A Performance-Driven Standard-Cell Placer Based on a Modified Force-Directed Algorithm*

Yih-Chih Chou
Department of Computer Science
National Tsing Hua University
Hsin-Chu 30043, Taiwan, R.O.C.
dr834329@cs.nthu.edu.tw

Youn-Long Lin
Department of Computer Science
National Tsing Hua University
Hsin-Chu 30043, Taiwan, R.O.C.
ylin@cs.nthu.edu.tw

ABSTRACT

We propose a performance-driven cell placement method based on a modified force-directed approach. A pseudo net is added to link the source and sink flip-flops of every critical path to enforce their closeness. Given user-specified I/O pad locations at the chip boundaries and starting with all core cells in the chip center, we iteratively move a cell to its force-balanced location assuming all other cells are fixed. The process stops when no cell can be moved farther than a threshold distance. Next, cell rows are adjusted one at a time starting from the top and bottom. After forming these two rows (top/bottom), all movable core cells' force-balanced locations are updated. The row-formation-and-update process continues until all rows are adjusted and, hence, a legal placement is obtained. We have integrated the proposed approach into an industrial APR flow. Experimental results on benchmark circuits up to 191K-cell (500K-gate) show that the critical path delay can be improved by as much as 11.5%. We also study the effect on both layout quality and CPU time consumption due to the amount of pseudo net added. We found that the introduction of pseudo net indeed significantly improves the layout quality.

Keywords

Placement, Force-directed, Timing Closure.

1. INTRODUCTION

Placement is one of the most important steps in physical design of integrated circuits. It significantly affects layout area, routability and performance, among others. Standard-cell-based layout style is very popular for semi-custom layout design because it enables the success of both synthesis methodology and automatic placement and route (APR) flow. In the past, many approaches have been proposed for the problem. Notable approaches include partitioning-based, quadratic programming, iterative refinement by simulated annealing or force-directed analogy. In the very deep submicron era, large chip complexity, multiple metal layer, and wire delay dominance together call for re-investigation of the current placement solutions.

Iterative refinement is a popular approach to placement quality improvement. Given an initial placement obtained by any constructive methods, it repetitively modifies the locations of a small set of cells. A force-directed cost function is natural for the choice of the set. It is intuitive that a cell should be moved to where the sum of all attractive forces between it and all its adjacent cells (fan-in and fan-out) is minimized. Although many force-directed placement algorithms [1][2][3] have been proposed for iterative refinement, they have some drawbacks. First, the quality of the initial placement may affect the quality of the final placement result. Second, if the zero-force location of a cell is occupied by another cell, one needs to displace one or more cells and may get a worse timing result [4][5]. Third, path delay cannot be dealt with directly [6][7][8]. Instead, nets are weighted based on some distributed timing slack.

We propose to reduce a path's delay by pulling its starting and ending points closer. This is achieved by adding a pseudo net with path-length-dependent force between the source and sink flip-flops (or PI or PO). We also propose to obtain globally force-balanced cell positions by resolving cell overlapping later. Our algorithm consists of three steps: (1) graph construction (2) force-equilibrium cell positioning, and (3) cell row formation. First, we construct a graph to represent the circuit netlist to be placed. The second step finds a force-equilibrium position for every core cell. All core cells are initially placed at the chip center while all I/O pad cells are fixed around the chip boundary. We iteratively move one core cell at a time to its force-equilibrium position assuming all other cells are fixed. Overlapping in the horizontal direction is allowed during this step. In the third step, we form cell rows one at a time. When a row is formed, no overlapping is allowed among its constituent cells. The positions of the remaining cells are updated using the same algorithm of Step II. We repeat the row-formation-and-update process until all rows are formed and, hence, a legal placement is obtained.

Experimental results on large benchmark circuits implemented in a TSMC 0.18μm CMOS process have demonstrated the effectiveness and efficiency of the proposed approach. We achieve about 11.5% improvement over a state-of-art commercial tool in the critical path delay at the expense of about 21% more CPU time.

The rest of this paper is organized as following. In Section 2, we define some terminologies and formulate the problem. In

* Supported in part by a grant from the National Science Council of R.O.C. under contract no. 89-2218-E-007-069.

