Efficient and Effective Placement for Very Large Circuits

Wern-Jieh Sun and Carl Sechen

Abstract—**We present a new approach to simulated annealing and a new hierarchical algorithm for row-based placement which has obtained the best results ever reported for a large set of MCNC benchmark circuits. Our results indicate that chip area reductions up to 15% are achieved compared with TimberWolfSC v6.0 [13], [14], [17]. Our new hierarchical annealing-based placement algorithm (TimberWolfSC v7.0) yields chip area reductions up to 21% while consuming up to 7.5 times less CPU time in comparison to TimberWolfSC v6.0. Furthermore, TimberWolfSC v7.0 produces lower total wire length by an average of 8% than Gordian/Domino [2], [3], [8], [12], 11% lower wire length than Ritual/Tiger [15], while using comparable run time. TimberWolfSC v7.0 also supports precise timing driven placement [16].**

I. Introduction

CHIP area and performance are the two critical requirements for circuit placement today. Algorithms which are timing driven but yield relatively poor chip areas are of little interest, even if they are efficient in terms of computation time. Algorithms which yield near minimum chip areas but which are not precisely timing driven are similarly of little interest. A state-of-the-art placement method must not only yield minimum chip areas but it also must be precisely timing driven. Furthermore, it must be efficient. Current industrial row-based placement problems, and even an MCNC benchmark circuit, contain as many as 100 000 standard cells.

Up to now, based on benchmark results reported to MCNC, results presented at the 1992 International Workshop on Layout Synthesis, and as reported in [2], there are two effective placement methods. One of these is Gordian/Domino [2], [3], [8], [12], based on recursive partitioning and quadratic programming. While this method may be efficient enough to handle a 100 000 cell circuit, no timing driven results were presented. The other effective placement method is based on simulated annealing. This implementation, called TimberWolfSC v6.0 [13], [14], [17], yielded placement results close in quality to those reported in [2]. On the negative side, it is around 5 times slower than Gordian/Domino. But, on the positive side, precise timing driven placement is efficiently provided [16]. Hence this method would be a good candidate for the state-of-the-art placement approach if its overall efficiency could be significantly improved. Another placement method, called Ritual/Tiger, also based on quadratic programming, claims to be applicable to large placement problems [15]. However, no benchmark results have ever been submitted by its developers. Furthermore, while it is timing driven, when in the timing driven mode, its time complexity is $O(n^2)$, where n is the number of cells, which makes it impractical for very large circuits. This method also can not handle cells of varying widths.

In this paper we show that an improved approach to simulated annealing [7] is not only faster than Gordian/Domino but also yields better placement results, in addition to supporting three different modes of timing driven placement [16]. In the first part of the paper we present a simplified objective function which is based on the concept that every generated placement configuration is a valid placement, that is, the cells are not permitted to overlap during the annealing process. Consequently, wire length and timing calculations are more accurate, uniform row lengths are achieved, and the annealing schedule can be shortened. Generating new configurations without overlapping usually requires that the net lengths must be updated for up to half the cells in each row involved in the cell exchange. This problem was addressed in [5]. However, in [5], the changes in the wire lengths of the nets connected to the cells which were shifted to accommodate the cell exchange were not evaluated. This results in unacceptable error when evaluating the quality of the proposed cell exchange. When evaluating an exchange, the method computes precisely the new lengths of the nets connected to the two cells involved in the exchange. However, when computing the total change in wire length, they take the difference between the precise new value and the previously stored value for these nets. Since the previously stored values are arbitrarily incorrect due to unaccounted for shifting of cells caused by previous cell exchanges, the net change in wire length presented to the acceptance criterion in simulated annealing is generally substantially incorrect. Consequently, the quality of the results yielded by this approach were inferior to those reported in [14]. Furthermore, it is not possible to perform precise timing driven placement with this method, since net lengths may increase arbitrarily when they are not monitored (e.g., when these nets are connected to cells being shifted).

