FastPlace: Efficient Analytical Placement using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model

Natarajan Viswanathan
nataraj@iastate.edu

Chris Chong-Nuen Chu
cnchu@iastate.edu

Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011-3060

ABSTRACT

In this paper, we present *FastPlace* – a fast, iterative, flat placement algorithm for large-scale standard cell designs. *FastPlace* is based on the quadratic placement approach. The quadratic approach formulates the wirelength minimization problem as a convex quadratic program, which can be solved efficiently by some analytical techniques. However it suffers from some drawbacks. First, the resulting placement has a lot of overlap among cells. Second, the resulting total wirelength may be long as the quadratic wirelength objective is only an indirect measure of the linear wirelength. Third, existing net models tend to create a lot of non-zero entries in the connectivity matrix, which slows down the quadratic program solver. To handle the above problems we propose: (1) An efficient *Cell Shifting* technique to remove cell overlap from the quadratic program solution and produce a global placement with even cell distribution. (2) An *Iterative Local Refinement* technique, to reduce the wirelength according to the half-perimeter measure. (3) A *Hybrid Net Model* which is a combination of the traditional clique and star models. This net model greatly reduces the number of non-zero entries in the connectivity matrix and results in a significant speedup of the solver. Experimental results show that *FastPlace* is on average 13.0 and 97.4 times faster than *Capo* and *Dragon* respectively. Correspondingly, the average wirelength is just 1.0% and 1.6% higher.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuits, Design Aids**]: Placement and routing

General Terms

Algorithms, Design

Keywords

Analytical Placement, Standard Cell Placement, Net Models

1. INTRODUCTION

Placement happens to be one of the most persistent challenges in present day Integrated Circuit design. Designs in current deep sub-micron technology often contain over a million placeable components, and are getting larger by the day. Moreover, because of the dominance of interconnect delay, placement has become a major contributor to timing closure results [17]. It needs to be performed early in the design flow. Hence, it becomes imperative to have an ultra-fast placement tool to handle the ever increasing placement problem size.

In recent years, many placement algorithms have been proposed to handle the widely-used objective of wirelength minimization. These algorithms apply various approaches including *analytical placement* [6, 7, 10, 13, 16, 18], *simulated annealing* [15, 19], and *partitioning/clustering* [3, 4, 20]. Analytical placement is the most promising approach for fast placement algorithm design. Analytical placement algorithms commonly utilize a quadratic wirelength objective function. Although the quadratic objective is only an indirect measure of the wirelength, its main advantage is that it can be minimized quite efficiently. As a result, analytical placement algorithms are relatively efficient in handling large problems. They typically employ a flat methodology so as to maintain a global view of the placement problem [6, 7, 10, 13, 16, 18]. For simulated annealing and partitioning/clustering based approaches, a hierarchical methodology is almost always employed to reduce the problem size to speed up the resulting algorithms [3, 4, 15, 19, 20]. Note that, when the placement problem is so large that a flat analytical approach cannot handle it effectively, a hierarchical analytical approach is beneficial. One of the methods to convert to a hierarchical approach is by incorporating the fine granularity clustering technique proposed by Hu et al. [11]. This technique essentially introduces a two-level hierarchy to reduce the size of large-scale placement problems.

A major concern with the quadratic objective is that it results in a placement with a large amount of overlap among cells. Also, the quadratic objective by itself does not give the best possible wirelength. To handle these problems, Kleinhans et al. [13] use a placement-based bisection technique to recursively divide the circuit and add linear constraints to pull the cells in each partition to the center of the corresponding region. The FM [8] min-cut algorithm is used to improve the bisection and hence the wirelength. Vygen [18] applies a position-based quadrisection technique instead. A splitting-up technique to modify the netlist is also proposed to ensure that the cells will stay in the assigned region. The splitting-up technique also breaks down long nets and hence makes the objective behave like a linear function to some extent. Eisenmann et al. [6] introduce additional constant forces to each cell based on cell distribution to pull cells away from dense regions. Etawil et al. [7] add repelling forces for cells sharing a net to maintain a target distance between them and attractive forces by fixed dummy cells to pull cells from dense to sparse regions. Hu et al. [10] introduce the idea of fixed-point as a more general way to add forces for cell spreading. The last three papers [6, 7, 10] mainly focus on cell spreading. They have not discussed ways to improve the wirelength by a quadratic objective.

In this paper, we present a fast, iterative, flat placement algorithm called *FastPlace* for large-scale standard cell designs. The main contributions of our work are:

- An efficient *Cell Shifting* technique to remove cell overlap. The cell shifting technique roughly maintains the order of the cells in both horizontal and vertical directions as we believe that the quadratic objective function can determine a proper cell ordering. Hence, a high-quality global placement with even cell distribution can be produced in a short time.