Section 3, we propose our approach. Our experimental setup and results are presented in Section 4. Finally, in Section 5 we draw conclusions and point to possible directions for future research.

2. PROBLEM FORMULATION

A circuit netlist consists of a set of m core cells **C** = $\{c_1, c_2, \ldots, c_m\}$ and a set of n I/O pad cells **P** = $\{p_1, p_2, \ldots, p_n\}$. Each cell $c_i \in$ **C** has variable width w_i and constant height H. The I/O pad locations are fixed around the chip boundary according to user specification while the positions of core cells are to be determined. All core cells and I/O pad cells are interconnected by a set of k nets **N** = $\{n_1, n_2, \ldots, n_k\}$. The layout plane is depicted in Figure 1. The I/O pad cells are fixed around the boundary according to user specification. The core area bounded within I/O pad cells is divided into cell rows of equal height H by a floorplanner. Unlike traditional 2 or 3-metal technology, modern process technology requires no routing channel between cell rows. A placement is the assignment of core cells to cell rows and location within the row such that no cell overlaps with one another. A timing-driven placement is a placement in which the critical path delay meets a user-specified target.

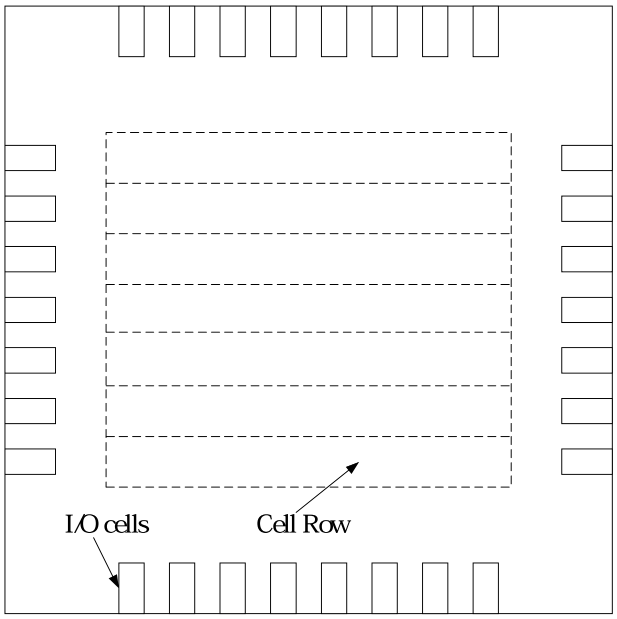


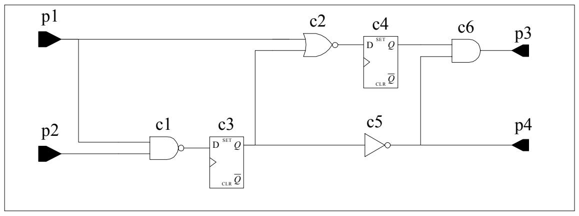
Figure 1: A typical layout plane

We define a force pulling a pair of cells towards each other if there is a link between their corresponding nodes in the representative graph [9]. Let the cells be c_i at location (x_i, y_i) and c_j at (x_j, y_j). We define the force as

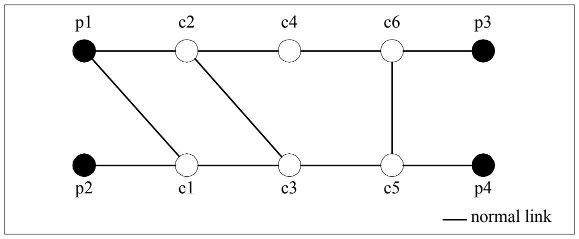
$$f(c_i, c_j) = \begin{cases} (x_i - x_j)^2 + (y_i - y_j)^2 & \textit{normal link} \\ ((x_i - x_j)^2 + (y_i - y_j)^2)(length(path_{i \to j}))^{\alpha} & \textit{pseudo link} \end{cases}$$

where α is a user-specified parameter, a *normal link* exists between a driver and a load cell, and a *pseudo link* exists between the source flip-flop (or primary input) and the sink flip-flop (or primary output) of a signal path. The *length* represents for the cell count of the signal path from c_i to c_j. Since we want to explore the relationship between the normal link and the pseudo link, we add a polynomial weight factor $(length(path_{i \to j}))^{\alpha}$ on the pseudo link and test different α values to find the best form. Our experimental result shows that the best α value is between 2 and 2.5. Because the force associated with a pseudo link is much stronger than that with a normal link, the path length is properly taken into account during cell movement. In this model, a long path will have closer end points. Therefore, its constituent wires need not to stretch too much.

