A Force-Directed Macro-Cell Placer

Fan Mo, Abdallah Tabbara and Robert K. Brayton
Dept. of EECS, University of California, Berkeley, CA94720

Abstract

In this paper we present a novel force-directed placement algorithm, which is used to solve macro-cell placement problems. A new wire model replaces the traditional clique model and makes possible early awareness of routing congestion. Issues such as cell orientation, overlap elimination, and pad positioning are also considered. Experiments show satisfactory performance and fast run time.

1. Introduction

The force-directed placement algorithm [2,3,7] simulates the mechanics problem in which particles are attached to springs and their movement obeys Hooke's law. In the conventional force-directed methods cells are regarded as points no matter what real shapes and sizes they have. In addition, all the terminals of a cell are concentrated onto a single point. One of the benefits of the single point model is that it clusters wires that connect terminals of the same pair of cells. This approximation is acceptable in standard cell design, because standard cells are small as compared to the entire placement region. However in macro cell placement, such approximation cannot be adopted because large inaccuracy will be introduced. In some cases the largest cell can occupy over half of the entire placement region, so approximating it as a point and gathering all its terminals to that point is obviously not a good idea because of how it may affect the compactness of the final placement. What this means is that we have to use the real shapes and sizes of the macro cells to form the force equation, resulting in a cell overlap problem. The cell overlap problem has been well studied, and several methods were suggested such as introducing repulsive forces for non-connected modules [3], using additional forces to make the placement area have an even density distribution [1], using recursive partitioning [4], and creating some pseudo cells called attractors in low density areas as well as repellors in dense areas [5]. Cell orientation is another problem that is encountered, since it allows more flexibility in placing a cell and decreasing wire length. In addition, previous methods usually require the pad ordering be fixed before placement can proceed. Furthermore, in modern VLSI design flow, placers have to foresee what may happen in routing because the quality of a placement is convincing only after routing is done.

In our algorithm real cell shapes and sizes are used to form the force equation, so terminal positions really matter. The clique wire model is replaced by a star model where an additional point, called star, is created for each net. Thus, while placing macro cells, we also have to place the stars. This model not only saves considerable computation time but also provides a way for estimating routing congestion during placement. Cell orientation is optimized by calculating the reduction in force when taking a new orientation. To eliminate overlapping we adopt a modified version of the density method [1]. Pad positions are determined using a force method and a one dimensional density method. The algorithm is iterative, allowing other algorithms like don't care wire choice to be easily incorporated into the placement flow [9]. The resulting macro-cell placer is fast as compared to modern commercial placers, especially for large designs.

The rest of this paper is organized as follows. In Section 2, we formulate the placement problem and describe our algorithm. Section 3 presents the design flow. Experimental results are given in Section 4, while Section 5 gives some concluding remarks.

2. Algorithm

2.1 The Macro-Cell Placement Problem

A macro-cell is a hard rectangular block with fixed size and shape that can flip and/or rotate during placement. Terminals are connecting points fixed on the cell. Terminals are not restricted to being on the cell boundary, and their position is defined relative to the cell origin. Each cell has a position inside the chip, which is represented as a soft rectangle with a flexible size and shape, and where pads are terminals located on the boundary. A net is a set of terminals physically implemented by wires that connect those terminals. In our algorithm, a star wire model is used where an additional connecting point other than terminals is added for each net, where each terminal connects to its corresponding star through a wire. The goal for a macro cell placement problem is to minimize the chip area, minimize the total wire length, or both. The constraint is that all cells lie within the chip boundary and that no two cells overlap. A further constraint is to find an implementable placement such that it results in a valid routing; this is because a well-compacted placement may not be completely routable.

