Attractor-Repeller Approach for Global Placement

Hussein Etawil, Shawki Areibi, and Anthony Vannelli

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

Abstract

Traditionally, analytic placement used linear or quadratic wirelength objective functions. Minimizing either formulation attracts cells sharing common signals (nets) together. The result is a placement with a great deal of overlap among the cells. To reduce cell overlap, the methodology iterates between *global optimization* and *repartitioning* of the placement area. In this work, we added new attractive and repulsive forces to the traditional formulation so that overlap among cells is diminished without repartitioning the placement area. The superiority of our approach stems from the fact that our new formulations are convex and no hard constraints are required. A preliminary version of the new placement method is tested using a set of MCNC *benchmarks* [1] and , on average, the new method achieved 3.96% and 7.6% reduction in wirelength and CPU time compared to TimberWolf v7.0 in hierarchical mode [10].

1. Introduction

Due to its complexity, VLSI layout task has always been performed in two phases: cell *placement* and cell *routing*. Cell placement is a crucial element to accomplish design objectives such as minimum interconnection wirelength, minimum chip area, minimum power dissipation and other performance related targets. Popular placement techniques include *Min-cut* ([9],[1]), *randomized iterative improvement* ([3],[10]), and *analytic* placement [7, 5]. To the best of our knowledge, the state-of-art placers are based on randomized methods (Simulated Annealing) [10], and the combination of analytic and local improvement methods (Gordian/Domino [7], [8]). While the Annealing method produces very good quality answers in terms of wirelength and timing requirements, it suffers from excessive computational efforts. On the other hand, Gordian/Domino achieves comparable solution quality in terms of wirelength and it is about 5 times faster [10].

[1] www.cbl.ncsn.edu/benchmarks/layoutsynth92

Gordian [7] iterates between minimizing a quadratic objective function of wirelength and slicing the placement area. In each iteration, the bounds on the variables are changed, and more constraints are added to the formulation to account for the new regions resulting from the slicing process. The iterations are terminated once the size of a partition (region) become less than a prespecified threshold. The result of the global optimization and exhaustive partitioning is a *relative* placement in which the relative location of every cell with respect to every other cell is determined. The relative placement is unacceptable from the physical standpoint because of the overlap among the cells. Depending on the design style, overlap is eliminated by *legalizing* the relative placement. In case of gate arrays, standard cells and FPGA design style, cells are snapped to rows, and different techniques are used for other design styles. Following the *legalization* phase, further improvement of the initial placement is performed to account for incorrect enforcement of some cells to non-optimal locations during the computation of the relative placement and the legalization phases.

In our approach, the placement procedure (like Gordian) is broken down into relative placement phase followed by legalization and improvement phases. However, our formulation of the problem is different. We extended the traditional wirelength formulation by adding a repelling term such that upon minimization, a target distance is maintained between the locations of cells sharing common signals. We have also added attracting forces to pull cells from dense to sparse regions resulting in a uniform distribution of the cells on the placement area. Our formulations are convex and neither partitioning nor hard constraints are used. Moreover, they are versatile in the sense that they are applicable to a variety of problems where a target distance is desired between connected components.

2. Placement Problem and the Traditional Quadratic Measure

A circuit is represented by a hypergraph $G(V, E)$, where the vertex set $V = \{v_1, v_2, \cdots, v_n\}$ represent the nodes of the

0-7803-5832-X /99/$10.00 ©1999 IEEE.

hypergraph (set of cells to be placed), and $E = \{e_1, e_2, \cdots, e_m\}$ represents the set of edges of the hypergraph (set of nets connecting the cells). The two dimensional placement region is represented as an array of legal placement locations. The hypergraph is transformed into a graph (a hypergraph with all hyperedge sizes equal to 2) via clique model for each net. Each edge e_j is an unorder pair of vertices with a nonnegative weight w_j assigned to it. The placement task seeks to assign all cells of the circuit to legal locations such that cells do not overlap. Each cell i is assigned a location (x_i, y_i) on the XY-plane. The cost of an edge connecting two cells i and j with locations (x_i, y_i) and (x_j, y_j) is computed as the product of the squared l_2 norm of the difference vector $(x_i - x_j, y_i - y_j)$ and the weight of the connecting edge w_{ij}. The total cost, denoted $\phi(x, y)$, can then be given as the sum of the cost over all edges; i.e:

$$\phi(x, y) = \sum_{1 \leq i < j \leq N} w_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2] \quad (1)$$

As we pointed out previously, minimizing (1) produces a placement with a great amount of overlap among the cells because it attracts cells sharing common nets together [2].