- An *Iterative Local Refinement* technique to reduce the wirelength according to the half-perimeter measure. This technique, applied during the final iterations of global placement, makes use of the wirelength and cell distribution information provided by a coarse global placement and hence is very effective.

- A *Hybrid Net Model* which is a combination of the traditional clique and star [14] net models. We prove the equivalence of the Hybrid net model to the traditional clique and star models. On average, the Hybrid Net Model results in a 2.95 times reduction in the number of non-zero entries in the connectivity matrix as compared to the clique model. This results in a significant speedup of the quadratic program solver.

The rest of the paper is organized as follows: Section 2 provides an overview of the algorithm. Section 3 describes the Global Optimization step. Section 4 describes the Hybrid Net Model. Section 5 describes the Cell Shifting technique. Section 6 describes the Iterative Local Refinement technique followed by Section 7 which describes the Detailed Placement technique. Experimental results are presented in Section 8 followed by Section 9 which gives the Conclusions.

2. OVERVIEW OF THE ALGORITHM

FastPlace essentially 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. It is composed of an iterative procedure in which we alternate between Global Optimization and Cell Shifting. Global Optimization involves minimizing the quadratic objective function. During Cell Shifting, the entire placement region is divided into equal sized bins and the utilization of each bin is determined. The standard cells are then shifted around the placement region based on the bin in which they lie and its current utilization. Finally, a spreading force is added to all the cells to account for their movement during shifting.

The second stage is to refine the global placement by interleaving an Iterative Local Refinement technique with Global Optimization and Cell Shifting. The Iterative Local Refinement technique is employed to reduce the wirelength based on the half-perimeter measure and to speed up the convergence of the algorithm. This stage of global placement yields a very well distributed placement solution with a very good value for the total wirelength.

The third stage is that of Detailed Placement. This 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 *FASTPLACE* is summarized in Figure 1 and the individual components of the flow are discussed in more detail in Sections 3-7.

```
Algorithm FASTPLACE

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.
```

Figure 1: The *FASTPLACE* Algorithm.

3. GLOBAL OPTIMIZATION

This section describes the quadratic programming step of global placement refered as Global Optimization, which is the terminology used in [13]. The quadratic placement approach uses springs to model the connectivity of the circuit. The total potential energy of the springs, which is a quadratic function of their length, is minimized[1] to produce a placement solution. In order to model the circuit by a spring system, each multi-pin net needs to be transformed into a set of two-pin nets by a suitable net model. In the following, we assume that this transformation has been applied. The net model used will be discussed in Section 4.

Let n be the number of movable cells in the circuit and (x_i, y_i) the coordinates of the center of cell i. A placement of the circuit is given by the two n-dimensional vectors $x = (x_1, x_2, .., x_n)$ and $y = (y_1, y_2, .., y_n)$. Consider the net between two movable cells i and j in the circuit. Let W_{ij} be its weight. Then the cost of the net between the cells is:

$$\frac{1}{2} W_{ij} [(x_i - x_j)^2 + (y_i - y_j)^2] \tag{1}$$

[1]Equivalently, a force equilibrium state of the spring system is found.

If a cell i is connected to a fixed cell f with coordinates (x_f, y_f), the cost of the net is given by:

$$\frac{1}{2}W_{if}[(x_i - x_f)^2 + (y_i - y_f)^2] \qquad (2)$$

Consequently, the objective function which sums up the cost of all the nets can be written in matrix notation as [9]:

$$\Phi(x,y) = \frac{1}{2}x^T Qx + d_x^T x + \frac{1}{2}y^T Qy + d_y^T y + \text{constant} \quad (3)$$

where Q is an $n \times n$ symmetric positive definite matrix and d_x, d_y are n-dimensional vectors. Since equation (3) is separable into $\Phi(x,y) = \Phi(x) + \Phi(y)$, only the the x-dimension is considered for subsequent discussion, which is:

$$\Phi(x) = \frac{1}{2}x^T Qx + d_x^T x + \text{constant} \qquad (4)$$

Let q_{ij} be the entry in row i and column j of matrix Q. From expression (1), the cost in the x-direction between two movable cells i and j is $\frac{1}{2}W_{ij}(x_i^2 + x_j^2 - 2x_i x_j)$. The first and second terms contribute W_{ij} to q_{ii} and q_{jj} respectively. The third term contributes $-W_{ij}$ to q_{ij} and q_{ji}. From expression (2), the cost in the x-direction between a movable cell i and a fixed cell f is $\frac{1}{2}W_{if}(x_i^2 + x_f^2 - 2x_i x_f)$. The first term contributes W_{if} to q_{ii}. The third term contributes $-W_{if}x_f$ to the vector d_x at row i and the second term contributes to the constant part of equation (4). The objective function (4) is minimized by solving the system of linear equations represented by:

$$Qx + d_x = 0. \qquad (5)$$

Equation (5) gives the solution to the unconstrained problem of minimizing the quadratic function in (4). In *FastPlace*, we solve such an unconstrained minimization problem throughout the placement process. We do not add any constraint to the problem formulation. This is because the spreading forces added during Cell Shifting are produced by pseudo nets connecting the cells to the chip boundary. This only introduces some terms in the form of expression (2) and causes some changes to the diagonal of matrix Q and the vector d_x as described above.