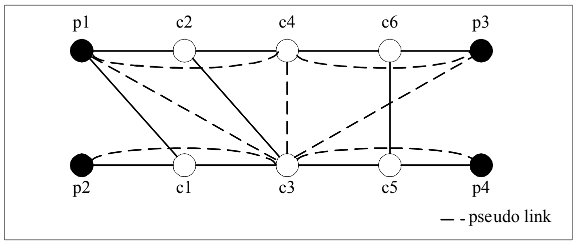
Let's illustrate the graph model with an example. Figure 2(a) shows a sample circuit netlist. In Figure 2(b), we model the circuit connectivity as a graph where cells are modeled as nodes and nets are modeled as edges (normal links). In addition to these edges, we introduce new edges (pseudo links) for all signal paths. That is, if there exists a path between a flip-flop (or primary input), c_{from} (p_{from}), and another flip-flop (or primary output), c_{to} (p_{to}), we add a pseudo link connecting c_{from} (p_{from}) and c_{to} (p_{to}). For example, in Figure 2(c), edge (*c3*, *c4*) is the pseudo link between flip-flops *c3* and *c4*.



(a) a sample netlist



(b) the graph model without path information



(c) the full graph model

Figure 2: Graph representation of a netlist

3. PROPOSED APPROACH

Our proposed method consists of three phases: (1) graph construction (2) force-equilibrium cell positioning, and (3) cell row formation. In the first phase, we construct a graph to represent the circuit netlist to be placed. The graph is denoted as $G = \{P \cup C, E \cup F\}$, where P is the set of I/O pad cells, C is the set of core cells, E is the set of edges each represents a wire connection between a driver and a load cell, and F is the set of pseudo edges each links the source flip-flop (or primary input) and the sink flip-flop (or primary output) of a signal path. We call an edge in E a *normal link* and an edge in F a *pseudo link.*

In second phase, we gradually move core cells to their force-equilibrium positions. The vertical location of a core cell must align with cell rows while its horizontal location within a row is unrestricted and overlapping with other cells is allowed during this phase.

At the beginning, all I/O pad cells are fixed and all core cells are located at the chip center. We iteratively move one core cell at a time to its force-equilibrium location until no more cells can be moved.

The pseudo code of our algorithm is given in Figure 3. We use a queue data structure to aid this process. Initially, all core cells adjacent to any I/O pad cells are queued. As long as the queue is not empty, we take one core cell from the queue and calculate its force-equilibrium position assuming that all of its adjacent cells are fixed at their present locations. If the distance between its current and calculated positions is greater than a threshold value (here we use the cell height $\boldsymbol{H}$), we move the cell to the calculated position and queue all of its movable adjacent core cells if they are not already there.

On the other hand, if the distance is smaller than the threshold, we just keep the cell location unchanged. A cell may be moved again after one or more of its adjacent cells are moved. This process continues itself until the queue becomes empty. That is, all cells are in a force-equilibrium position.

In the third phase, we form cell rows one at a time. After the second phase, cells are positioned onto rows. Because we allow cell overlapping in the horizontal direction, some rows may have more cells than their capacity allows while some other rows may have a lot of empty space. Starting with the top-most and bottom-most rows, we fill cell row with appropriate amount of cells one row at a time. For ECO operation, layout area is usually not 100% utilized. User-specified empty space should be uniformly distributed among all cell rows. For the row currently under formation, if it is under-utilized, we take some cells from its adjacent unformatted row. On the other hand, if it is over-utilized, we move some cells to its adjacent row. The selection of cells is again force-directed. After that, the selected cells are fixed within the row such that empty space is uniformly distributed and aligned to the pitch. After a row is formed, all cells in the remaining unformatted rows are adjusted to their force-equilibrium locations using the same procedure as Phase II. This process continues itself until all rows are formed. By now, we obtain a feasible placement.

