Quadratic VLSI Placement

Manolis Pantelias

1. INTRODUCTION

Placement is a classical problem in VLSI physical design. It can be driven by different objectives, such as timing, routability, thermal distribution, or a combination of these. However the classical objective function is the total wire length, as it has been found that minimizing it, helps placement to meet the target timing or routability requirements. We can classify the placement algorithms into four categories: simulated-annealing, quadratic or force-directed, min-cut and nonlinear programming.

In this work we focus on three papers that fall in the second category. Their algorithm names are PROUD, Kraftwerk and FastPlace. Unfortunately, the optimum solution of the unconstrained quadratic programming has a large proportion of overlapping cells. In order to overcome this issue PROUD adopts a partitioner in the quadratic placement flow. In Kraftwerk the authors formulated the density-based additional forces that drive the cells away from the dense regions into the sparse ones. Finally FastPlace enhances this force-directed approach with a new cell-spreading strategy and a local refinement procedure.

2. PROUD: A SEA-OF-GATES PLACEMENT ALGORITHM

This paper is written by R. S. Tsay, E. Kuh and C. P. Hsu and it is published in 1988. The authors assume that all modules are point modules and all nets are two-pin nets. Thus, they preprocess multipin nets and replace them with two-pin nets. They also use the term of connectivity between two modules. For example the connectivity c_{ij} between module i and module j could be the number of nets connecting the two modules. They introduce the symmetric connectivity matrix $\mathbf{C} = [c_{ij}]$ with $c_{ii} = 0$. The objective function they use is the following:

$$L(\mathbf{x},\mathbf{y}) = 1/2 \sum_{i,j=1}^{n} c_{ij} [(x_i - x_j)^2 + (y_i - y_j)^2] = \mathbf{x}^T\mathbf{B}\mathbf{x} + \mathbf{y}^T\mathbf{B}\mathbf{y}$$

where (x_i, y_i) represents the coordinates of module i and **B** is a modified connectivity matrix defined as $\mathbf{B} = \mathbf{D} - \mathbf{C}$. **D** is a diagonal matrix with:

$$d_{ii} = \sum_{j=1}^{n} c_{ij}$$

x and **y** are n-vectors that specify the coordinates of the n modules. The problem is to minimize L subject to the slot constraints: all point modules are placed on an evenly spaced, two-dimensional grid.

In order to solve this problem they consider the electric network analogy. The objective function $\mathbf{x}^T\mathbf{B}\mathbf{x}$ (and accordingly $\mathbf{y}^T\mathbf{B}\mathbf{y}$) can be interpreted as the power dissipation of an n-node linear resistive network, where **x** represents the voltages at

the nodes. **B** contains the conductance of the nodes; $-b_{ij}$ is equal to the conductance between node i and node j. The problem is then equivalent to that of choosing the node voltages for which the power dissipation is a minimum. They also model the I/O pads as fixed voltage sources applied to the network.

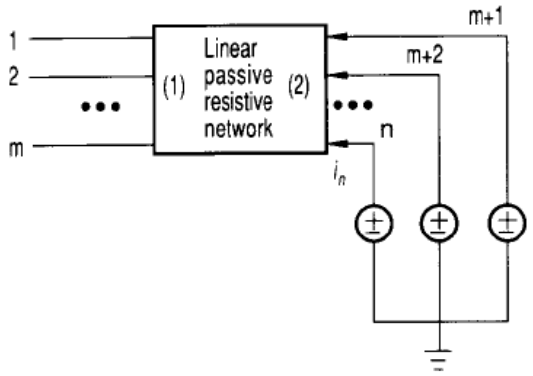

Figure 1: An n-terminal linear, passive-resistive network whose first m nodes are floating. The remaining n-m nodes are connected to voltage sources.