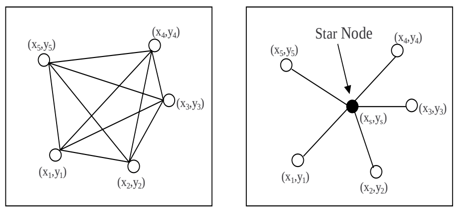
4. HYBRID NET MODEL

To handle the large placement problem size, a fast and accurate technique is needed to solve equation (5). Since matrix Q is sparse, symmetric and positive definite, we solve equation (5) by the pre-conditioned Conjugate Gradient method with the Incomplete Cholesky Factorization of matrix Q as the preconditioner [2, 12]. The runtime of this method is directly proportional to the number of non-zero 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 the matrix Q.

We propose a Hybrid Net Model which is a combination of the clique model and the star model. We show experimentally in Section 8 that the Hybrid Net Model reduces the number of non-zero entries in the matrix Q by 2.95 times over the traditional clique model. In the subsequent discussion, we give a brief overview of the clique and star net models, and introduce the Hybrid Net Model. Then, we prove the equivalence of the clique and star models, and hence the consistency of the Hybrid Net Model.

4.1 Clique, Star and Hybrid Net Models

The clique model is the traditional model used in analytical placement algorithms. In the clique model, a k-pin net is replaced by $k(k-1)/2$ two-pin nets forming a clique. Let W be the weight of the k-pin net. Some commonly used values for the weight of the two-pin nets are $W/(k-1)$ (e.g., [18]) and $2W/k$ (e.g., [6, 13]). The clique model for a 5-pin net is illustrated in Figure 2(a).



(a) Clique Model (b) Star Model
5-Pin Net

Figure 2: Net Models.

Recently, Mo et al. [14] utilized the star net model in a macro-cell placer. 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 2(b). Mo et al. [14] point out that the clique model generates on average 30% more two-pin nets than the star model for the MCNC92 macro block benchmarks, eventhough a star node is created in their model even for two-pin nets. Vygen [18] also switches to a star model for very large nets to reduce the number of terms in the objective function, but has not shown the validity of mixing the clique and star models in quadratic placement. In addition, neither paper has discussed the method to set the weight of the nets introduced by the star model.

In the following subsection we 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 $k\gamma W$ in the star model for any γ, the clique model is equivalent to the star model. Therefore, the two models can be used interchangeably. We 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. We set γ to $1/(k-1)$ in *FastPlace* as it works better experimentally. By using the star model for nets with four or more pins, we will generate much fewer nets and consequently fewer non-zero entries in the matrix Q, than the clique model. By using the clique model for two-pin nets, we will not introduce one extra net and two extra variables per two-pin net as in [14]. We choose to use the clique model for three-pin nets because it is better than the star model for the following reasons: First, if two cells are connected by more than one two-pin or three-pin net in the original netlist, the two-pin nets generated by the clique model between the two cells can be combined and will only introduce a single non-zero entry in the matrix Q. Second, there is no need to introduce an extra pair of variables.

4.2 Equivalence of the Hybrid Net Model to the Clique and Star Net Models

In this subsection, we show that the clique model is equivalent to the star model in quadratic placement if net weights are set appropriately. It follows that the clique, star and Hybrid net models are all equivalent.

LEMMA 1. *For any net in the star model, the star node under force equilibrium is at the center of gravity of all pins of the net.*

PROOF. Consider a k-pin net. Let x_s be the x-coordinate of the star node and let W_s be the weight of the two-pin nets introduced. Then the total force on the star node by all the pins is given by:

$$F = \sum_{j=1}^{k} W_s(x_j - x_s).$$

Under force equilibrium, the total force $F = 0$. Therefore,

$$x_s = \frac{\sum_{j=1}^{k} x_j}{k}. \tag{6}$$

Hence the lemma follows. $\square$

THEOREM 1. *For a k-pin net, if the weight of the two-pin nets introduced is set to W_c in the clique model and kW_c in the star model, the clique model is equivalent to the star model in quadratic placement.*