Figure 4 depicts some snapshots by applying the procedure on the example circuit of Figure 2. Figure 4(a) is the initial placement where four I/O pad cells are fixed at four boundaries respectively while six core cells, *c1*, *c2*, … *c6*, are located at the chip center. Figure 4(b) shows the result after the first core cell *c2* is moved. Figure 4(c) shows the force-equilibrium placement where some overlapping exists as shown in gray. In Figure 4(d), we form the outermost cell rows *R0* and *R2*. Because *R0* is too sparse, a cell *c2* is taken from row *R1*. Then we update the remaining cells to new force-equilibrium as depicted in Figure 4(e). Finally, in Figure 4(f), the last cell row *R1* is formed and the placement is completed.

Algorithm FDP

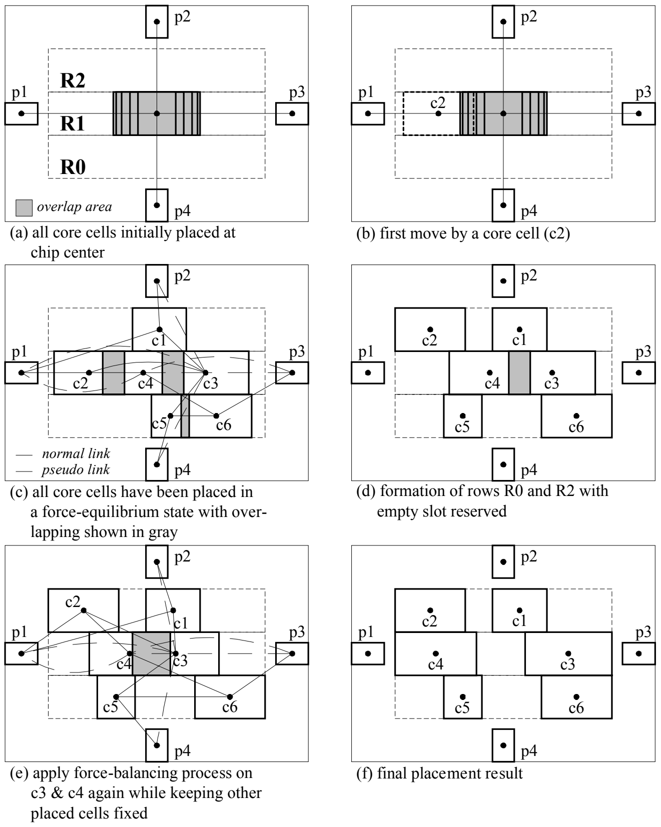
```
begin
  for each path chosen according to parameter $\beta$
    add a pseudo link between source and sink I/O or flip-flop;
  end for
  for each I/O pad cell $p$
    fix $p$'s location according to user specification;
  end for
  force_balance;
  repeat
    place the topmost unplaced row;
    place the bottommost unplaced row;
    force_balance;
  until all rows are placed
end

procedure force_balance
begin
  initialize a queue $Q$;
  for each movable core cell $c$ adjacent to either an I/O pad cell
  or a fixed core cell
    enqueue($Q$, $c$);
  end for
  while ($Q \neq \phi$)
    $c$ = dequeue($Q$);
    $\delta$ = distance from $c$'s current location to its force-balanced location;
    if ($\delta > \boldsymbol{H}$) then /* $\boldsymbol{H}$ is the cell height */
      move $c$ to its force-balanced location;
      for each movable and unqueued core cell $c'$ adjacent to $c$
        enqueue($Q$, $c'$);
      end for
    end if
  end while
end procedure
```