In Figure 1 nodes 1 to m represent movable modules and nodes m+1 to n represent fixed modules with voltages specified by I/O pads. The n-port network equation can be written as:

$$\mathbf{B_{11}x_1} + \mathbf{B_{12}x_2} = \mathbf{0} \qquad (1)$$
$$\mathbf{B_{21}x_1} + \mathbf{B_{22}x_2} = \mathbf{i_2} \qquad (2)$$

$\mathbf{B_{11}}$, $\mathbf{B_{12}}^T = \mathbf{B_{21}}$ and $\mathbf{B_{22}}$ are the familiar submatrices of **B**. The voltage vector, $\mathbf{x_1}$, is of dimension m and is to be determined, the vector $\mathbf{x_2}$ is the voltage source, and the vector $\mathbf{i_2}$ is the current flowing into n-m terminals from the sources. Equation 1 is the key equation and can be written as $\mathbf{Ax_1} = \mathbf{b_x}$ where:

$$\mathbf{A} = \mathbf{B_{11}} \text{ and } \mathbf{b_x} = -\mathbf{B_{12}x_2} \qquad (3)$$

For the other dimension we have $\mathbf{Ay_1} = \mathbf{b_y}$ with $\mathbf{b_y} = -\mathbf{B_{12}y_2}$. These sparse linear equations can be solved with the SOR method (successive over-relaxation) which is a generalized Gauss-Seidel method. The running time in each iteration is linear.

Until now it has been assumed that all modules are point modules so replacing them with real modules may cause overlapping. In order to overcome this, the authors introduce an iterative partitioning scheme. The location of the partition line is determined as follows: We sort the modules from left to right and add up the module areas until we get roughly half the total module area. If the partition line happens to coincide with the center line, the partition has been done. If not, we move all the modules that lie between the center line and the partition line to the other side of the center line. At the end of this process, the center line divides the modules into two sets. Each set occupies half the area of the chip required by all the modules.

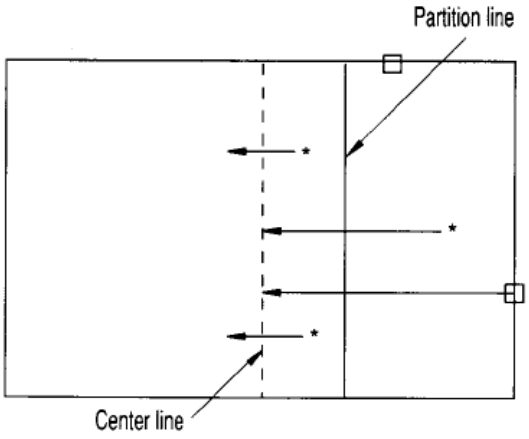

Figure 2: Projecting modules from the right half to the center line to determine the left-half placement.

To obtain a new global placement after partitioning the authors use a simple heuristic. Assume that the partition line is to the right of the center, as shown in Figure 2. All modules to the right of the partition line are projected horizontally on the center line to form part of the fixed modules. Only modules that lie to the left of the partition line are in the set of movable modules. We then solve the global placement problem in the left half by the SOR method and determine a temporary top-bottom partitioning in that half. We next project the movable modules on the left half horizontally to the center line as part of fixed modules to calculate the global placement on the right half. We also determine a temporary top-bottom partitioning. We repeat this process two or more times to fix the partition on each half. We repeat the same procedure at the next hierarchical level and continue until each block contains one module to be placed at the only legal location in the block. Legal location is the location that satisfies the slot constraints.

At each hierarchy we perform the BGS (block Gauss-Seidel) iteration. Suppose that after a vertical cut, we partition the movable modules into the left half and right half of the center line according to the global placement in x direction. In the next hierarchy, we make a horizontal cut in each half in y direction. We use the BGS iteration to solve a global solution of $\mathbf{y_1}$. Let the coordinates of the movable modules be depicted with subindices according to the two halves. Thus we partition $\mathbf{y_1}$ into

$$\begin{bmatrix} \mathbf{y}_{1a} \\ \mathbf{y}_{1b} \end{bmatrix}.$$

An extension of equation (1) becomes:

$$\mathbf{A_{11}y_{1a}} + \mathbf{A_{12}y_{1b}} = \mathbf{b_{y1}} \qquad (4)$$
$$\mathbf{A_{21}y_{1a}} + \mathbf{A_{22}y_{1b}} = \mathbf{b_{y2}} \qquad (5)$$

where:

$$\begin{bmatrix} \mathbf{b}_{y1} \\ \mathbf{b}_{y2} \end{bmatrix} = \mathbf{b}_y$$

is the contribution from the fixed modules or the given I/O pads as in equation (3). To get the global solution of $\mathbf{y_1}$ after the modules are partitioned, we have:

$\mathbf{y_{1a}} = \mathbf{A}^{-1}_{11} [\mathbf{b_{y1}} - \mathbf{A_{12}}\mathbf{y_{1b}}^*]$
$\mathbf{y_{1b}} = \mathbf{A}^{-1}_{22} [\mathbf{b_{y2}} - \mathbf{A_{21}}\mathbf{y_{1a}}^*]$

where $\mathbf{y_{1a}}^*$ and $\mathbf{y_{1b}}^*$ are the perturbed solution from $\mathbf{y_{1a}}$ and $\mathbf{y_{1b}}$ because of the partitioning process or linear placement. Two to five BGS iterations give very good results at each level of hierarchy.

The quality of the placement can be further improved by considering actual pin position and local perturbation (such as module rotation, I/O pad position adjustment, module swap or insertion). This is referred by the authors as "detailed placement improvement".

We can now define the complete algorithm, which is the following:

PROUD_PLACEMENT:

Input: netlist, cell library, I/O pad location, net weighting, number of hierarchies, number of BGS iterations.
Output: placement result.

Algorithm:

```
Assign all modules to one internal block, and assign all I/O pads to one I/O block.
Construct the modified connectivity matrix in the sparse form.
For each hierarchy
{
  Repeat number of BGS iterations (repeat only once for the first hierarchy)
  {
    For each internal block
    {
      Reconstruct the modified connectivity matrix for this block.
      Calculate the right hand side of the linear equation according to the boun-
        dary condition determined by the projections of modules or I/O pads in
        other blocks.
      For global Placement, use the SOR method to solve for relative module
        positions.
      Temporarily cut the block into two subblocks according to the cut method for
        this hierarchy. Assign the modules into separate sub-blocks by sorting.
    }
  }
  Fix the cut of each internal block and permanently associate the modules to the
    corresponding subblocks, which form a new set of internal blocks for next
    hierarchy.
}
Do detailed placement improvement.
```

We assume that the number of pins is proportional to the number of modules, so the memory complexity in the sparse matrix formulation (matrix **B**) is O(n) where n is the number of modules. As for the timing complexity, the number of iterations of the SOR method is smaller than a constant which makes global placement and the iterative linear-system solver linear. The complexity for the sorting algorithm is O(nlogn) at each partitioning step and the hierarchical cut creates a binary tree that can have at most logn levels. Thus, the runtime complexity of the whole method is $O(n\log^2 n)$. However if we use detailed placement then it will dominate runtime.

3. GENERIC GLOBAL PLACEMENT AND FLOORPLANNING (KRAFTWERK)

This paper is written by Hans Eisenmann and Frank M. Jones and it was published in 1998. As the previous work it separates the cells of a circuit into two categories: the fixed (e.g. I/O pads) and the movable ones. Let n be the number of movable cells, with x_i the x-coordinate of the center of cell i and y_i the y-coordinate of the center of cell i. The problem is to find the 2n-dimensional vector:

$$\vec{p} = (x_1, \ldots, x_i, \ldots, x_n, y_1, \ldots, y_i, \ldots, y_n)^T.$$

A net connecting k cells yields a clique in the graph. The clique consists of k·(k-1)/2 edges with weight 1/k connecting each vertex with all other vertices.

The cost of an edge is the same as previous: the squared Euclidean distance between the adjacent vertices multiplied with the weight of the edge. The squared Euclidean distance between two movable cells i and j is $(x_i - x_j)^2 + (y_i - y_j)^2$. If cell i is connected to a fixed cell with coordinates (x_f, y_f), the distance is $(x_i - x_f)^2 + (y_i - y_f)^2$. The objective function now (which is the same as the previous paper) sums up the cost of all the edges and is

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

C is a 2n x 2n symmetric matrix and **d** is a 2n-dimensional vector. For example the x-part of the connection between two movable cells i and j is $(x_i - x_j)^2 = x_i^2 - 2x_ix_j + x_j^2$. The first term contributes to the diagonal of **C** at row i, the second term causes negative entries at row i, column j and at row j, column i. The third term is a contribution to the diagonal of **C** at row j. In case of a fixed connection, $(x_i - x_f)^2$ evaluates to $x_i^2 - 2x_ix_f + x_f^2$. The first term is a contribution to the diagonal of **C**, the second term gives a negative entry at **d** in row i and the third term affects the constant part of (1). In order to minimize (1) we must solve the following linear equation system:

$$C\vec{p} + \vec{d} = 0 \qquad (2)$$

This formulation is equivalent to modeling nets as springs and calculating the state of equilibrium. In detail, row i (row i + n) of equation system (2) states that the force working on cell i is zero in x direction (y direction).