PROOF. For the clique model, the total force on a pin i by all the other pins is given by:

$$F_i^{clique} = W_c \sum_{j=1, j\neq i}^{k} (x_j - x_i) \tag{7}$$

For the star model, all the pins of the net are connected to the star node. The force on a pin i due to the star node is given by:

$$\begin{aligned} F_i^{star} &= kW_c\,(x_s - x_i) \\ &= kW_c \left(\frac{\sum_{j=1}^{k} x_j}{k} - x_i \right) \qquad \text{by Lemma 1} \\ &= W_c \left(\sum_{j=1}^{k} x_j - kx_i \right) \\ &= W_c \sum_{j=1, j\neq i}^{k} (x_j - x_i) \\ &= F_i^{clique} \end{aligned}$$

As the forces are the same in both models for all pins, the lemma follows. $\square$

5. CELL SHIFTING

Global Optimization gives a placement which minimizes the quadratic objective function. However, it does not consider the overlap among cells. Therefore, the resulting placement has a lot of cell overlap and is not distributed over the placement area. Cell Shifting evens out the placement by distributing the cells over the placement region while retaining their relative ordering obtained during the Global Optimization step.

5.1 Calculation of Bin Utilization

Initially, the placement region is divided into equal sized bins as shown in Figure 3. The area of each bin is such that it can accommodate an average of 4 cells. Based on the placement obtained from the Global Optimization step, the utilization of each bin (U_i) is then computed. U_i is defined as the total area of all cells inside bin i. In calculating U_i

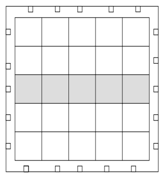


Figure 3: Regular Bin Structure.

we sum the areas of all standard cells which are completely covered by bin i and the overlap area between the bin and the standard cell for cells which partially overlap with bin i. The standard cells are then shifted around the placement region based upon the bin in which they lie and its current utilization.

5.2 Shifting of Cells

Let us consider the case where the cells are shifted in the x-dimension. To shift cells in the x-dimension, we go through every row of the regular bin structure and move cells present in the row. Shifting of cells is a two step process. First, based on the current utilization of all the bins in a particular row an unequal bin structure reflecting the current bin utilization is constructed. Second, every cell belonging to a particular bin in the regular bin structure is then linearly mapped to the corresponding bin in the unequal bin structure. As a result of this mapping, cells in bins with a high utilization will shift in a way so as to reduce its utilization and the overlap among themselves. For shifting cells in the y-dimension we consider every column of the regular bin structure, after all the rows have been considered and follow the two steps mentioned above.

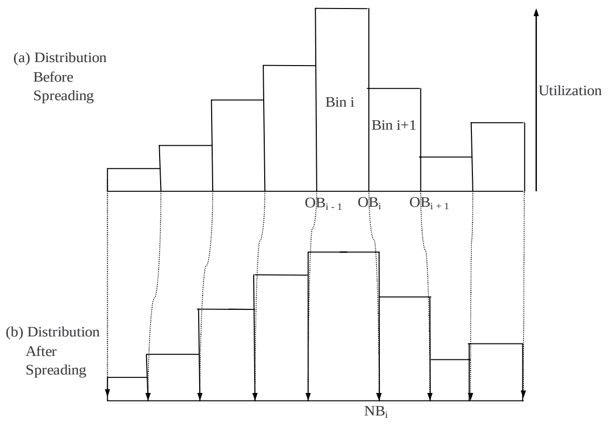

(a) Distribution Before Spreading
Bin i
Bin i+1
Utilization
OB_{i-1}
OB_i
OB_{i+1}
(b) Distribution After Spreading
NB_i

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

To illustrate the shifting in the x-direction, consider a particular row in the regular bin structure (shaded row in Figure 3). 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

Then,

$$NB_i = \frac{OB_{i-1}(U_{i+1}+\delta) + OB_{i+1}(U_i+\delta)}{U_i + U_{i+1} + 2\delta} \quad (8)$$

The intuition behind the above formula is to construct the new bin such that it averages the utilization of bin i and bin $i+1$. The reason for having the parameter δ is as follows: Let, $\delta = 0$ and $U_{i+1} = 0$, then from equation (8) it can be seen that, $NB_i = OB_{i+1}$ and $NB_{i+1} = OB_i$. This results in a cross-over of bin boundaries in the new bin structure which results in improper mapping of the cells. To avoid this cross-over we need the parameter δ which is set to a value of 1.5.

For performing the linear mapping of cells, If,

- x_j: x-coordinate of cell j in bin i before mapping (obtained from the Global Optimization step)
- $x\prime_j$: x-coordinate of cell j in bin i after mapping

Then,

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

or,

$$x\prime_j = \frac{NB_i(x_j - OB_{i-1}) + NB_{i-1}(OB_i - x_j)}{OB_i - OB_{i-1}} \quad (9)$$

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 will perturb the current placement solution by a large amount. 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, we introduce two *movement control parameters,* α_x and α_y (< 1) for the x and y dimensions. α_x and α_y are increasing functions, inversely proportional to the maximum bin utilization and have a very small value during the initial placement iterations. For the x-dimension, the actual distance moved by cell j is $\alpha_x \mid x\prime_j - x_j \mid$. This is just a fraction of the total distance to be moved by the cell.