We have developed a fast technique which precisely updates the net lengths for the two cells involved in an exchange and which estimates effectively the net lengths for the other cells in each affected row. We present two estimation models and compare the placement results achieved by using our new models versus using a very slow, exact updating scheme (which updates precisely all the nets connected to the cells

Manuscript received January 27, 1994; revised June 8, 1994 and October 19, 1994. This paper was recommended by Associate Editor T. Yoshimura

The authors are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA.

IEEE Log Number 9408617.

involved in an exchange including any shifted cells). Experiments show our models yield comparable or better results. Our results indicate that chip area reductions up to 15% are achieved compared with TimberWolfSC v6.0.

In the second part of the paper we present a hierarchical placement algorithm. There are two stages, clustering and placement. In the clustering stage, the input network is condensed and hierarchically clustered into various levels of netlists. The run time of our new clustering algorithm is less than 5% of the hierarchical placement time. After we have generated hierarchical netlists, the placement stage then uses the new approach to simulated annealing presented in the first part of the paper to place the condensed, clustered network. In the current implementation, the original circuit is clustered into three levels of hierarchy. We first place the top level netlist in the high temperature regime. After the placement for the top level netlist is done, placement for the second level netlist then takes place in the middle temperature regime. The constituent cells belonging to a cluster are randomly placed within the confines of the location of the bounding rectangle of the cluster as determined in the previous placement level. Finally, placement for the bottom level netlist takes place in the low temperature regime. Since the number of clusters and inter-cluster nets are both substantially reduced in the higher levels of hierarchy, the computation time for the hierarchical placement approach is about an order of magnitude less than what it would be for the original flat network.

There is another advantage of combining a clustering technique with simulated annealing. Namely, the simulated annealing algorithm is known to behave in a pseudo top-down manner. In the high temperature regime, it behaves more like a random method and cells move freely around the core region. In the low temperature regime, it behaves more like a greedy algorithm and only local moves are accepted. Therefore the combination of clustering and simulated annealing effectively can be viewed as a combined bottom-up and top-down approach. Our new hierarchical annealing-based placement program yields chip areas up to 21% less while consuming up to 7.5 *times* less CPU time in comparison to TimberWolfSC v6.0.

Clustering usually serves as a bottom-up preprocessing stage in a hierarchical placement environment. Clustering cells together greatly reduces the complexity of the circuits. A good clustering method should identify groups of cells which will eventually end up together in the final placement. This can be difficult because clustering decisions are made prior to the start of the hierarchical placement process, and hence these decisions are made without a global view of the circuit structure, timing information, or floorplan. Because of this difficulty, a top-down global netlist partitioning methodology has usually been preferred to a bottom-up clustering methodology. But the sizes of today's circuits are so large that a top-down partitioning scheme is infeasible. Therefore, an effective bottom-up clustering approach is a necessity as a pre-processing stage in a hierarchical placement approach. However, for clustering to be used as a practical bottom-up approach, there are two important concerns. 1) The computation time used to generated the clusters must be negligible in comparison to the time needed to place the condensed network. 2) The sizes of the generated clusters should vary over as small a range as possible. Our new objective function, described in the first part of the paper, is precisely accurate when the clusters have exactly the same size. The accuracy, while still good for variances on the order of those for flat standard cell netlists, degrades as the variance in cluster sizes increases. In general, experience has shown that the effectiveness of interchange algorithms degrades significantly if the sizes of the objects vary by orders of magnitude. Furthermore, in our new approach, moves are prevented which would create an unacceptable amount of unevenness in the rows. If clusters had widely varying widths, it would be very difficult to find feasible moves for the very wide clusters and this would prevent the annealing algorithm from finding a good local optimum.

Several clustering algorithms have been reported. These approaches are based on prioritized attributes [11], circuit partitioning [19], random walk [1], [6], and graph connectivity [4]. In the prioritized method [11], the modules are merged according to an attribute list. The attributes include terminal count, common net count, the number of nets localized, common net fan-out, and the cluster size. A disadvantage of the prioritized list is that it differs from circuit to circuit. In the ratio cut method [19], the number of clusters generated is unknown until the algorithm finishes. Moreover, there are generally few resulting clusters, and these clusters vary widely in size. Its time complexity is more than quadratic which makes it impractical for today's large circuits. In the random walk method [6], the length of the random sequence used to find cycles is $O(n^2)$, where n is the number of cells in the input. The total time complexity is $O(n^3)$. This long run time makes it inappropriate to serve as a bottom-up preprocessing stage in a hierarchical placement environment. In the (k,l)-connectivity method [4], the use of $l > 1$ often leads to unnatural results. Cells which are farther away from each other (in terms of the number of intervening nets) are more likely to be placed in the same cluster than cells which are closer together. Also, the selection of k and l is not an easy task. They are usually selected based on experiments [4].