2.2 Force-Directed Method Based on the Star Wire Model

The use of star wire model instead of a clique model is one essential feature of our algorithm. Since a star is created even for two terminal net, the new wire model causes the number of wires to change. A comparison between those two models can be made as follows. The total wire number for the clique model, which is widely used in other force-directed algorithms, is:

$$n_C = \frac{N_n}{2} \cdot \sum_{i=2}^{\infty} TpN[i] \cdot i \cdot (i-1) \quad (1)$$

where N_n is the net number, $TpN[i]$ is the percentage of nets that have i terminals. The result of star model is:

$$n_S = N_n \cdot \sum_{i=2}^{\infty} TpN[i] \cdot i \quad (2)$$

Statistic from the MCNC92 benchmark examples we use in our experiments shows that averagely n_C is 30% greater than n_S, so considerable computation time can be saved by using star wire model. Another interesting feature of the star wire model is the improved estimation of wire congestion, and this point will be discussed in sub-section 2.5.

In force-directed methods, attractive forces, which obey Hooke's law, are applied on objects connected by wires. These objects can be terminals, pads or stars. The attractive force of a cell is the sum of attractive forces of its terminals:

$$\vec{F}_A[c] = \sum_{t=1}^{NT[c]} \vec{F}_T[c][t] = -\sum_{t=1}^{NT[c]} k[c][t] \cdot \left(\vec{P}_C[c] + \vec{O}_T[c][t] - \vec{P}_S[S[c][t]] \right) \quad (3)$$

where $NT[c]$ is the terminal number of cell c, $P_C[c]$ is the position of the left bottom corner of cell$[c]$, $P_S[S]$ is the star position, $O_T[c][t]$ is the offset of a terminal with respect to the left bottom corner of the cell it belongs to, and $k[c][t]$ is the wire weight (more weight is assigned to wire with tight timing constraint). If only attractive forces are applied, the new position of a cell after one force directed movement is:

$$\vec{P}_C[c] -= \vec{F}_A[c] \Big/ \frac{\partial \vec{F}_A[c]}{\partial \vec{P}} \quad (4)$$

where,

$$\frac{\partial \vec{F}_A[c]}{\partial \vec{P}} = -\sum_{t=1}^{NT[c]} k[c][t] \quad (5)$$

Notice that there isn't a factor 2 in the denominator of (4) as described in [6]. The reason given in [6] is that when two cells are moving towards each other, each of them just needs to move half distance. But in our algorithm, cells are not connected directly by wires. Instead they connect to stars. Similarly attractive forces are computed for stars, and stars move according to the same equation as (4).

2.3 Overlap Elimination

To eliminate cell overlap, we adopt the method in [1]. Instead of adding a repulsive force for overlapping cells, we try to make the cells distribute in the placement region more evenly by introducing a filling force. So it's natural to let regions with higher density become sources of this filling force while regions with lower density become sinks. A two-dimensional bin structure is created on the chip. Square bins are used, so x and y direction can have different bin numbers that depends on the chip aspect ratio. Bin density is calculated as:

$$D[bx][by] = -D_{BAL} + \sum_{\substack{AllCellsCovering \\ bin[bx][by]}} A[c] \Big/ BinSize^2 \quad (6)$$

in which, $A[c]$ is the area that cell c covers the bin. Notice that a cell's contribution is the its fraction of overlap with the bin in question, so if a cell completely covers the bin, its contribution is 1. The term D_{BAL} is the balance density:

$$D_{BAL} = \frac{area_{TOTAL}}{area_{CHIP}} \quad (7)$$

where $area_{CHIP}$ is the area enveloped by the chip boundary, and $area_{TOTAL}$ is the sum of area of all macro-cells and possible routing space. Positive $D[bx][by]$ means the bin is over filled while negative $D[bx][by]$ means the bin needs filling. The bin size is an important factor in the algorithm. A small bin size can ensure the detection of cell overlap but increases computation time, while a large bin size saves time but can hardly detect small overlaps. A reasonable bin size has been experimentally determined as being half the minimum cell size. The additional force formed with respect to bin density is:

$$\vec{f}[bx][by] = BinSize^2 \times \sum_{bx'=1}^{BinNumX} \sum_{by'=1}^{BinNumY} D[bx'][by'] \cdot \frac{P[bx][by] - P[bx'][by']}{\left| P[bx][by] - P[bx'][by'] \right|^2} \quad (8)$$

where $P[bx][by]$ is the center position of a bin. More efficient computation can be achieved by using variable resolution based on distance. Also, for adjacent bins, calculation is done bin by bin, but for distant bins, average value of a group of bins can be used.