Figure 3: The proposed force-directed placement (FDP) algorithm

4. EXPERIMENTS

We have implemented the proposed algorithm in a C++ program running on a SUN UltraSparc 80 workstation. The experiment setup is illustrated in Figure 5. First, we read the RTL benchmark and synthesize it using Synopsys's Design Analyzer. Second, we import the synthesized netlist to Cadence's Silicon Ensemble Ultra (SEU 5.2) for floorplanning that defines I/O pad locations and the number of cell rows. Then we export a DEF [12] file as the input to our program. Output of our program is also in DEF format. It contains our placement result for Silicon Ensemble



p1
p2
p3
p4
R2
R1
R0
overlap area
c1
c2
c3
c4
c5
c6
normal link
pseudo link
(a) all core cells initially placed at chip center
(b) first move by a core cell (c2)
(c) all core cells have been placed in a force-equilibrium state with overlapping shown in gray
(d) formation of rows R0 and R2 with empty slot reserved
(e) apply force-balancing process on c3 & c4 again while keeping other placed cells fixed
(f) final placement result

Figure 4: Some snapshots of the proposed placement process

Ultra to perform its routing step. Finally, we extract parasitic RC values using Cadence's HyperExtract and back-annotate the cell/net delay information calculated by Cadence's SEU in the SDF [10] format to Design Analyzer for critical path timing analysis using Synopsys's DesignTime within Design Analyzer.

We also run the placement step using a commercial tool (CT) as well as its placement-based optimization flow (CT+PBO) for comparison.

We use seven benchmark circuits to evaluate the proposed approach. Their characteristics are summarized in Table 1. The cell counts range from 3K to 191K in a TSMC 0.18μm CMOS cell library from Artisan [11]. The benchmark set is available to the public at *http://www.cs.nthu.edu.tw/~ylin/placement.*

We study the effect of two parameters on the placement quality. Recall that we model the force between two flip-flops as $f(i,j) = ((x_i - x_j)^2 + (y_i - y_j)^2)(length(path_{i \to j}))^{\alpha}$. We evaluate five values of α: 1, 1.5, 2, 2.5 and 3. The second parameter is the threshold value on the length of a path for which a pseudo net should be added between its starting and ending flip-flops or primary I/O. We try four cases: β = 0%, 10%, 50% and 100 %. With β = 0%, we add no pseudo link; on the other hand, with β = 100%, we add pseudo link for every path. For β = 10%(50%), we add pseudo link for those paths whose length is larger than 90%(50%) of the critical path. We obtain the length of the critical path by the synthesis tools (Synopsys).

Table 1. Benchmark Characteristics

Benchmark	# cells	# nets	# I/O	Area(μm^2)
matrix	3375	3603	119	227405
sdram_rdr	4125	4559	95	365698
32bMAC	8655	8941	213	562695
VP2	10063	10542	323	657251
64bMAC	27043	27458	417	1210814
a259K	95765	104683	153	4392336
a518K	191592	209354	153	8230874

Table 2 compares our results with that of the commercial tool (CT) and that of CT with Placement-Based Optimization (CT+PBO). Here we let α = 2 and β = 100%. Compared with CT, our result is 11.5% better in terms of critical path delay at the expense of about 21% more CPU time. When CT's result is further optimized (buffer insertion, sizing etc) with its PBO option, our approach is still 4% better even without those optimizations. The experimental also unveils that the timing improvement is more significant as the circuit size gets larger.

Table 2. Comparison between FDP, CT and CT+PBO

Benchmark	CT		CT+PBO		FDP	
	Delay(ns)	CPU(s)	Delay	CPU	Delay	CPU
matrix	8.42	6	0.935	1.66	0.938	1.51
sdram_rdr	2.81	35	0.966	3.31	0.949	1.73
32bMAC	4.85	154	0.907	3.75	0.863	1.11
VP2	13.66	276	0.942	2.40	0.895	1.06
64bMAC	4.97	509	0.915	3.54	0.870	1.15
a259k	12.35	13196	0.887	2.51	0.836	1.01
a518k	14.26	46313	0.913	2.39	0.843	0.93
average			0.924	2.79	0.885	1.21

* α = 2, β = 100%; FDP's and CT+PBO's numbers are relative to CT's

Table 3 shows the impact on placement quality due to parameter α. The large the α value is, the more emphasis we place on the relative weight of pseudo link. Empirically, α = 2 is a good choice.