In order to overcome the problem of cell overlapping, the authors introduce additional forces and extend equation (2) with the force vector **e**:

$$C\vec{p} + \vec{d} + \vec{e} = 0 \qquad (3)$$

Equation (3) transforms the problem of finding a placement into the problem of finding additional forces **e**.

The authors derive four requirements for the additional forces.

1. For a given placement, the additional force working on a cell depends only on the coordinates of the cell.

2. Regions with higher density are the sources of the forces. Regions with lower density are the sinks.
3. The forces do not form circles.
4. In infinity, the force should be zero.

The first requirement allows us to write the additional force $\mathbf{f_i}$ at cell i as a function of x and y:

$$\vec{f}_i = \vec{f}(x,y)|_{x=x_i,y=y_i}$$

For the second requirement we must define somehow the density at a given point. We first define the rectangle function R(z):

$$R(z) = \begin{cases} 1 & \text{if } -1/2 < z < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

We also define the function $a_i(x,y)$ which describes the area of cell i:

$$a_i(x,y) = R\left(\frac{x-x_i}{w_i}\right) \cdot R\left(\frac{y-y_i}{h_i}\right)$$

w_i is the width of cell i and h_i is the height of cell i. The value of a_i is one for a point which is covered by cell i and zero otherwise. In a similar way for the placement area we define:

$$A(x,y) = R\left(\frac{x}{W}\right) \cdot R\left(\frac{y}{H}\right)$$

W is the width of the placement area and H is its height. The value of A is one for a point within the placement area and zero otherwise. Now we can define the density of a given point as:

$$D(x,y) = \sum_i a_i(x,y) - s \cdot A(x,y) \quad \text{with} \quad s = \frac{\sum_i w_i \cdot h_i}{W \cdot H} \qquad (4)$$

s is the quotient of total cell area and placement area. In other words: The density at a certain point within the placement area is the number of cells which cover the point, minus s. The integral of D(x,y) over the hole area is zero. Also note that $D(x,y) > 0$ at locations with higher density than desired and $D(x,y) < 0$ at locations with lower density than desired. Using a proportional constant k requirement 2 gives:

$$\vec{\nabla}\vec{f}(x,y) = k \cdot D(x,y) \qquad (5)$$

Requirement 3 means that f is conservative which means that there exists a scalar function $\Phi(x,y)$ with

$$\vec{\nabla}\Phi(x,y) = \vec{f}(x,y) \qquad (6)$$

Combining (5) and (6) results in Poisson's equation

$$\Delta\Phi(x,y) = k \cdot D(x,y) \qquad (7)$$

with boundary conditions from requirement 4

$$\lim_{|\vec{r}|\to\infty} |\vec{\nabla}\Phi(x,y)| = 0 \text{ with } \vec{r} = (x,y)^T \qquad (8)$$

This standard problem has a unique solution

$$\vec{f}(x,y) = \frac{k}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} D(x',y')\frac{\vec{r}-\vec{r'}}{|\vec{r}-\vec{r'}|^2}dx'\,dy' \qquad (9)$$

with $\mathbf{r'} = (\mathbf{x'},\mathbf{y'})^T$.

If we model the cells as points and subdivide the placement area into places with unit area then equation (9) reveals the following features of **f**:

1. The force working on a cell is the superposition of the forces originating from other cells and places.
2. The force exerted on a cell by another cell is repelling and proportional to the inverse of their distance.
3. The force exerted on a cell by a place is attracting and proportional to the inverse of their distance.
4. The direction of the force is parallel to the straight line between the two cells or the cell and the place.