The filling force F_F a cell receives is the sum of filling forces asserted by bins the cell covers. Combining the filling force with the attractive force formulated above, we get a new force equation:

$$\vec{F}_C[c] = a \cdot \vec{F}_A[c] + d \cdot \vec{F}_F[c] \quad (9)$$

where a and d are weights that allows us to keep a balance between the effects of these two forces. Notice that when attractive and filling forces are combined, the equation (4) can hardly be used, because the differential equation for F_F is not easy to compute. Thus the movement equation becomes:

$$\vec{P}_C[c] += \begin{cases} \vec{F}_C[c] & \left| \vec{F}_C[c] \right| < f_{LIMIT} \\ f_{LIMIT} \cdot \dfrac{\vec{F}_C[c]}{\left| \vec{F}_C[c] \right|} & \left| \vec{F}_C[c] \right| \geq f_{LIMIT} \end{cases} \quad (10)$$

in which f_{LIMIT} is a limit of the maximum distance a cell can move.

2.4 Orientation Selection

A cell has a total of eight possible orientations, and these can be derived by flipping, rotating, or the combination of these two operations. To evaluate each of these orientations, a gain of force reduction and/or density improvement is computed. Cells with negative or zero gain are removed from the candidate list immediately, while cells with larger positive gain are good candidates. However we have to be very careful in choosing cells to change orientation. If two or more cells simultaneously change orientation, the overall result may sometimes get worse. The reason is that the gain is calculated with the assumption that all other cells are fixed. Therefore two additional criteria for selecting the best orientation are added. First, we limit the number of cells that can take new orientation. In the extreme one can allow only one cell to change orientation at each step. But this is inefficient, so after some experimentation a limit of 10% of total number of cells was chosen. The second criteria is that if the current candidate has a large number of connections to some already selected cell(s), then this cell is discarded. Although chances might be that choosing this cell as well can actually lead to better result, the cell, if discarded at the current step, will still be among the candidates at the next step.

2.5 Routing Estimation

There exists two different routing modes, over-the-cell routing and non over-the-cell routing [7,8]. Over-the-cell routing is becoming more popular with the increasing number of metal layers, so most modern integrated circuits belong to this category. The flexibility of routing over the cell greatly relieves the burden of routing estimation from the placement stage. We consider three routing regions for non-over-the-cell routing, that is, the star region, the terminal region, and the star-to-terminal

region. For over-the-cell routing, only the terminal region is taken into account.

The star connects two or more wires, so additional density is added to the bin the star stays in. The contribution to bin density by a star is:

$$d_S = N_T[s] \cdot \frac{w}{2 \cdot BinSize} \cdot \frac{1}{L} \tag{11}$$

where $N_T[s]$ is the number of terminals star s connects to, w is the wire pitch, and L is number of metal layers. Equation (11) is derived with the approximation that a wire consumes a metal segment with length of half bin size and width of metal pitch to leave that bin. We also assume that each layer has equal routing area consumption.