Table 4 shows the impact on both placement quality and run time due to parameter β. We expect that a high β value will lead to high placement quality at the expense of more CPU time consumption. The experimental results confirm our expectation. In fact, without the addition of pseudo links (i.e., β = 0%), our approach will not be able to compete with the commercial tool.

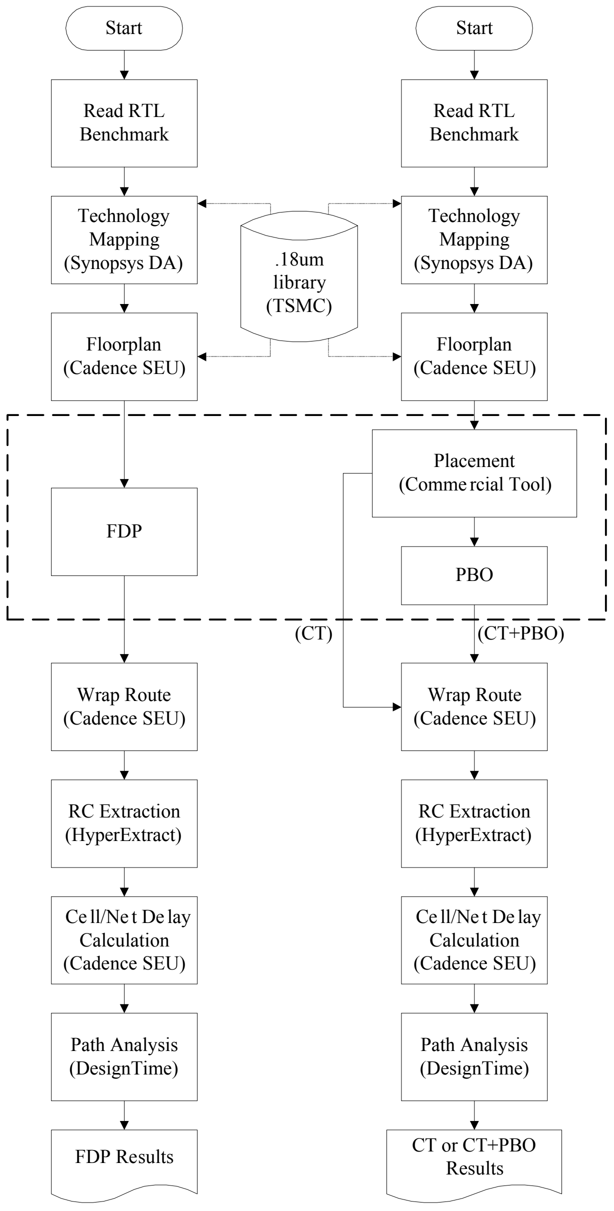

Start
Read RTL Benchmark
Technology Mapping (Synopsys DA)
.18um library (TSMC)
Floorplan (Cadence SEU)
FDP
Placement (Commercial Tool)
PBO
(CT)
(CT+PBO)
Wrap Route (Cadence SEU)
RC Extraction (HyperExtract)
Cell/Net Delay Calculation (Cadence SEU)
Path Analysis (DesignTime)
FDP Results
CT or CT+PBO Results

Figure 5: Experiment flow

5. CONCLUSIONS AND FUTURE WORK

We have proposed a performance-driven standard-cell placement method based on a modified force-directed approach. In this approach, we take the path delay into consideration by introducing a new type of edge in our graph model. Our experimental results confirm that the pseudo link indeed significantly contribute to the timing improvement. Experimental results also demonstrate the efficiency and effectiveness of our algorithm. It took about 12 hours to place a half-million-gate design.

In the future, we would like to extend this work by incorporating ECO capability such as buffer insertion. We would also extend it to handle large macros or preplaced blocks. Another interested point is the relationship between initial placement and the layout quality and time consumption.

6. REFERENCES

[1] N. R. Quinn, ''The placement problem as viewed from the physics of classical mechanics,'' in *Proc. of the12th Design Automation Conference*, pp. 173-178, 1975.