Before the definition of the basic algorithm we must define what a placement transformation is. In a placement transformation (which is performed in every iteration) we determine the additional forces of equation (9) and choose constant k so that the maximum strength of all forces $\mathbf{f_i}$ is equivalent to the force of a net with length K·(W+H). K is constant and user specified. Then we solve equation (3) by using a conjugate gradient approach with preconditioning and assuming that **e** remains constant.

The basic algorithm now is the following:

1. Initially all cells are placed at the center of the placement area and forces **e** are set to zero.
2. We iteratively apply placement transformations. The choice of K depends on the desired speed. For example K = 0.2 is used for standard behavior while K = 1.0 is used for fast.
3. The iterations are stopped when there exists no empty square which is larger than four times the average area of a cell.

In order to optimize the timing behavior the authors suggest the following weighting approach. Before each placement transformation, we carry out a longest path search. Then the criticality of nets is updated which (for a net j at iteration m) is defined as:

$$c_j^{(m)} = \begin{cases} (c_j^{(m-1)} + 1)/2 & \text{if net } j \text{ is among the 3 percent most critical nets} \\ c_j^{(m-1)}/2 & \text{otherwise} \end{cases}$$

For example, if a net was never critical its criticality is zero whereas an always critical net has a criticality of one. The weight of a net j at iteration m is given by:

$w_j^{(m)} = w_j^{(m-1)}(1+c_j^{(m)})$, with $w_j^{(0)} = 1$

In other words, a net which has never been critical keeps its weight and the weight of a net which has always been critical is multiplied by a factor of 2: $w_j^{(m)} = 2w_j^{(m-1)}$.
In a similar way with net weight adaptation, before each placement transformation, we can meet any given timing requirements we have, without having to rerun the algorithm from the beginning with changed input data in case of violation.

4. FASTPLACE: EFFICIENT ANALYTICAL PLACEMENT USING CELL SHIFTING, ITERATIVE LOCAL REFINEMENT AND A HYBRID NET MODEL

This paper is written by Natarajan Viswanathan and Chris Chong-Nuen Chu and it was published in April of 2004. As the previous work the authors model the nets between cells as springs. However they don't use additional forces to avoid cell overlapping but a different approach called cell shifting.

FastPlace consists of three stages. The aim of the first stage is to minimize the wirelength and spread the cells over the placement region to obtain a coarse global placement. The second stage is to reduce the wirelength further based on the half-perimeter measure. The third stage consists of legalizing the current placement by assigning cells to pre-defined rows in the placement region and removing any overlap among them. It also consists of further reducing the wirelength by a greedy heuristic. The algorithm is the following:

Stage 1: Coarse Global Placement (CGP)

1. **Repeat**
2. Perform Global Optimization.
3. Perform Cell Shifting and Add Spreading Forces.
4. **Until** the placement is roughly even.

Stage 2: Wirelength Improved Global Placement (WIGP)

1. **Repeat**
2. Perform Global Optimization.
3. Perform Iterative Local Refinement.
4. Perform Cell Shifting and Add Spreading Forces.
5. **Until** the placement is very even.

Stage 3: Detailed Placement (DP)

1. **Repeat**
2. Further reduce the wirelength using a greedy heuristic.
3. Legalize the current placement solution.
4. **Until** no significant improvement in wirelength.

Global optimization consists of solving the equations known from the previous paper:

$\mathbf{Qx} + \mathbf{d_x} = \mathbf{0}$. (1)
$\mathbf{Qy} + \mathbf{d_y} = \mathbf{0}$. (2)

Since matrix Q is sparse, symmetric and positive definite, the authors solve equations (1) and (2) by the pre-conditioned Conjugate Gradient method with the Incomplete Cholesky Factorization of matrix Q as the preconditioner. The runtime of this method is directly proportional to the number of nonzero entries in matrix Q. This in turn is equal to the number of two-pin nets in the circuit. Hence, it becomes imperative to choose a good net model so as to have minimal non-zero entries. In order to accomplish it the authors suggest a hybrid net model which is a combination of the clique and star models. They also prove the equivalence of the clique and star models, and hence the consistency of the Hybrid Net Model.