3. New Formulation

3.1. Cell Repelling and Target Distance problem

In an endeavor to reduce overlap among cells through maintaining a target distance between the geometric locations of cells sharing same nets, we explored a whole class of functions that accomplish this aim. Let $\mathbf{v} = \{v_1, v_2, \cdots, v_n\}$ be n-dimensional vector and let z be the scaled square of the ℓ_2 norm of $\mathbf{v}$; i.e:

$$z = \beta \mid\mid \mathbf{v} \mid\mid_2^2$$

where β is constant. In [6], we proved that the function

$$\omega(z) = z + \rho(z)$$

is convex for $\rho(z) \in \pi$ where

$$\pi = \{-\mathbf{ln}(z) - 1, \mathbf{e^{(1-z)}} - 2, -\mathbf{1}\}, \quad z \in [1, \infty)$$

We also generalized the proof to the case $\mathbf{v} = \mathbf{u} - \mathbf{r}$, and $\mathbf{u}$ and $\mathbf{r}$ are also n-dimensional vectors. Specifically, we proved that $\omega(z)$ is quasiconvex provided that $\rho(z) \in \pi$, $\mid\mid \mathbf{u} \mid\mid_2^2 \in [1, \infty)$ and $\mid\mid \mathbf{r} \mid\mid_2^2 \in [1, \infty)$, and convex if either the elements of $\mathbf{u}$ or $\mathbf{r}$ are constant.

In the context of VLSI placement, let $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ be the position vectors of the geometric locations of cells i and j, and $\mathbf{s}_{ij} = p_i - p_j$ then based on what we proved in [6], the function

$$f(z) = \sum_{1 \leq i < j \leq N} \psi_{ij}(z) \quad (2)$$

$$\psi_{ij}(z) = \begin{cases} z + \rho(z) & z > 1 \\ 0 & \text{Otherwise} \end{cases} \quad (3)$$

$$z = \beta(w_{ij} \mid\mid \mathbf{s}_{ij} \mid\mid_2^2) \quad (4)$$

$$\mid\mid \mathbf{s}_{ij} \mid\mid_2^2 = (x_i - x_j)^2 + (y_i - y_j)^2$$

is convex if one cell k is fixed. Clearly, $f(z)$ is flat in the interval $z \in [0, 1]$. This implies that $f(z)$ has multiple solutions in this region. However, line search methods concludes the search when the first minimum is encountered. Thus, if the initial solution is outside the flat region, the search will be concluded when $z = 1$. Notice that z is the weighted squared distance between cell i and cell j, and w_{ij} is the connectivity weight between them. If we let

$$\beta = \frac{1}{d}$$

in equation (4) where again d is constant, then at convergence[3]; i.e, $z = 1$, the squared distance between the geometric locations of cell i and cell j is equal to a (*target*) estimate of d units [4]. In other words, cell i is spatially shifted from cell j location and vice versa. Thus, $f(z)$ is a repeller model. Note that if $\rho(z) = 0$ and $d = 1$, $f(z)$ reduces to formulation (1).

3.2. Cell Spreading

Minimizing model (2) ensures cell spreading if the target estimate d is set to high value with respect to the average cell width or cell height. But spreading the cells this way deteriorates the wirelength due to excessive stretching of short nets. Setting the estimate d to a lower value prevents excessive net-stretching, but yields poor utilization of the placement area (poor spreading of cells). To force cells to spread over the placement area while prohibiting excessive stretching of nets, new attracting forces are added. The basic idea is to displace a (free) cell located in a dense region to the closest sparse region with respect to its most recent geometric location. This is achieved via appending a connection (net) between that particular cell and a *dummy* fixed cell in the sparse region so that the free cell moves towards the dummy fixed cell in the subsequent iteration.

First, dense and sparse regions on the placement floor are identified as follows. For each cell i, an $\ell_w \times \ell_h$ rectangular window w_i centered at (x_i, y_i) is imposed on the placement floor and the total area A_a of all cells enclosed by w_i is computed. Next, region $\mathcal{R}_i$ surrounded by w_i is regarded sparse only if $A_a < A_w$ where $A_w = \ell_w \ell_h$. Cells that have been collapsed in a sparse region are not considered when new sparse regions are being identified. Each

[2] We refer to formulation (1) as the QP (Quadratic Programming) formulation.