Routing is estimated in the terminal region by calculating the keep-out distance of a cell edge. The keep-out distance is composed of two parts. The first part, denoted by *KOc*, is a constant halo reserved for ground or power ring. The second part is a variable that relates to the routing near the cell edge. Suppose, for a horizontal edge e of cell c, *left* is the number of wires going left and *right* the number of wires going right, the keep out distance $KO[c][e]$ is:

$$KO[c][e] = w \cdot \max(left, right) + KO_C[c][e] \tag{12}$$

where, $KO_C[c][e]$ is a constant keep-out distance reserved for gound or power rings, and w is the wire pitch. Whether a wire is *left* or *right* is determined by the relative position of the star to the shadow of the cell edge. If the star is within the shadow of the cell edge, then it's neither *left* nor *right*. Vertical cell edges have a similar equation. Clearly, if all wires have their stars on one side of the cell shadow, the keep-out distance is at a maximum.

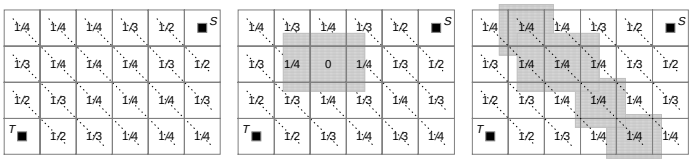

Figure. 1. Routing estimation in the region between terminal and star

The estimation for the star-to-terminal region is illustrated in Figure 1, where a routing rectangle is bounded by terminal T and star S. We assume the probability of a wire taking any bin on the same wave front (the 45 degree line) is the same. Therefore as shown in the left figure, the two bins nearest to T both have 1/2 probability. Their neighbouring bins have 1/3 and so on. There are special cases that one or more bins are fully occupied (i.e. bin density ≥ 1) then wires are unlikely going across these bins. As shown in the central figure, bin(3,3) is completely covered by macro cells. In such case, this bin cannot be used for routing, therefore the other bins on the same wave front should share the left over routing density. If all bins on the same wave front are fully occupied by macro cells, it is no way of completing the connection within the routing region. To simplify the problem we just put equal probability to each of these bins as shown in the right figure. After deriving probability p_R for each bin, the density contributed to the bin by routing is:

$$d_R[bx][by] = p_R[bx][by] \cdot \frac{w}{BinSize} \cdot \frac{1}{L} \tag{13}$$

in which w is the wire pitch, and L is the number of metal layers. Notice that the computation of d_R doesn't include the bin T and S occupies because KO and d_S are formed to reflect their effects respectively. However, directly employing the model derived above may lead to an over estimation of routing resources because in a multi-terminal net the bounding rectangles formed for each terminal may overlap. Actually two wires leaving from two different terminals can merge into one wire before they reach the star. Thus, after calculating d_R for each terminal of a net independently with (13), we sum all d_R's and then divide by R, the number of routing rectangles covering the bin:

$$d'_R[bx][by] = \frac{1}{R} \sum d_R[bx][by] \tag{14}$$

$d'_R[bx][by]$ is then added to the corresponding bin density $D[bx][by]$ in (6).

2.6 Pad Positioning and Chip Boundary Determination

The chip itself is a kind of soft cell with variable size and aspect ratio. The pads, terminals of this special cell, are also soft in the sense that their positions are not determined. However, two main constraints apply. First, the chip aspect ratio has an upper limit. The other constraint is that a pad has its own size and occupies a certain segment of the chip boundary. The force-directed method is again used for pad positioning, where the only difference is that movement is one-dimensional. To make the pads spread evenly along the chip boundary, a one-dimensional bin structure is built for the computation of density. Similar to (8), we have

$$f[b] = BinSize_P \cdot \sum_{\substack{b'=b-BM \\ b' \neq b}}^{b+BM} D[b'] \cdot \frac{P[b] - P[b']}{\left|P[b] - P[b']\right|^2} \tag{15}$$

where $D[b]$ is the density of one-dimensional bin, and $P[b]$ is the position of the bin. Notice that the summation is within the range of $\pm BM$ bins due to the circular nature of the one-dimensional bin structure. BM is chosen to cover half the perimeter of the chip boundary. The resultant filling force is combined with attractive forces to determine the movement of the pad. This enables pad positioning during placement.