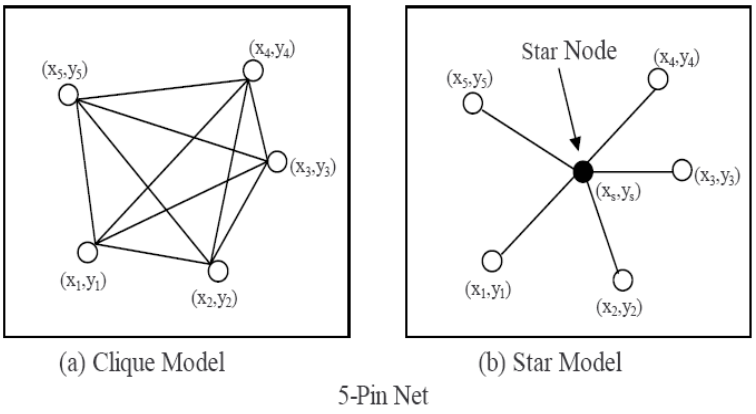

Figure 3: Net Models.

In the clique model, a k-pin net is replaced by k(k −1)/2 two-pin nets forming a clique. The clique model for a 5-pin net is illustrated in Figure 3(a). In the star model, each net has a star node to which all pins of the net are connected. Hence, a k-pin net will yield k two-pin nets. The star model for a 5-pin net is illustrated in Figure 3(b). The authors prove that for a k-pin net of weight W, if we set the weight of the two-pin nets introduced, to γW in the clique model and kγW in the star model for any γ, the clique model is equivalent to the star model. Therefore, the two models can be used interchangeably. To prove it they show that at equilibrium positions the total forces on each cell are the same for both clique and star net models. They propose a Hybrid Net Model which uses a clique model for two-pin and three-pin nets, and a star model for nets with four or more pins. They set γ to 1/(k − 1) in FastPlace as it works better experimentally.

In order to overcome the cell overlapping the authors use Cell Shifting. This works as follows: Initially, the placement region is divided into equal sized bins. Based on the placement obtained from the Global Optimization step, the utilization of each bin (Ui) is then computed. Ui is defined as the total area of all cells inside bin i. The standard cells are then shifted around the placement region based upon the bin in which they lie and its current utilization. To illustrate the shifting in the x-direction, consider a particular row in the regular bin structure. The utilization of all the bins in this row is given in Figure 4(a). The unequal bin structure constructed from the

regular bin structure is illustrated in Figure 4(b). To get the equation for the new bin structure, from Figure 4 let,

- OB_i: x-coordinate of the boundary of bin i corresponding to the regular bin structure
- NB_i: x-coordinate of the boundary of bin i corresponding to the unequal bin structure