[3] Geometrically, in general z is a circle of radius d, and at convergence it reduces to *unit* circle.

[4] We refer to formulation (2) as the Target Distance (TD) formulation.

sparse region $\mathcal{R}_i$ is then split into 4 quadrants and the center of the sparsest quadrant (in terms of number of cells that fall inside a quadrant) (x_a^i, y_a^i) become a center of a new cell attractor. A cell attractor is, merely, a *dummy* fixed cell located at (x_a^i, y_a^i).

Let $\mathcal{A} = \{(x_a^1, y_a^1), (x_a^2, y_a^2), \cdots, (x_a^q, y_a^q)\}$ be the set of locations of the q attractors. A two cell net is then established between cell i and the closest attractor. The overall objective function can now be given:

$$f(z) + g(x) + h(y) \tag{5}$$

$$l_x \leq x_i \leq u_x$$

$$l_y \leq y_i \leq u_y$$

$$g(x) = \sum_{1 \leq i \leq N} min\{(x_i - x_a^1)^2, \cdots, (x_i - x_a^q)^2\}$$

$$h(y) = \sum_{1 \leq i \leq N} min\{(y_i - y_a^1)^2, \cdots, (y_i - y_a^q)^2\}$$

Parameters l_x, l_y, u_x and u_y are lower and upper bounds on x and y. Notice that each cell i is assigned to an attractor with a coordinate (x_a^μ, y_a^ν) where μ and ν are the closest attractors to cell i in the x and y directions respectively. We refer to formulation (5) as the Attractor-Repeller (AR) formulation.

4. Basic Algorithm

Figure (1) illustrates the flow of the algorithm. Throughout the remaining parts of this paper, we refer to the new placement method as **ARP** (**A**ttractor **R**epeller **P**lacer). Following parsing circuit description, an initial placement is determined using the quadratic formulation of wirelength given in (1). In the subsequent iterations, the algorithm proceeds iteratively by solving the AR model. In each iteration, cells move to fill sparse regions and better spreading of the cells over the placement floor is achieved. In a subsequent iteration, current attractors are deleted and new ones are created; the AR-model is updated accordingly. This process continues until the algorithm reaches a point where no significant movement of the cells is attained. The algorithm stops when a maximum number of iterations $\mathcal{K}$ is exceeded, or if the ratio of the total area of sparse regions to that of the placement area is $< \kappa\%$. Experimentally, we found that $5 \leq \mathcal{K} \leq 10$ and $\kappa = 10\%$ are quite sufficient to uniformly spread the cells over the placement area and achieve good wirelength.

5. Results and Discussion

The proposed method is implemented in C language on Sun Ultra1/140 workstation. Total wirelength is measured as the sum of the half perimeter wirelength. Parameter d is computed as the $\sqrt{w_a}$ and w_a is the average cell width. We used

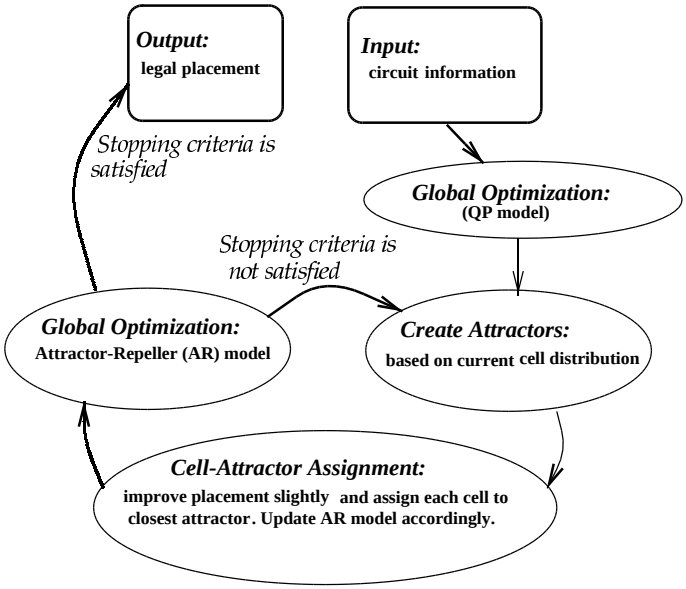

Figure 1: An outline of the placement procedure ARP.

a limited memory BFGS solution methodology, namely, L-BFGS-B [4] to minimize the ARP model. As a final placer, we used a Tabu-search based local improvement algorithm that had been developed in collaboration with other members in our group [2].

