)}$ is set to zero. For cells still within their mobility distances, we use the constraint of EQ9 to adjust their $\overrightarrow{CF(i)}$ while keeping the directions unchanged.

$$CF_{x,y}(i) \le \begin{cases} (M(i) - |m_{x,y}(i)|)FS(cnfp(i)) & \text{if } CF_{x,y}(i)m_{x,y}(i) > 0 \\ M(i)FS(cnfp(i)) & \text{otherwise} \end{cases} \qquad \text{(EQ9)}$$

In EQ9, $M(i) - |m_{x,y}(i)|$ measures how far a cell *i* is from a mobility distance violation. Intuitively, the constraint in EQ9 says that $\overrightarrow{CF(i)}$ cannot be too large to be cancelled by *cnfp(i)* within *i*'s mobility distance.

BSG, as the global placer, proceeds through a sequence of transformations. Each transformation is divided into the following five steps: (1) transform the current placement into an equilibrium state; (2) adjust strengths of the constraining fixed points; (3) insert the density-induced perturbing fixed points; (4) determine the transformation by solving the linear equations; (5) evaluate the result.

8. Experimental Results

We build FAR-based placer using data structure of Capo[10]. The linear equations system solver is Laspack from [12]. MCNC benchmarks from [11] are used for experiments. Experiments are carried on 850Mhz Pentium III processor with 256M memory. CPU times are reported in seconds.

8.1 FAR-Based Global Placement

In the following experiments, we denote [1] based approach as CF and our approach as FP. Cliques model multiple-pin nets. Each edge in a clique has the weight *1/(k-1)*, where k is the degree of a net. In the CF implementation all the density-induced forces $\overrightarrow{CF(i)}$ are scaled such that the maximum $\overrightarrow{CF(i)}$ is equal to the force of a $K \cdot (W+H)$ long net. *W* and *H* are the chip's width and height, respectively. In our current implementation, the strength of the controlling fixed points ranges from 0.01 to 1, depending on the chip size. The larger the chip is, the smaller the strength is. The scaled $\overrightarrow{CF(i)}$ is used for perturbing fixed points. The difference between the CF and FP is that in FP we use fixed points instead of constant forces to maintain the force equilibrium state.

circuit	CF(s)	FP(s)	imp%
biomed	4.5	2.17	+51.8
industry2	10.1	4.68	+53.6
industry3	10.7	4.87	+54.5
avqsmall	22.5	7.7	+65.8
avqlarge	30	8.9	+70
Average			**+59.1**

TABLE 1. CPU time for one transformation

First, we show that FP has inherently better placement controllability than CF because it captures the chip geometry. Figure 2 shows the results of one transformation of the same netlist performed by CF and FP. Figure 2a depicts the initial force equilibrium state for the benchmark *biomed*. We set K = 0.1 to scale the $\overrightarrow{CF(i)}$ for CF and K = 0.2 for FP. This means that our density-induced forces are twice as strong as those used in CF. Figures 2b and 2c depict the placements after one transformation performed by CF and by FP, respectively. The smaller rectangle in the figure 2b shows the real boundary of the chip. The quadratic solver places a large fraction of cells outside the chip boundary. This is so, because the constant forces are not related to the chip's geometry and do not change for changing cell locations. Figure 2c shows the placement performed by FP. Even though $\overrightarrow{CF(i)}$ in FP is twice as strong as in CF, FP is still able to control the global placement structure very well.

Secondly, we show the results on the MCNC benchmarks. Global placement terminates when the current solution occupies more than 85% of the chip area or a predefined MAX_iteration is exceeded. In our experiments, MAX_iteration is set to be 30. We developed a legalization procedure to fit cells into rows and a simple swapping-based detailed placer to evaluate the relative quality of placements from CF and FP. We compare the total wire length, which is measured by summing the half perimeters of bounding boxes for all the nets, after legalization and after detailed placement. The reported CPU times are only for the global placement. We set K = 0.05 for CF and FP.

In table 2, we observe 54.4% CPU speedup of FP over CF with comparable total wire lengths. CPU efficiency comes from the performance of the linear equation system solver and from the fact that fixed points are reflecting the chip's geometry. Experiments indicate that FP is generally more solver-efficient than CF. Table 1 gives the CPU times of CF and FP for one transformation. FP has controlling fixed points with strength 1. The average speedup for one transformation is 59.1% and the maximum speedup is up to 70%.