The remainder of this paper is organized as follows. Section II presents our new approach to simulated annealing for row-based placement. In Section III, we present our new hierarchical placement and clustering technique. The results are presented in Section IV. Finally, the conclusion is the subject of Section V.

II. New Approach to Simulated Annealing

A. The Drawbacks of Allowing Cell Overlapping

Allowing cell overlapping enables the simplest implementation of simulated annealing for row-based placement. It also yields the fastest implementation in terms of computation time. When cell overlapping is permitted, the only nets whose lengths change are those belonging to the two cells involved in an interchange. In contrast, if overlapping is not permitted, not only must one update the net lengths for the nets on the two

cells being interchanged, but also the nets for up to half the cells in each row involved in the cell exchange (see Fig. 2).

The cost function for the simulated annealing based TimberWolfSC v6.0 [13], [14], [17] is shown in (1). W is the total wire length, P_o is the overlap penalty, and P_R is the row length control penalty.

$$C = W + \mu P_o + \lambda P_R \tag{1}$$

The role of the two penalty functions is to ensure that when the cost C has been minimized, the total amount of cell overlapping has been nearly reduced to zero and the row lengths are fairly uniform. When two cells with different size are exchanged, an overlap and an unused space will be created at their original positions. Permitting overlapping causes inaccurate cell positions, inaccurate net length computations, and inaccurate timing computations. The two controlling parameters μ and λ in (1) are crucial. If μ is too small, the cells will tend to collapse to the centers of the rows. However, if μ is too large, for the most part only cells with equal size will be exchanged. This will greatly impede the algorithm's search for a good local optimum. Similarly, if λ is too small, the final row lengths will be very uneven. If λ is too large, the algorithm's search will again be impeded.

To combat this problem, the method in [14] uses a sophisticated negative feedback control scheme to determine the optimum values of μ and λ. Nonetheless, at the end of the annealing process, there is a significant amount of residual cell overlaps and gaps. As a final step, the cells are shifted and compacted to eliminate the overlaps and gaps. This final shifting can greatly disturb wire lengths and often results in timing paths failing to meet their specifications. For large circuits, on average, our experiments have shown that the longest row is about 12% above the average row length. This increases the chip area, since the width of the core is determined by the longest row.

B. New State Generator

Fig. 1 shows our algorithm for generating new placement configurations. First, we randomly select cell a. A single cell move is attempted if the target row's length limit is not exceeded. Otherwise, a cell b which covers the target location is noted and an interchange of a and b is attempted if no row length limits are violated. The length limit is set to be the smaller of one percent of the average row length or one average standard cell width. If a limit was violated, the new state generator begins anew. If a single cell or interchange move is feasible, then the change in cost is computed in the manner presented in the next section. The probability of accepting the new configuration is one if $\Delta C \leq 0$; otherwise, it is equal to $e^{(-\Delta C/T)}$, where T is the temperature [7].

If the new configuration is accepted, the cells in the affected rows are shifted to avoid any cell overlapping. Every new configuration thus generated is legal and physically feasible. Therefore, at the end of run, there is no need for shifting and compacting cells. In Fig. 2, when cell A and B are exchanged, the cells to the left of B (cells in the shaded region) will be shifted to the left to ensure no overlapping. Cells on the

1. Randomly select cell *a*
2. Randomly select row *r* and location *x* in *r*
3. /* *x* in *r* is within the range limiter window span for *a* [14] */
4. **If** (adding *a* to *r* doesn't exceed length limit for *r*) **then**
5. Compute ΔC for moving *a* to location *x* in *r*
6. **Else**
7. /* now consider an exchange of *a* and *b* */
8. Note cell *b* covering location *x* in *r*
9. **If** (length limit of neither row is exceeded) **then**
10. Compute ΔC for exchanging *a* and *b*
11. **Else**
12. Go to line 1
13. **If** (*accept*(ΔC)) **then**
14. Eliminate overlaps for the row(s) of *a* and *b*
15. Update estimation model for *a* and *b*

Fig. 1. New state generator.