This way, the cells are shifted over very small distances during the initial placement iterations. During the later placement iterations, the cells will be distributed quite evenly and hence will not have a tendency to shift over large distances. Then, α can take a larger value to accelerate convergence. The expressions for α_x and α_y are:

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

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

5.3 Addition of Spreading Forces

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.

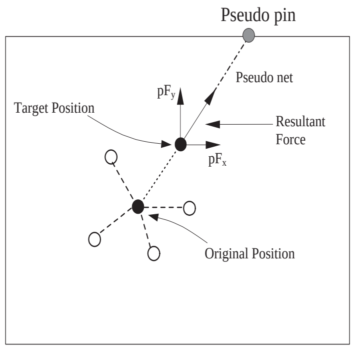


Figure 5: Pseudo Pin and Pseudo Net Addition.

Let (x_j, y_j) and (x_j^f, y_j^f) be the *original* and *target* position of cell j before and after Cell Shifting. Since (x_j, y_j) is the equilibrium position obtained by the Global Optimization step, the total force acting on cell j in this position is zero. When it is moved to the target position it will experience a force due to its connectivity with the other cells or star nodes in the placement region. This force can also be viewed as the force required to move the cell from the *original* to the *target* position. The spreading force added to the cell corresponds to this force experienced by it in its *target* position. 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,

- pF_x: x-component of resultant force on cell j at its *target* position due to cells/star nodes connected to it.
- pF_y: y-component of the force
- pD_x: x-component of the distance between the pseudo pin and target position of cell j
- pD_y: 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 $\beta = \frac{\sqrt{pF_x^2 + pF_y^2}}{\sqrt{pD_x^2 + pD_y^2}}$.

Since the pseudo pin is a fixed pin present at the boundary, we know from expression (2) and the subsequent analysis in Section 3, that only the diagonal of matrix Q and the d_x and d_y vectors need to be updated for every cell. Hence, it takes only a single pass of $O(n)$ time, where n is the total number of movable cells in the circuit, to regenerate the connectivity matrix for the next Global Optimization step. Thus we have incorporated an extremely fast Cell Shifting technique to distribute the cells over the placement region.

6. ITERATIVE LOCAL REFINEMENT

As previously stated, the quadratic objective function on its own does not yield the best possible result in terms of wirelength as it is just an indirect measure of the linear wirelength.To offset this disadvantage, we incorporate an Iterative Local Refinement technique to further reduce the wirelength.

The Iterative Local Refinement technique is interleaved with the Cell Shifting and Global Optimization steps during the WIGP stage. This technique acts on a coarse global placement obtained from the previous stage and hence is very effective in minimizing the wirelength. Unlike other approaches, this technique uses the actual position of a cell and the half-perimeter bounding rectangle measure of all nets connected to the cell for moving it around the placement region. The technique is based on a greedy heuristic which mainly tries to minimize the wirelength while trying to reduce the current maximum bin utilization so as to speed-up the convergence of the algorithm.

6.1 Bin Structure

This technique also 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 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. Such large bins are constructed to enable cell movement over large distances. This is to minimize the wirelength of long nets which might span a large part of the placement area. 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.

6.2 Description of the Technique

Once the utilization of all the bins in the placement region has been determined, we traverse through all the cells in the placement region and determine their respective *source* bins. 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. That is, we move the cell by one bin width. Each score is a weighted sum of two components: The first being the wirelength reduction for the move. The wirelength is computed as the total half-perimeter of the bounding rectangle of all nets connected to the cell. Hence it is much more accurate than the quadratic objective function used in the Global Optimization step. The second being a function of the utilization of the source and target bins. Since the Local Refinement technique is mainly used to reduce the wirelength, a higher weight is used for the first component. 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. During one iteration of the Local Refinement, we traverse through all the bins in the placement region and follow the above steps for cell movement. Subsequently, this iteration is repeated until there is no significant improvement in the wirelength.

The Iterative Local Refinement technique is then followed by Cell Shifting in which we add the spreading forces as described previously to reflect the current placement.

7. DETAILED 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. The difference is that during detailed placement, the technique acts on cells which have been assigned to the actual rows present in the placement region. Besides, it puts a higher weight on the utilization factor than the wirelength factor because the emphasis is on removal of overlap among cells to obtain a legalized placement.