3. The Iterative Placement Flow

A placement flow is designed to organize all the algorithmic parts such that performance and speed goals are achieved by tuning the order and weight of different part at different stage. In the first stage, we focus on finding a good relative position for each macro cell without paying much attention to cell overlap and orientation. Macro-cells can move long distance to reach their new position. The completion of this stage will give a good starting point for the next stage where factors like cell orientation and overlap are considered. In the following stage, attractive and filling forces both take action but the latter plays a more important role by receiving a gradually increasing weight. In the final stage, cells are almost at their final position and are unlikely to find a new and better orientation, so there is no big change in total wire length. Filling forces dominate and cells just move within a small range to eliminate overlap. Finally a clean-up procedure is invoked to eliminate remaining cell

Example	Characters						Over-the-cell routing, 4 metal layers						Non-over-the-cell routing, 2 metal layers					
	no. cell	no. pad	no. term	no. net	cell area (um^2) min/ave/max	cell aspect min/ave/max	area (mm^2)		wire length(mm)		run time(min)		area (mm^2)		wire length(mm)		run time(min)	
							SE	This	SE	This	SE	This	BE	This	BE	This	BE	This
ami33 #	33	42	522	123	58/350/744	1.0/2.0/4.2	0.052	0.030	11.6	11.0	2.67	1.20	0.034	0.042	18.7	**20.5**	1.35	**7.48**
ami49 #	49	22	953	408	635/7.2k/55k	1.7/2.2/3.2	0.882	0.609	123.5	100.4	2.77	**2.92**	0.751	0.816	172.4	**173.9**	0.80	**4.75**
playout #	62	192	4.6k	1.6k	96/14.2k/150k	1.4/1.9/5.6	2.631	2.187	1599.6	1545.8	26.50	3.20	3.039	2.562	5250.9	4945.8	3.13	**4.12**
alu2_50 *	71	16	511	81	11/102/1k	3.9/11.6/23.0	0.091	0.057	24.6	21.5	5.03	4.40	0.063	0.061	55.8	55.7	2.62	**6.28**
apex6_50 *	142	234	1.2k	277	11/111/495	4.8/9.4/27.0	0.131	0.079	82.3	73.5	6.72	5.53	0.186	0.164	374.9	358.2	4.32	**8.32**
random1b&	2k	60	7.6k	1.8k	68/81/99	1.0/1.1/1.2	0.628	0.471	1465.5	1359.3	920.00	27.33	0.485	0.462	4519.8	4449.1	543.05	83.20
random2b&	100	60	3.3k	497	21/6.4k/36k	1.0/4.0/8.0	1.158	1.040	1044.1	928.1	141.33	15.75	3.543	3.494	3660.5	**3681.6**	30.33	25.20
random3b&	200	180	30.6k	8.1k	1k/14k/22k	1.0/1.1/1.2	---	12.062	---	78.6k	>1000	37.43	41.254	38.147	228.2k	215.4k	114.4	72.33

Metal 1/2, 0.3um, spacing 0.3um; metal 3/4, 0.6um, spacing 0.6um. Area is measured without pads. Time out limit is 1000 minutes.
MCNC92 benchmark[10], cell size scaled by 10. * PLA[11]. & randomly generaged IC.