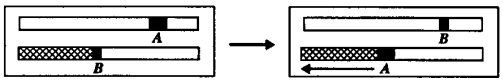


Fig. 2. Shifting operation.

short side of an inserted cell (B in Fig. 2) are shifted. No compaction is done in the row receiving the narrower cell. There are several reasons for this: 1) The fewest number of cells will have to be shifted. Therefore computation time is minimized. 2) As will be explained in Section II.4, when cells are shifted, one of the two models has to be employed to estimate the changes in the net lengths connecting to the cells being shifted. If fewer cells are shifted, then fewer nets will have to be estimated. Less estimation leads to less estimation inaccuracy and therefore a better solution can be obtained. 3) No compaction will leave some small gaps in the rows. This phenomenon will generally help to reduce the impact of subsequent shifting operations by "absorbing" the shifting amount that is needed to accommodate a new cell. 4) We control the row lengths to be no more than one average cell width over their ideal length and therefore the total width of the gaps in a row is at most a few average cell widths, a rather small amount. Overall, experiments verified that appreciably less computation time was used and yet a small improvement in the quality of the results was obtained compared with actually performing the compaction. The estimation model mentioned in line 15 will be explained in Section II.4.

C. New Cost Function

We now introduce the new incremental cost function.

$$\Delta C = \Delta W + \Delta W_S \tag{2}$$

ΔW is the change in the net lengths for those nets connected to the cell (or two cells) participating in the single cell (or exchange) move. In Fig. 2, this means those nets connected to A and B. ΔW_S represents the change in the net lengths for

those nets connected to the cells in the affected rows which must be shifted to avoid the creation of cell overlapping. In Fig. 2, this means those nets connected to the cells in the shaded region.

When a new cell is inserted into a row, other cells in this row generally need to be shifted to accommodate the new cell. As will be described in the next section, we have developed effective methods for estimating ΔW_S, the change in the net lengths for those nets connected to the shifted cells. That is, these *shifted* nets are not updated precisely. Consequently, when a cell is selected as a or b in Fig. 1, the net lengths stored for its nets are potentially inaccurate. To ensure calculation of the precise value of ΔW, before moving cell a (or exchanging cells a and b), we first recompute the current net lengths for all nets connected to cell a (or cells a and b). Then the move is made and the new net lengths are computed. Hence we obtain the precise value of ΔW. Precisely updating ΔW_S is prohibitively expensive. We therefore developed effective methods for estimating its value, as described next.

D. Wire Length Estimation Model

The prohibitively high computation cost for ΔW_S in (2) is the apparent drawback of this simplified cost function. A row usually consists of hundreds or thousands of standard cells and therefore a shifting operation on a row may disturb thousands of nets. We therefore proposed two models to effectively estimate ΔW_S.

Model A: When the standard cells in the core area are shifted in the rows, the shifting will change the wire length of those nets connected to the shifted cells. Under a half-perimeter net bounding box model, the shifting will only change the x-component of the bounding box and the size of the bounding box may increase, decrease, or stay the same.