8. EXPERIMENTAL RESULTS

Ckt	#Nodes	#Tnls	#Nets	#Pins	#Rows
ibm01	12506	246	14111	50566	96
ibm02	19342	259	19584	81199	109
ibm03	22853	283	27401	93573	121
ibm04	27220	287	31970	105859	136
ibm05	28146	1201	28446	126308	139
ibm06	32332	166	34826	128182	126
ibm07	45639	287	48117	175639	166
ibm08	51023	286	50513	204890	170
ibm09	53110	285	60902	222088	183
ibm10	68685	744	75196	297567	234
ibm11	70152	406	81454	280786	208
ibm12	70439	637	77240	317760	242
ibm13	83709	490	99666	357075	224
ibm14	147088	517	152772	546816	305
ibm15	161187	383	186608	715823	303
ibm16	182980	504	190048	778823	347
ibm17	184752	743	189581	860036	379
ibm18	210341	272	201920	819697	361

Table 1: Placement Benchmark Statistics.

The benchmarks used in our experiments are derived from the ISPD-02 suite downloaded from [1]. These benchmarks consist of macro blocks and hence had to be modified to be tested on *FastPlace*. The height of all the macro blocks was brought down to the standard cell height. The average width of all the modules in the original benchmark was computed and the width of all macros exceeding 4 times the average width was assigned to a value of 4× average width. All designs in the derived set have a whitespace of 10%. The IBM-Place Benchmarks used in *Dragon* [19] cannot be used because they do not have any connectivity information between the movable cells and the fixed terminals on the placement boundary. This information is essential for a quadratic placement approach. Statistics for the placement benchmarks are given in Table 1.

To determine the effect of the Hybrid net model on the number of entries in matrix Q and on the runtime, we consider two implementations of *FastPlace* in C. One incorporating the clique model and the other incorporating the Hybrid net model. Table 2 gives the results for the two implementations. It can be seen that on average, the Hybrid model leads to $2.95X$ fewer non-zero entries in matrix Q as compared to the clique model over the 18 benchmark circuits. Also, on average, the total runtime of the placer is $1.5X$ lesser for the Hybrid net model.

Table 3 gives a break-up of the total runtime of *FastPlace* for all circuits. We incorporate the Hybrid net model in

Ckt	#Non-zero Entries		Ratio	Runtime
	(Clique)	(Hybrid)	*(#Clique/ #Hybrid)*	*(Clique/ Hybrid)*
ibm01	109183	41164	2.65	1.5
ibm02	343409	70014	4.90	2.4
ibm03	206069	74680	2.76	1.4
ibm04	220423	84556	2.61	1.2
ibm05	349676	108282	3.23	1.3
ibm06	321308	106835	3.01	1.6
ibm07	373328	147009	2.54	1.3
ibm08	732550	173541	4.22	2.0
ibm09	478777	185102	2.59	1.4
ibm10	707969	251101	2.82	1.6
ibm11	508442	230865	2.20	1.2
ibm12	748371	270849	2.76	1.6
ibm13	744500	295048	2.52	1.5
ibm14	1125147	456474	2.46	1.3
ibm15	1751474	607289	2.88	1.4
ibm16	1923995	668491	2.88	1.3
ibm17	2235716	753507	2.97	1.4
ibm18	2221860	711702	3.12	1.4
Avg			**2.95**	**1.5**

Table 2: Clique net model vs Hybrid net model.

FastPlace to obtain these results. Columns 2-4 of Table 3 give the Global Optimization, Cell Shifting, Iterative Local Refinement and Detailed Placement times respectively. It can be seen that on average, Cell Shifting takes only 9.6 % of the total runtime over the 18 benchmarks. This demonstrates the efficiency of the Cell Shifting technique in distributing the cells over the placement region in a very short time.

Ckt	Global Opt. (sec)	Cell Shifting (sec)	Iterative Local Rfnment (sec)	Det. Place (sec)	Total Time
ibm01	3.75	1.44	6.37	1.55	13s
ibm02	8.43	3.05	17.87	3.83	33s
ibm03	10.03	3.59	16.74	2.12	33s
ibm04	11.83	4.13	19.72	3.55	39s
ibm05	10.91	6.23	25.83	8.27	51s
ibm06	13.27	3.91	25.04	3.21	45s
ibm07	33.12	7.81	33.09	4.47	1m 19s
ibm08	32.19	8.94	44.47	7.31	1m 33s
ibm09	43.03	12.47	37.40	8.65	1m 42s
ibm10	57.91	12.38	62.03	12.35	2m 25s
ibm11	56.80	14.67	49.20	11.86	2m 13s
ibm12	59.78	12.43	59.71	10.55	2m 23
ibm13	81.31	17.30	63.98	11.80	2m 54s
ibm14	144.06	32.50	135.30	21.90	5m 34s
ibm15	230.72	43.32	214.36	36.06	8m 45s
ibm16	257.41	53.93	292.74	47.99	10m 52s
ibm17	251.69	39.24	348.08	51.37	11m 30s
ibm18	285.57	57.09	345.28	52.98	12m 21s

Table 3: Break-up of total runtime.