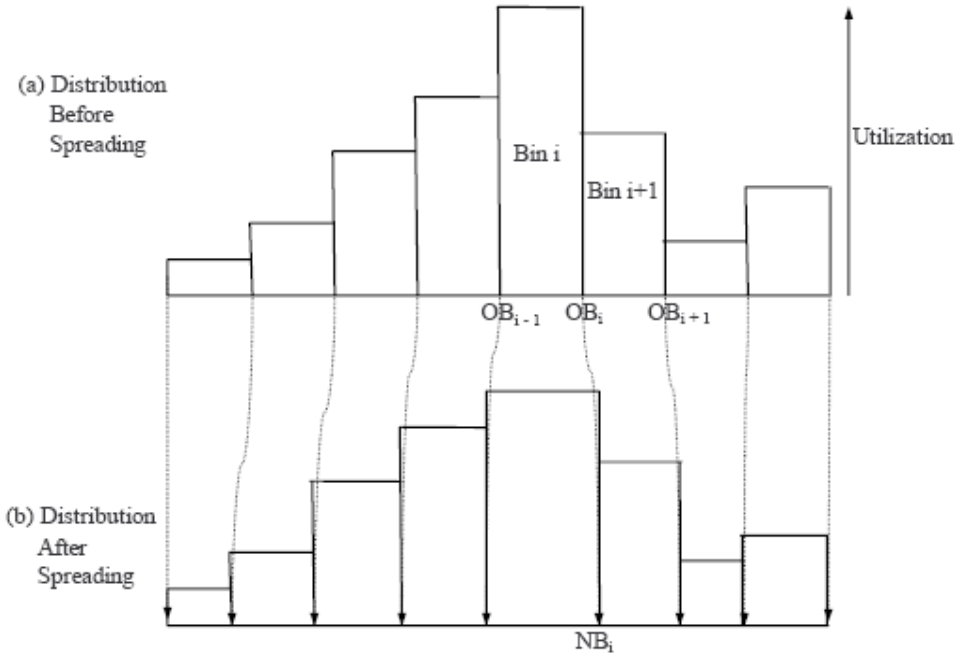

Figure 4: (a) Regular Bin Structure (b) Unequal Bin Structure and Utilization after Shifting.

Then,

$$NB_i = \frac{OB_{i-1}(U_{i+1} + \delta) + OB_{i+1}(U_i + \delta)}{U_i + U_{i+1} + 2\delta}$$

.

The intuition behind the above formula is to construct the new bin such that it averages the utilization of bin i and bin i + 1. To avoid cross-over of boundaries we need the parameter δ which is set to a value of 1.5.

For performing the linear mapping of cells, if x_j is the x-coordinate of cell j in bin i before mapping (obtained from the Global Optimization step) and x'_j is x-coordinate of cell j in bin i after mapping then:

$$\frac{x_j - OB_{i-1}}{OB_i - OB_{i-1}} = \frac{x\prime_j - NB_{i-1}}{NB_i - NB_{i-1}}$$

.

During the initial placement iterations, bins in the center of the placement region have an extremely high bin utilization value. Consequently, cells in such bins will have a tendency to shift over large distances. This effect will get added over iterations and result in a final placement with a high value of the total wirelength. Therefore, to control the actual distance moved by any cell during shifting, the authors introduce two movement control parameters, α_x and α_y (< 1) for the x and y

dimensions. For the x-dimension, the actual distance moved by cell j is $\alpha_x \cdot |x'_j - x_j|$. The expressions for α_x and α_y are:

$$\alpha_y = 0.02 + \frac{0.5}{max(U_i)}$$

$$\alpha_x = 0.02 + \left(\frac{0.5}{max(U_i)}\right)\left(\frac{averageCellWidth}{cellHeight}\right)$$

After the cells have been shifted in the x and y dimensions, additional forces need to be added to them so that they do not collapse back to their previous positions during the next Global Optimization step. This is achieved by connecting each cell to a corresponding pseudo pin added at the boundary of the placement region. The pseudo pin and pseudo net addition is illustrated in Figure 5.

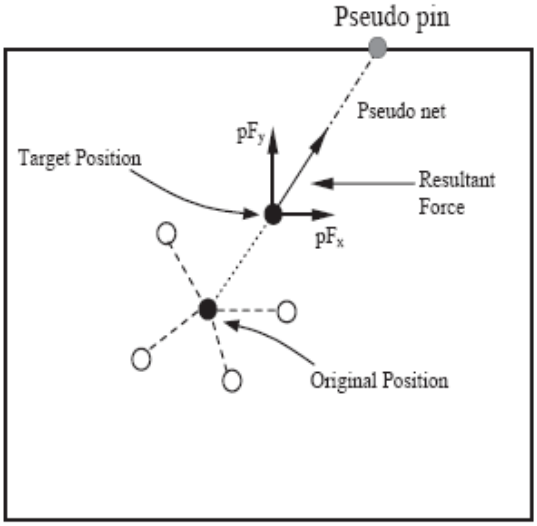

Figure 5: Pseudo Pin and Pseudo Net Addition.

During each iteration of Global Placement, the spreading forces are generated afresh based on the cell positions obtained after the Cell Shifting step. They are not accumulated over iterations. If,

- pFx: x-component of resultant force on cell j at its target position
- pFy: y-component of the force
- pDx: x-component of the distance between the pseudo pin and target position of cell j
- pDy: y-component of the distance