Fig. 3 shows the changes in the x-span of a net's bounding box when one of the cells on the net is shifted. Here we show an example of a three-pin net n and three different conditions where the shifted cell j is at the left boundary of its bounding box in Fig. 3(a), at the right boundary of the bounding box in Fig. 3(b), and in the middle of the bounding box in Fig. 3(c). The top part of Fig. 3(a) depicts the position of shifted cell j in the bounding box. The graph of W_n versus x is generated by moving cell j while fixing all other cells. It shows the value of W_n as a function of the position of cell j in the x-direction, where W_n is wire length of net n in the x direction. W_n will decrease as cell j is shifted to the right until it reaches the *second* leftmost cell k. Then W_n stays the same until cell j is shifted to the right boundary of bounding box. Afterwards, W_n will begin to increase. The bottom graph show D_n, the derivative of W_n, versus x. Note that D_n is always a combination of two step functions and it has two break points. It can be computed efficiently by recording the two break points and then increasing D_n by 1 whenever passing a break point from left to right. Fig. 3(b) and (c) show the conditions where the shifted cell j is at the right boundary and the middle of its bounding box, respectively. Although we only show an example of a three-pin net, it is easy to extend the model to handle nets with more than three

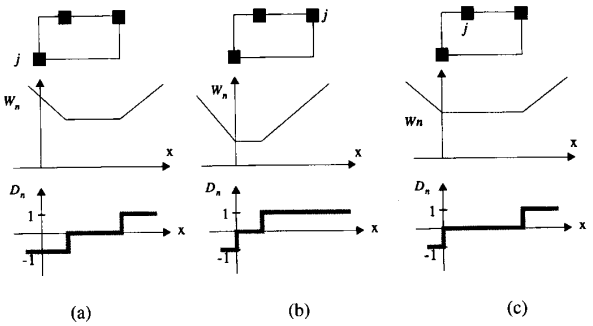


Fig. 3. Cell shifting for a net with three pins. The shifted cell j is (a) at the left boundary of the net's bounding box, (b) at the right boundary of the bounding box, (c) in the middle of the bounding box.

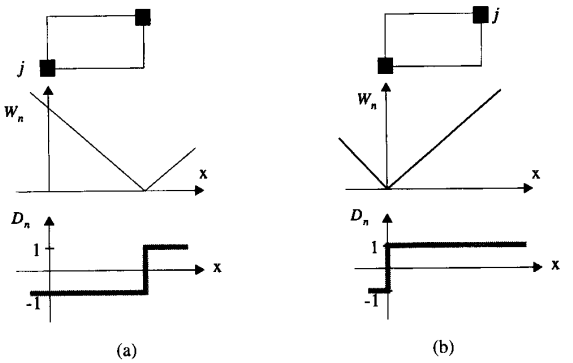


Fig. 4. Effect of cell shifting for a two-pin net. The shifted cell j is (a) at the left boundary of the net's bounding box, (b) at the right boundary of the bounding box.

pins. In this case D_n, still has two break points. The only modification is that when cell j sits at a boundary of its net bounding box, similar to the situation of Fig. 3(a) and (b), the position of the first break point is set to the position of the closest cell to j in the x direction.

Fig. 4 shows the cases when a net has only two pins. This is a degenerate case in which the two break points in D_n are coincident.

The estimation model referred to in line 15 of Fig. 1 can then be implemented as shown in Fig. 5. We call this *model A*:

If we assume the number of nets connected to any cell in a circuit is bounded and is not related to circuit size, then the estimation model in Fig. 5 can be updated in constant time. The assumption is reasonable; although the number of cells on a *net* is unbounded, the number of nets connected to one *cell* is always limited due to the limited area of a cell and the limited number of pins on a cell. Then ΔW_S can be estimated effectively using the following equation.

$$\Delta W_S = \sum_{i \in \{\text{shifted cells}\}} W_N(\text{ShiftAmount}(i)) \qquad (3)$$

Model B: In this section we describe a simpler model based on the more elaborate model presented in the previous section. Fig. 6 shows the similar scenario as presented in Fig. 3, but instead of calculating the full range of W_n and D_n, we are

1. For each net N_i connected to cell a
2. Record the two break points according to the position of a in N_i
3. Superimpose all of the break points to get the complete W_N versus x

Fig. 5. Model A.