FastPlace is compared with two state-of-the-art academic placers - *Capo 8.8* [3] and *Dragon 2.2.3* [19]. All experiments are run on a Sun Sparc-2, 750 MHz machine. We run *MetaPl-Capo8.8* for Solaris, which incorporates *Capo*, orientation optimizer and row ironing, in the default mode. *Dragon* is run in the fixed die mode. The half-perimeter wirelength and runtime results of *Capo*, *Dragon* and *FastPlace* are given in Table 4.

From column 10 of Table 4, it can be seen that on average, *FastPlace* is 13.0 times faster than *Capo* over the 18 benchmarks. The average wirelength of *FastPlace*, from column 5, is just 1.0% higher than *Capo*. From column 11 of Table 4, it can be seen that on average, *FastPlace* is 97.4 times faster than *Dragon*. The average wirelength of *FastPlace*, from column 6, is just 1.6% higher than *Dragon*.

To determine the scalability factor of FastPlace, we plot the runtime versus the total number of pins, which is a good measure of the circuit size, in logarithmic scale for all 18 benchmarks in Figure 6. The data points can be closely approximated by a straight line with slope 1.370. Hence, the runtime of FastPlace is roughly $O(n^{1.370})$, where n is the circuit size given by the number of pins.

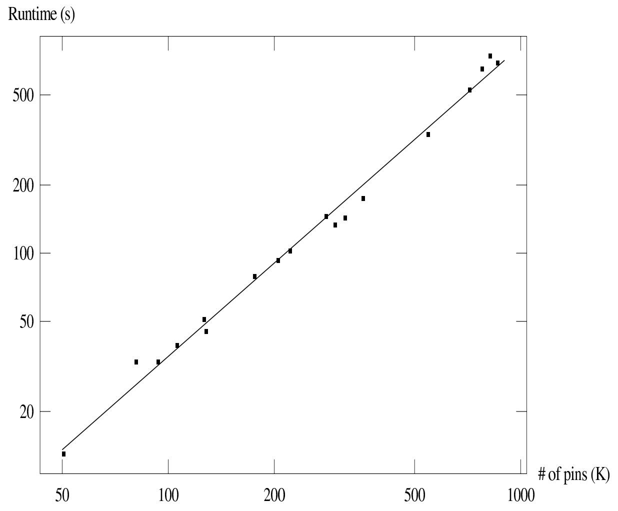


Figure 6: Runtime of FastPlace verses number of pins in circuits in logarithmic scale.

9. CONCLUSIONS

In this paper, we propose an efficient and scalable flat placement algorithm *FastPlace* for large-scale standard cell circuits. *FastPlace* is based on the analytical placement approach and utilizes the quadratic wirelength objective function. The current implementation handles the wirelength minimization problem. It produces comparable placement solutions to state-of-the-art academic placers, but in a significantly lesser runtime. Such an ultra-fast placement tool is very much needed for the timing convergence of the layout phase of IC design.

The runtime of *FastPlace* can be further reduced by incorporating it into the FPI framework in [11] or a general hierarchical framework, and by applying the algebraic multigrid method to solve the system of linear equations (5) [5]. The *FastPlace* algorithm can also be extended to consider other placement objectives like mixed-mode placement, timing driven placement, routing congestion, variable whitespace allocation, etc. Future extensions to the algorithm would be in dealing with the above objectives.