Table 1. Testbench Examples and Results

overlap that has not been detected because of limited bin resolution. The placement flow is shown below.

```
//STAGE 1
Initialize_and_ZeroSize_Cells();
SetCellsSizeZero();
loop Stage1IterationNumber {
        ComputeAttractiveForces_and_Move_Stars();
        ComputeAttractiveForces_and_Move_Cells();
        DetermineChipBoundary();
        ComputeAttractiveAndFillingForce_and_Slide_Pads();
}
explode();
//STAGE 2
loop Stage2IterationNumber {
        IncreaseCellSize();
        ComputeAttractiveForces_and_Move_Stars();
        ComputeAttractiveForces_Cells();
        Routing_Estimation();
        ComputeBinDensity();
        ComputeFillingForces_Cells();
        MoveWithLimit_Cells();
        ComputeOrientationGain_and_ChooseOrientation_for_Cells();
        DetermineChipBoundary();
        ComputeAttractiveAndFillingForce_and_Slide_Pads();
}
//STAGE 3
while (exist bin, density > densityThreshold) {
        ComputeAttractiveForces_and_Move_Stars();
        ComputeAttractiveForces_Cells();
        Routing_Estimation();
        ComputeBinDensity();
        ComputeFillingForces_Cells();
        MoveWithLimit_Cells();
        DetermineChipBoundary();
        ComputeAttractiveAndFillingForce_and_Slide_Pads();
}
CleanUp()
```

4. Experimental Results

The entire placement system was written in JAVA. We compared our algorithm with *Cadence Silicon Ensemble* version 5.0 with over-the-cell routing, and *Cadence Block Ensemble* 97 version with non-over-the-cell routing. Their routers are also employed to route the designs placed by our algorithm. All the experiments were done on a Sun Ultra2 workstation with 256M memory and 2 CPUs. The JAVA interpreter is Sun's jdk1.2 and the operating system is SunOS5.5.1. A comparison is made in terms of area, total wire length and run time. In Table 1, the testbench examples are from MCNC92 block placement benchmark [10], PLA based ICs [11] and randomly generated macro cell ICs. *Random1b* is designed to have a large number of cells number with similar sizes, and aspect ratios, and a moderate number of nets. *Random2b* has a great difference in cell sizes and aspect ratios. *Random3b* has a large number of nets. The bold items in the table are where our algorithm gives worse results. Experimental results show that our algorithm can give satisfactory performance in most cases, and for large examples runs relatively fast.

5. Conclusion

In this paper, we present a macro-cell placer based on a force-directed method. Cell size and shape are taken into consideration so that terminal positions affect the placement. The star wire model reduces computation time and improves routing estimation. Bin density is employed to make cells and pads evenly distributed. Experimental results show satisfactory performance and fast speed.

Acknowledgements

The authors would like to acknowledge support from GSRC.

References

[1] H.Eisenmann and F.M.Johannes, "Generic Global Placement and Floorplanning", In *Proceedings of the 35th Design Automation Conference*, pages.269-274, 1998

[2] S.Goto, "An Efficient Algorithm for the Two-Dimensional Placement Problem in Electrical Circuit Layout", *IEEE Trans. CAS*, vol.28, no.1, Jan1981, pages 12-18

[3] N.R.Quinn,JR. and M.A.Breuer, "A Forced Directed Component Placement Procedure for Printed Circuit Boards", *IEEE Trans. CAS*, vol.26, no.6, Jun1979, pages 377-388

[4] 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, pages.356-365

[5] H.Etawil, S.Areibi and A.Vannelli, "Attractor-Repeller Approach for Global Placement", In *Proceedings of International Conference on Computer-aided Design*, pages.20-24, 1999

[6] "Placement and Routing of Electronic Modules", edited by M.Pecht, New York: M.Dekker, 1993

[7] "Algorithms for VLSI physical design automation", edited by N.Sherwani, 3rd ed. Boston: Kluwer Academic Publishers, 1999

[8] Routing in the Third Dimension from VLSI chips to MCMs, edited by N.Sherwani, S.Bhingarde, A.Panyam, IEEE Press,1995

[9] P.Chong, Y.Jiang, S.Khatri, F.Mo, S.Sinha and R.Brayton, "Don't Care Wires in Logic/Physical Design", In the Proceedings of *International Workshop on Logic Synthesis*, pages 1-9, 2000

[10] MCNC92 benchmark, http://www.cbl.ncsu.edu/CBL_Docs/lys92.html

[11] S.Khatri, "Cross-talk Noise Immune VLSI Design using Regular Layout Fabrics", PhD thesis, University of California at Berkeley, Spring 2000