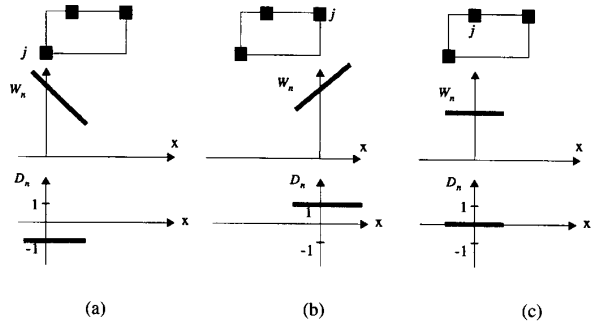


Fig. 6. Simplified cell shifting model. The shifted cell j is (a) at the left boundary of the net's bounding box, (b) at the right boundary of the bounding box, (c) in the middle of the bounding box.

1. **For each cell** a
2. *gradient(a)* = 0
3. **For each of the nets connected to** a:
4. **If** (a is at the *maximum* of its bounding box in the x-direction) **then**
5. *gradient(a)* ++
6. /* shifting a in the positive x-direction will increase the net's length */
7. **Else if** (a is at the *minimum* of its bounding box in the x-direction) **then**
8. *gradient(a)* --
9. /* shifting a in the negative x-direction will increase the net's length */

Fig. 7. Gradient computation.

now only interested in the derivative of W_n at the origin, i.e., $D_n(0)$.

Now we define the *gradient* value for a cell j as $\sum_{i\in\{\text{nets on cell } j\}} D_i(0)$, which is simply the sum of $D_n(0)$ for all the nets connected to cell j. Fig. 7 shows the details.

After we calculate the *gradient* value for each cell, ΔW_S is then estimated by multiplying the *gradient* value of the cell being shifted by the distance it is being shifted.

$$\Delta W_S = \sum_{j\in\{\text{shifted cells}\}} \text{gradient}(j)\cdot\text{ShiftAmount}(j) \quad (4)$$

This simplified *model B* is actually the first order linear approximation of the more detailed *model A*. It correctly predicts the effect of cell shifting if the amount of shifting is small relative to the net bounding boxes. Note that our new models correctly predict the local effect of a shifting operation. This distinguishes our method from that in [5] which presumes that all shifts increase the lengths of the affected nets.

E. Acceptance Rate Calculation

In Fig. 8, when cell A moves to another row, the change in cost, ΔC, will consist of two components, the x displacement and the y displacement. However, when cell A moves within the same row, ΔC will only have an x displacement. The value of the y displacement will be at least the distance between the rows. On the other hand, the x displacement can be as small as one grid size. In a typical circuit, the distance between the rows is usually tens or hundreds grid sizes. Hence the y displacement is generally several times larger than the x displacement. This phenomenon causes $|\Delta C|$ for an *inter-row* move, which has both x and y components, to generally be several times larger than for an *intra-row* move, which has only an x component.

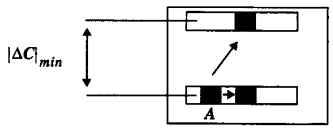


Fig. 8. Moves between the rows.

TABLE I
WIRE LENGTH IMPROVEMENT

Circuit	# cells	W Improvement
c1	1K	3%
c2	6K	5%
c3	15K	6%

When the simulated annealing acceptance criterion is used to evaluate moves, those moves that generate higher cost changes will generally have a much higher chance of being rejected. Thus, inter-row moves, which generate higher changes in cost, are much more likely to be rejected than intra-row moves at a given temperature. We found this to be especially severe when the temperature is low; we discovered that only intra-row moves were being accepted. This greatly hampered the search ability of simulated annealing since inter-row moves were not adequately exploited. We included only inter-row moves in calculating the acceptance rate. That is, when determining the actual acceptance rate we divide the number of inter-row moves accepted by the number of inter-row moves attempted and ignore all other moves. Using the annealing schedule described in [17], we obtained much better results as shown in Table I. Notice that the total wire length was 6% lower for the 15 000-cell circuit if the new acceptance rate calculation procedure is used.

III. HIERARCHICAL PLACEMENT

Fig. 9 shows our new hierarchical placement methodology which combines a new clustering technique with the new approach to simulated annealing. The original netlist is hierarchically clustered into various levels of netlists. Then the new approach to simulated annealing is used to place those various levels of netlists.