Ckt	Half-Perimeter Wirelength			Wirelength Ratio		RunTime			Speed-up	
	Capo ($\times$1e6)	Dragon ($\times$1e6)	FastPlace ($\times$1e6)	$\frac{FastPlace}{Capo}$	$\frac{FastPlace}{Dragon}$	Capo	Dragon	FastPlace	$\frac{Capo}{FastPlace}$	$\frac{Dragon}{FastPlace}$
ibm01	1.86	1.84	1.91	1.03	1.04	3m 59s	29m 6s	13s	$\times$18.4	$\times$134.3
ibm02	4.06	3.98	4.02	0.99	1.01	7m 15s	31m 13s	33s	$\times$13.2	$\times$56.8
ibm03	5.11	5.31	5.45	1.07	1.03	8m 23s	31m 49s	33s	$\times$15.2	$\times$57.8
ibm04	6.39	6.22	6.63	1.04	1.07	10m 46s	1h 5m	39s	$\times$16.6	$\times$100.0
ibm05	10.56	10.35	10.96	1.04	1.06	10m 44s	1h 48m	51s	$\times$12.6	$\times$127.1
ibm06	5.50	5.45	5.55	1.01	1.02	12m 08s	1h 21m	45s	$\times$16.2	$\times$108.0
ibm07	9.63	9.26	9.56	0.99	1.03	18m 32s	1h 47m	1m 19 s	$\times$14.1	$\times$81.3
ibm08	10.26	9.66	10.01	0.98	1.04	19m 53s	4h 30m	1m 33s	$\times$12.8	$\times$174.2
ibm09	10.56	11.03	11.26	1.07	1.02	22m 50s	3h 43m	1m 42s	$\times$13.4	$\times$131.2
ibm10	19.70	19.46	19.31	0.98	0.99	29m 04s	3h 19m	2m 25s	$\times$12.0	$\times$82.3
ibm11	15.73	15.36	16.03	1.02	1.04	31m 11s	2h 22m	2m 13s	$\times$14.1	$\times$64.1
ibm12	25.83	24.74	25.04	0.97	1.01	30m 41s	3h 48m	2m 23	$\times$12.9	$\times$95.7
ibm13	18.73	19.32	19.46	1.04	1.01	39m 27s	3h 4m	2m 54s	$\times$13.6	$\times$63.4
ibm14	36.69	35.77	36.09	0.98	1.01	1h 12m	7h 37m	5m 34s	$\times$12.9	$\times$82.1
ibm15	43.85	43.39	45.21	1.03	1.04	1h 30m	10h 34m	8m 45s	$\times$10.3	$\times$72.4
ibm16	49.63	49.54	48.43	0.97	0.98	1h 31m	12h 6m	10m 52s	$\times$8.4	$\times$66.8
ibm17	69.07	73.45	68.09	0.99	0.93	1h 43m	26h 54m	11m 30s	$\times$9.0	$\times$140.3
ibm18	47.46	48.59	46.89	0.99	0.96	1h 44m	23h 39m	12m 21s	$\times$8.4	$\times$114.9
Average				**1.010**	**1.016**				**$\times$13.0**	**$\times$97.4**

Table 4: Comparsion of placement results with Capo 8.8 and Dragon 2.2.3.

10. REFERENCES

[1] *http://vlsicad.eecs.umich.edu/BK/ISPD02bench/.*

[2] R. Barrett, M. Berry, and et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994.

[3] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection produce routable placements. In *Proc. ACM/IEEE Design Automation Conf.*, pages 477–482, 2000.

[4] T. Chan, J. Cong, T. Kong, and J. Shinnerl. Multilevel optimization for large-scale circuit placement. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pages 171–176, 2000.

[5] H. Chen, C.-K. Cheng, N.-C. Chou, A. Kahng, J. MacDonald, P. Suaris, B. Yao, and Z. Zhu. An algebraic multigrid solver for analytical placement with layout based clustering. In *Proc. ACM/IEEE Design Automation Conf.*, pages 794–799, 2003.

[6] H. Eisenmann and F. Johannes. Generic global placement and floorplanning. In *Proc. ACM/IEEE Design Automation Conf.*, pages 269–274, 1998.

[7] H. Etawil, S. Arebi, and A. Vannelli. Attractor-repeller approach for global placement. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pages 20–24, 1999.

[8] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proc. ACM/IEEE Design Automation Conf.*, pages 175–181, 1982.

[9] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17:219–229, 1970.

[10] B. Hu and M. Marek-Sadowska. Far: Fixed-points addition and relaxation based placement. In *Proc. Intl. Symp. on Physical Design*, pages 161–166, 2002.

[11] B. Hu and M. Marek-Sadowska. Fine granularity clustering for large scale placement problems. In *Proc. Intl. Symp. on Physical Design*, pages 67–74, 2003.

[12] D. S. Kershaw. The incomplete cholesky-conjugate gradient method for the iterative solution of systems of linear equations. *Journal of Computational Physics*, 26:43–65, 1978.

[13] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich. Gordian: VLSI placement by quadratic programming and slicing optimization. *IEEE Trans. Computer-Aided Design*, 10(3):356–365, 1991.

[14] F. Mo, A. Tabbara, and R. Brayton. A force-directed macro-cell placer. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pages 177–180, 2000.

[15] C. Sechen and A. L. Sangiovanni-Vincentelli. Timberwolf 3.2: A new standard cell placement and global routing package. In *Proc. ACM/IEEE Design Automation Conf.*, pages 432–439, 1986.

[16] G. Sigl, K. Doll, and F. M. Johannes. Analytical placement: A linear or a quadratic objective function. In *Proc. ACM/IEEE Design Automation Conf.*, pages 427–431, 1991.

[17] P. Villarrubia. Important placement considerations for modern vlsi chips. In *Proc. Intl. Symp. on Physical Design*, page 6, 2003.

[18] J. Vygen. Algorithms for large-scale flat placement. In *Proc. ACM/IEEE Design Automation Conf.*, pages 746–751, 1997.

[19] M. Wang, X. Yang, and M. Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pages 260–263, 2000.

[20] M. C. Yildiz and P. H. Madden. Global objectives for standard cell placement. In *Proc. 11th Great Lakes Symposium on VLSI*, pages 68–72, 2001.