[2] K. J. Antreich, F. M. Johannes, and F. H. Kirsch, ''A new approach for solving the placement problem using force models,'' in *Proc. of the IEEE International Symposium on Circuits and Systems*, pp. 481-486, 1982.

[3] Shahookar, K. and P. Mazumder, ''VLSI cell placement techniques,'' in *ACM Computing Surveys*, 23(2), pp. 143-220, June 1991.

[4] Sadiq M. Sait and Habib Youssef, ''VLSI Physical Design Automation – Theory and Practice,'' IEEE PRESS, pp. 176-181, 1995.

[5] Naveed A. Sherwani, ''Algorithms for VLSI Physical Design Automation,'' 3rd Edition, Kluwer Academic Publishers, pp. 232-233, 1999.

[6] Too-Seng Tia and C. L. Liu, ''A New Performance Driven Macro-Cell Placement Algorithm,'' in *Proc. of EURO-DAC'93*, pp. 66-71, 1993

[7] Maogang Wang, Xiaojian Yang and Majid Sarrafzadeh, ''Dragon2000: Fast Standard-Cell Placement for Large Circuits,'' in *Proc. Int. Conf. Computer-Aided Design*, 6A.2, 2000.

[8] Fan Mo, Abdallah Tabbara, and Robert K. Brayton, ''A Force-Directed Macro-Cell Placer,'' in *Proc. Int. Conf. Computer-Aided Design*, 4A.3, 2000.

[9] Hans Eisenmann and Frank M. Johannes, ''Generic Global Placement and Floorplanning,'' in *Proc. of the 35th Design Automation Conference*, pp. 269-274, 1998.

[10] Pran Kurup and Taher Abbasi, ''Logic Synthesis Using Synopsys,'' Second Edition, Kluwer Academic Publishers, 1998.

[11] Artisan Components Inc., http://artisan.com.

[12] Cadence Design Systems, Inc., ''LEF/DEF Language Reference,'' Product Version 5.0, February 1997.

Table 3. ***α*****'s impact on placement quality**

Benchmark	*α* = 1		*α* = 1.5		*α* = 2		*α* = 2.5		*α* = 3	
	Delay(ns)	CPU(s)	Delay(ns)	CPU(s)	Delay(ns)	CPU(s)	Delay(ns)	CPU(s)	Delay(ns)	CPU(s)
matrix	**7.80**	8	7.79	9	7.86	9	7.83	9	7.90	11
sdram_rdr	**2.61**	60	2.69	58	2.67	61	2.71	56	2.92	63
32bMAC	4.29	176	4.21	183	**4.19**	171	4.30	175	4.35	191
VP2	12.50	319	12.39	325	12.23	293	**12.10**	299	12.56	358
64bMAC	4.53	609	4.38	627	4.32	585	**4.29**	607	4.69	761
a259k	11.27	13408	11.08	14019	**10.32**	13328	10.36	13906	11.89	15123
a518k	12.51	43941	12.50	45032	**12.02**	43071	12.16	44613	13.07	46930

* *β* = 100%

Table 4. ***β*****'s impact on placement quality**

Benchmark	*β* = 100 %		*β* = 50%		*β* = 10%		*β* = 0%	
	Delay(ns)	CPU(s)	Delay	CPU	Delay	CPU	Delay	CPU
matrix	7.86	9	1.05	0.86	1.10	0.81	1.09	0.79
sdram_rdr	2.67	61	1.06	0.82	1.13	0.85	1.16	0.82
32bMAC	4.19	171	1.04	0.81	1.09	0.63	1.19	0.58
VP2	12.23	293	1.01	0.90	1.06	0.75	1.08	0.71
64bMAC	4.32	585	1.04	0.80	1.11	0.65	1.18	0.56
a259k	10.32	13328	1.03	0.83	1.12	0.58	1.16	0.49
a518k	12.02	43071	1.03	0.79	1.11	0.56	1.21	0.41
average			1.04	0.83	1.10	0.69	1.15	0.62

* *α* = 2; the numbers for *β* = 50%, 10% and 0% are relative to that of *β* = 100%