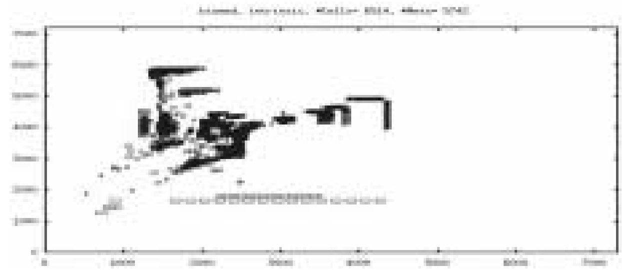

Figure 2a: initial equilibrium state for*biomed*

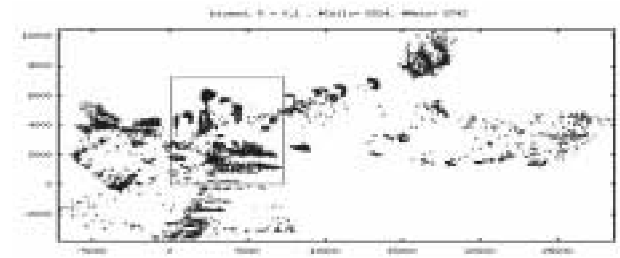

Figure 2b: after 1 transformation using constant force[1], K = 0.1

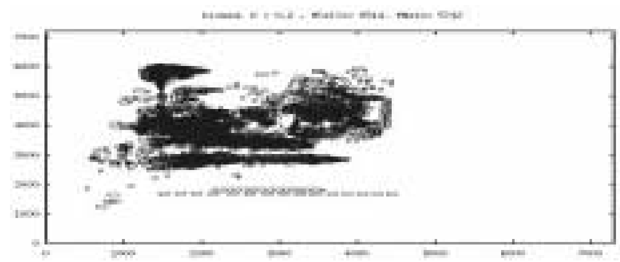

Figure 2c: after 1 transformation using fixed points, K = 0.2

8.2 FAR Based Buffer Site Generation

For BSG, we start from an initial placement with a total area of buffer site demand equal to 10% of the chip area. The size of a global bin is chosen to accommodate roughly 100 standard cells of average width. Then for each global bin we collect the buffer site supply information, and we introduce randomly a mismatch between the supply and demand. We generate randomly the cell mobility distances *M(i)* and express them as fractions of a user-defined maximum mobility distance MAX_MOBILITY. In the experiments, we set MAX_MOBILITY to be twice the average cell width. For experiments we selected 4 large circuits in MCNC benchmark set. Table 3 shows BSG statistics of all the test cases. The column labeled *GBS* shows the global bin structure imposed on the chip. *#BS*, *#BD* and *#BM* are the numbers of the total buffer supply, buffer demand, and buffer mismatch, respectively. *#BM* is obtained by summing the buffer mismatches *(BD(b) - BS(b))* for those bins where *(BD(b) - BS(b)) > 0*. *#BBM* is the number of bins with buffer mismatch.

In experiments, the strength of controlling fixed points and α in EQ7 are set to 1; the mobility violation penalizing factor *_cnfp_scale* is set to 2; and K = 0.05. The transformation terminates whenever (1) the predefined parameter *max_iteration* has been reached; or (2) all constraints in BSG formulation have been satisfied. The distribution parameter *cor_constant* is set to 0.95 in

circuit	#cells	#nets	#row	WL after legalization [m]			WL after simple opt [m]			CPU[s]		
				CF	FP	%imp	CF	FP	%imp	CF	FP	%imp
primary2	2907	3029	28	4.17	4.18	-0.2	3.44	3.45	-0.3	43.6	24.6	+43.6
biomed	6417	5742	46	5.98	5.75	+3.8	4.48	4.23	+5.6	142	31	+78.2
industry2	12142	13419	72	23.9	19.1	+20.1	16.7	14.4	+13.8	486	213	+56.2
industry3	15059	21940	54	50.8	51.1	-0.6	40.7	42.7	-4.9	250	131	+47.6
avq.small	21854	22124	80	10.9	11.3	-3.7	7.47	7.40	+0.93	771	527	+31.6
avq.large	25114	25384	86	12.5	12.2	+2.4	8.12	8.25	-1.6	903	277	+69.3
Total												**+54.4**

TABLE 2. CF vs. FP in global placement for total wire length minimization

the experiments, and the maximum number of iterations is set to 20.