Then, the position of the pseudo pin can be determined by the intersection of the resultant force vector with the chip boundary. A pseudo net for cell j is one which connects the cell from its target position to its pseudo pin. The spring constant for the pseudo net is given by:

$$\beta = \frac{\sqrt{pF_x^2 + pF_y^2}}{\sqrt{pD_x^2 + pD_y^2}}.$$

In order to reduce further the wirelength the authors use the Iterative Local Refinement technique. This technique is interleaved with the Cell Shifting and Global Optimization steps during the WIGP stage. It also tries to reduce the current maximum bin utilization so as to speed-up the convergence of the algorithm. This technique employs a regular bin structure to estimate the current utilization of a placement region for performing wirelength improvement. Cells are then moved from source to target bins based upon the wirelength improvement (based on the half-perimeter measure) and target bin utilization. During the first iteration of the WIGP stage, the width and height of each bin for the Refinement is set to 5 times that of the bin used during Cell Shifting. The width and height of the bins are gradually brought down to the values used in the Cell Shifting step over subsequent iterations of the WIGP stage.

For every cell present in a bin we compute four scores corresponding to the four possible cell movement directions. For calculating the score, we assume that a cell is moving from its current position in a source bin to the same position in a target bin which is adjacent to it. Each score is a weighted sum of two components: The first being the wirelength reduction for the move. The second being a function of the utilization of the source and target bins. If all the four scores are negative, the cell will remain in the current bin. Otherwise, it will move to the target bin with the highest score for the move. The Iterative Local Refinement technique is followed by Cell Shifting in which we add the spreading forces as described previously to reflect the current placement.

The Detailed Placement stage legalizes the solution obtained from global placement. It assigns all the standard cells to pre-defined rows in the placement region. Within each row, the cells are then assigned to legal positions. Once the cells are assigned to the rows in the placement region, any remaining overlap among them is removed. During legalization, the detailed placement also tries to further reduce the wirelength by employing a technique similar to Iterative Local Refinement.

Finally the authors have shown experimentally that the runtime of FastPlace is roughly $O(n^{1.370})$, where n is the circuit size given by the number of pins.

5. DRAWBACKS AND COMPARISONS

In this section we present some drawbacks of the previous algorithms and try to compare them.

Overall the algorithms are well behaved and their solution succeeds a very good wirelength value. However their solution is usually not the optimal as the objective function they use minimizes the squared wirelength and not the wirelength itself. They are also designed for flat and not hierarchical designs and this is a clear drawback for the running time.

PROUD deals with cell overlapping, by making partitions in a hierarchical way. However a bad decision at a higher level affects the placement results of lower levels. For example when PROUD makes a partition at a higher level it separates the modules based on their area but doesn’t take into account other factors like their shape. This could result in a bad decision and the following hierarchies will make their partitions based on this. The only way to overcome a bad partitioning would be with backtracking but this is clearly very time-consuming. Also the “detailed

placement improvement" that the authors propose can dominate the overall runtime of the algorithm.

KraftWerk doesn't suffer from the same drawbacks as PROUD. The iterative algorithm it uses doesn't consider information from the past like the number of iterations or existing partitions, so it is quite adaptable. Also it has the advantage that no hard constraints are used during the placement procedure. This gives the algorithm the flexibility to address a variety of applications and objective functions. However care must be taken when choosing a value for K. A large K could result in oscillation while a small one has the outcome of slow convergence. Another drawback of Kraftwerk is the additional force:

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

which is expensive to compute in each iteration.

Fastplace is superior to the other two. With the cell shifting technique, the analytical placement algorithm can converge in around 20 iterations while KraftWerk may need hundreds of iterations to converge. Also the hybrid net model reduces a lot the total number of nets and makes the execution of the algorithm even faster. Finally the Iterative Local Refinement technique is very effective in minimizing the wirelength. Experimental results have shown that Fastplace is 20-25 times faster than KraftWerk while it results in 10% better wirelength.

6. REFERENCES

[1] R.-S. Tsay, E. S. Kuh, and C.-P. Hsu, "Proud: A fast sea-of-gates placement algorithm", in Proc. ACM/IEEE 25th Design Automation Conf., 1988, pp. 318-323.

[2] H. Eisenmann and F. Johannes, "Generic global placement and floorplanning", in Proc. ACM/IEEE Design Automation Conf., 1998, pp. 269–274.

[3] N. Viswanathan and C. C.-N. Chu, "FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model", in Proc. ACM/IEEE Int. Symp. Physical Design, 2004, pp. 26-33.