Circuit	Cells	Pads	Nets	Pins	Rows
Fract	125	24	147	462	6
Prim1	752	81	904	5526	16
Prim2	2907	107	3029	18407	28
Struct	1888	64	1920	5471	21
Ind1	2271	814	2478	8513	15
Bio	6417	97	5742	26947	46
Ind3	15059	374	21940	176584	54
Avq.s	21854	64	22124	82601	80
Avq.l	25114	64	25384	82751	86

Table 1: MCNC Benchmarks used as test cases

Table (1) depicts statistics of the MCNC benchmark circuits used to evaluate ARP. Circuit statistics include number of: cells, I/O pads, nets, pins and number of required rows. We compared our results to TimberWolf v6.0, TimberWolf v7.0 [10], and Gordian/Domino [7, 8] (numerical results for these approaches are taken from [10])[5].

Timberwolf v7.0 is the latest release of TimberWolf placer and it has two modes of operation. Flat mode in which the flat netlist (no clustering) is used. The second is the hi-

[5]For some benchmarks, numerical results have not been reported in [10] and their entries in the result tables are left empty.

erarchical mode in which the netlist is clustered into various levels of netlists[10]. Tables (2), (3), (4) list the final wirelength, longest row and computation time for ARP and the other approaches. ARP outperforms TimberWolf v6.0, TimberWolf v7.0 and Gordian/Domino on the majority of the benchmarks. For benchmark *industry3*, ARP outperforms TimberWolf v6.0 but lacks compared to TimberWolf v7.0 and Gordian/Domino; for benchmark *Bio* and *prim2*, ARP lacks insignificantly compared to TimberWolf v7.0 in flat mode. We suspect a different positioning of the I/O pads compared to the other methods, or a scaling problem is the reason for the deteriorated results. On average, ARP achieves 7.78%, 1.02%, 3.96% and 8.68% reduction in wirelength compared to TimberWolf v6.0 TimberWolf v7.0 (flat), TimberWolf v7.0 (Hierarchical) and Gordian/Domino.

Tables (3) and (6) show comparisons of longest row width obtained by ARP and the other approaches. Longest row width determines chip width and accordingly total chip area. Thus, longest row width is an important parameter in the assessment of a given placement. On average, ARP outperforms TimberWolf v6.0 and TimberWolf v7.0 by 4.28% and 0.13%. No longest row width was reported for Gordian to compare with.

Ckt	TW6	TW7.FM	TW7.H	Gord/Dom	ARP
Fract	-	-	-	-	0.034
Prim1	1.0	0.93	0.99	1.08	0.79
Prim2	3.71	3.53	3.72	4.02	3.61
Struct	-	-	-	-	0.34
Ind1	-	-	-	-	1.50
Bio	1.97	1.8	1.88	1.98	1.83
Ind3	48.38	43.08	44.67	44.94	48.12
Avq.s	6.72	6.45	6.13	6.42	6.06
Avq.l	6.93	6.50	6.81	7.16	6.54

Table 2: Wire Length Comparison, TW v7.0 flat and hierarchical modes, Gordain/Domino and ARP

Ckt	TW6	TW7.0	ARP
Fract	-	-	704
Prim1	5260	5100	5170
Prim2	8380	8210	8201
Struct	-	-	2360
Ind1	-	-	4810
Bio	5114	4936	4928
Ind3	28832	26368	26176
Avq.s	9560	9128	9080
Avq.l	9744	9400	9344

Table 3: Chip width comparison: TimberWolf v7.0 and ARP. Width is measured in *microns*.

Ckt	TW6	TW7.FM	TW7.H	Gor/Dom	ARP
Fract	-	-	-		12
Prim1	467	488	130	168	95
Prim2	3127	4307	736	542	504
Struct	-	-	-		116
Ind1	-	-	-	-	376
Bio	8606	12224	1273	1553	1290
Ind3	38619	70873	5156	6087	4253
Avq.s	54681	78248	7657	10261	8534
Avq.l	56802	97612	9175	12403	11202

Table 4: Run-time comparison in CPU seconds: TimberWolf v7.0 in flat and hierarchical modes, Gordian/Domino and ARP.