	GBS	#BS	#BD	#BM	#BBM
biomed	7x8	639	535	348	12
industry3	12x12	1426	1126	734	33
avqsmall	14x14	2614	2166	1354	52
avqlarge	15x15	2934	2555	1597	64

TABLE 3. Test cases for BSG

Since CF does not have a direct capability of constraining a cell's movement as shown in figure 2, the experiments below are performed only on FP. Table 4 shows the results for BSG using FAR. *#bm* and *#mv* denote, respectively, the percentage of buffer mismatches and mobility violations left by the incremental placer. In 20 iterations, on the average, 89.2% buffer mismatches can be eliminated with only 0.57% mobility violations. We believe that efficiency is possible because fixed points are intrinsically controllable. By choosing appropriate fixed points to guard and perturb the given placement, the cells will not suffer dramatic disturbances and the overall placement structure can be maintained. *SD* is the value of the cost function, $SD = \sum(|m_x(i)| + |m_y(i)|)$. In the last column we list the CPU times in seconds.

	#bm	#mv	SD(m)	SD'(m)	SD/SD'	CPU
biomed	2.2%	0.34%	0.16	3.8	4.2%	38.59
industry3	15.2%	0.44%	0.69	30	2.3%	137
avqsmall	10.9%	0.65%	0.38	8.1	4.7%	155
avqlarge	15.1%	0.85%	0.46	41.8	1.1%	205
average	**10.8%**	**0.57%**			**3.1%**	

Table 4: Experimental Results for FAR based BSG

We have also compared the FAR-based incremental placement to a flow in which buffers are pre-placed according to the results from the first run *R1* of the placer, and the placement is restarted taking these buffer sites into account. We use the difference between *R1* and the result of the second run *R2* to compute the total cell deviation *SD'* in the fifth column of table 4. From the ratio *SD/SD'* in the sixth column it can be observed that the placement does not converge. A small disturbance in the early stage, in this case caused by pre-placed buffer sites, might lead to a very different result. On the other hand, FAR-based incremental placement is able to minimally perturb the initial placement, and the cell position deviation is on the average only 3.1% of the pre-place-and-restart flow.

9. Conclusions

We described the Fixed-points Addition and Relaxation (FAR) based placement technique. FAR can be applied to solve global and incremental placement problems. Experimental results indicate that FAR is stable and efficient in minimizing the total wire length and is able to accommodate buffer sites at desired locations.

10. Acknowledgment

This work was supported by the MARCO/DARPA Gigascale Silicon Research Center (GSRC).

11. References

[1] H.Eisenmann and F.M.Johannes, “Generic Global Placement and Floorplanning”, in *Proc. of the 35th Design Automation Conference,* pp. 269-274, 1998.

[2] J.M.Kleinhans, G.Sigl, F.M.Johannes and K.J.Antreich, “GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization”, *IEEE Trans. CAD*, vol.10, no.3, Mar1991, pp.356-365.

[3] H.Etawil, S.Areibi, A.Vannelli, “Attractor-repealer approach for global placement”, in *Proc. of International Conference on Computer-Aided Design*, pp.20-24, 1999.

[4] J.Cong, T.Kong and D.Z.Pan, “Buffer Block Planning for Interconnect-Driven Floorplanning”, *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 1999, pp.358-363

[5] F.F.Dragan, A.B.Kahng, I.Mandoiu, S.Umddu, “Provably Good Global Buffering Using an Available Buffer Block Plan”, *IEEE/ACM ICCAD*, 2000, pp. 104-109

[6] C.J.Alpert, Jiang Hu, S.S.Sapatnekar, P.G.Villarrubia, “A Practical Methodology for Early Buffer and Wire Resource Allocation”, *Design Automation Conference*, 2001, pp. 189-194.

[7] X.Tang and D.F.Wong, “Planning Buffer Locations by Network Flows”, *Intl. Symp. Physical Design*, 2000, pp. 180-185.

[8] W.-J.Sun and C. Sechen, “Efficient and effective placement for very large circuits”, *IEEE Trans. CAD*, vol 14, no. 3, pp.349-359, 1995

[9] M.Wang, X.Yang, M.Sarrafzadeh, “Dragon2000: standard-cell placement tool for large industry circuits”, *IEEE/ACM ICCAD*, 2000, pp. 260-263

[10] A.E.Caldwell, A.B.Kahng, I.L.Markov, “Can recursive bisection alone produce routable placements”, *Design Automation Conference*, 2000, pp. 477-482

[11] “http://vlsicad.cs.binghamton.edu/benchmarks.html”.

[12] “http://www.tu-dresden.de/mwism/skalicky/laspack/laspack.html”