Ckt	TW v6 %impr.	TW7.FM %impr.	TW7.H %impr.	Gor/Dom %impr.
Prim1	+21	+15	+20.2	+26.8
Prim2	+2.7	-2.2	+2.9	10.2
Bio	+7.1	-1.64	+2.9	+7.5
Ind3	+0.53	-10.4	-7.2	-6.6
Avq.s	+9.8	+6.0	+1.1	+5.6
Avq.l	+5.6	-0.6	+3.9	+8.6
avg.	+7.78	+1.02	+3.96	+8.68

Table 5: Relative wirelength improvement with respect to other approaches (+ means ARP is better).

As for computation times, the results are illustrated in Tables (4) and (7) (computation times of the other approaches are scaled). Evidently, ARP consumes less computation times to place each test case compared to the other approaches. On average, ARP outperforms TimberWolf v6.0 by 83%, TimberWolf v7.0 (flat mode) by 87%, TimberWolf v7.0 (Hierarchical mode) by 7.6% and Gordian/Domino by 13.8%.

Ckt	TW v6.0 %impr.	TW v7.0 %impr.
Prim1	+1.7	-1.3
Prim2	+2.1	+0.11
Bio	+3.6	+0.16
Ind3	+9.2	+0.73
Avq.s	+5.0	+0.53
Avq.l	+4.1	+0.60
avg.	+4.28	+0.13

Table 6: Relative chip-width improvement with respect to other approaches (+ means ARP is better).

Ckt	TW v6 %impr.	TW7.FM %impr.	TW7.H %impr.	Gor/Dom %impr.
Prim1	+79	+80	+27	+3.8
Prim2	+83	+88	+31	+7.0
Bio	+85	+89	-1.3	+17
Ind3	+88	+93	+17	+30
Avq.s	+84	+89	-10	+16
Avq.l	+80	+88	-18	+9
avg.	+83	+87	+7.6	+13.8

Table 7: Relative CPU time improvement with respect to other approaches (+ means ARP is better).

6. Conclusion

The fact that future placement tasks are much more complicated implies that faster placement tools should be developed to handle such immense complexity. Future placement tools must be capable of placing larger circuits in a reasonable time, besides they must be flexible enough to handle any modifications in the design style of VLSI circuits.

In light of these facts, we proposed a new approach for performing placement tasks. Our approach adds new forces to the classical quadratic wirelength approach to avoid the need for partitioning the placement area or the necessity of hard constraints to utilize the placement area. We demonstrated the feasibility of the new approach and showed that it is comparable to other popular placers (despite the fact that the current implementation is preliminary and was intended to show the feasibility of the ideas).

7. References

[1] B. W. Kernighan A. Dunlop. A procedure for placement of standard-cell VLSI placement. *IEEE Trans. on CAD*, 4, no. 4:92–98, 1985.

[2] S. M. Areibi. *Towards Optimal Circuit Layout Using Advanced Search Techniques*. PhD thesis, University of Waterloo, Ont. Canada, 1995.

[3] D. Braun C. Sechen and A. Sangiovanni-Vincentelli. Thunderbird: A complete standard cell layout package. *IEEE journal of Solid State Circuits, 23(2)*, pages 410–420, April 1988.

[4] P. Lu C. Zhu, R. H. Byrd and J. Nocedal. L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. Technical report, Dept. of Elec. Eng. and Comp. Scien., NorthWestern University, 1996.

[5] G. Sigl K. Doll and F. Johannes. Analytical placement: A linear or quadratic objective function. In *In proc. 23rd DAC*, pages 57–62, 1991.

[6] Hussein Etawil and Anthony Vannelli. Novel convex models for VLSI analytic placement. Technical Report 5, Dept. of Elec. and Comp. Engg., University of Waterloo, April 1998.

[7] F. Johannes J. M. Kleinhans, G. Sigl and K. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE. Trans. on CAD, vol. 10, no. 3*, pages 356–365, 1991.

[8] K. Doll F. Johannes and K. Antreich. Iterative placement improvement by network flow methods. *IEEE. Trans. on CAD, vol. 13, no. 10*, pages 1189–1200, 1994.

[9] U. Lauther. A min-cut placement algorithm for general cell assemblies based on a graph representation. In *proc. of the 16th Design Automation Conference*, pages 1–10, 1979.

[10] Wern-Jieh Sun and Carl Sechen. Efficient and effective placement for very large circuits. In *in IEEE/ACM ICCAD*, pages 170–177, 1993.