In the clustering stages, the clustering technique described in the next section is used to condense the original netlist into the first and then the second level netlists. The produced clusters in the higher level netlists have similar size, which greatly aids the annealing placement stages. In the placement stages, the condensed second level netlist is placed using the new

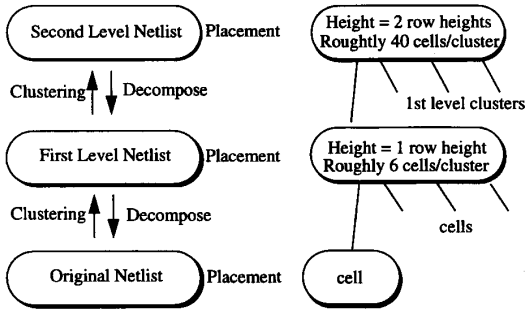


Fig. 9. Hierarchical placement.

approach to simulated annealing at the higher temperatures. Then we decompose the second level netlist back to the first level netlist. Cells of the lower level netlist will be randomly placed within the range of the cluster to which they belong in the higher level netlist. At the new lower level, these cells may then move outside the bounds of the higher level cluster. We then place the first level netlist at the middle temperatures, decompose the first level netlist back to the original netlist, and place the original netlist in the lower temperatures. In the current implementation, there are two clustering stages and three placement stages. For circuits much larger than 100 000 cells, we anticipate that it would be appropriate to add additional levels of clustering. Our combined clustering and simulated annealing methodology can be viewed as a combined bottom-up and top-down approach, where the two clustering stages provide bottom-up perspectives and the three placement stages provides a top-down view, from the course grain of the higher levels to the fine grain of the lower levels. Timing requirements are satisfied in all stages [16]. The right hand side of Fig. 9 shows some typical values in the current implementation.

A. The New Clustering Technique

We now present a new clustering technique based on graph connectivity. It produces clusters with about the same size, and emphasizes nets with small fan-out. Our new technique yields good results using a linear (in terms of the number of cells) time implementation. In our approach, each net i is given a weight w_i according to (5), where F_i is the set of pins for net i.

$$w_i = \frac{1}{|F_i| - 1} \tag{5}$$

Assume for the moment that N, U and L have been specified, where N is the number of clusters desired, U and L are the upper and lower bounds, respectively, for the total length (size) of the standard cells permitted in a cluster. The situation is depicted in Fig. 10. Our objective is to find a partitioning of the cells over the B_N bins such that no bin contains less than L total cell length nor more than U total cell length and that the total weight of the intra-cluster *edges* is maximized. Note that this is equivalent to minimizing the total weight of the inter-cluster edges. That is, we want to collapse, or eliminate, as many nets as possible. If we cannot collapse a net, then we want a net to occupy as few clusters as possible. The key point is that we want our clustering procedure to focus its attention on low fan-out nets since these are precisely the nets whose spans can be reduced by a placement algorithm. Fig. 10 shows a five-pin net which spans three clusters.

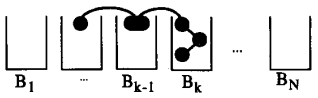


Fig. 10. Clusters.

Instead of using a clique model for multipin nets, we have obtained better results by using a tree model. In this model, given an n-pin net which has cells in m different bins, then there are $m-1$ edges for the inter-cluster connections, and if there are k pins in a given bin, then there are $k-1$ edges contained within the bin. Obviously we must have that the sum of the $k-1$ values for each of the m bins plus $m-1$ is equal to $n-1$. For example, consider the five-pin net in Fig. 10. Here $m=3$ and the sum of the three $k-1$ values are $0+0+2=2$, and 2 added to $m-1$ equals 4. The weight assigned to each of these 4 edges is that given by (5). For this example we find that w_i is 0.25. The weight value W_k for cluster k is defined in (6), where B_k is the set of all pins currently in cluster k. The weight W_k for cluster k is the sum of all edge weights in cluster k.

$$W_k = \sum_{\langle \forall i | (F_i \cap B_k) \neq \emptyset \rangle} (|F_i \cap B_k| - 1) \cdot w_i \tag{6}$$

The net in Fig. 10 contributes 0.50 to the weight W_k for cluster k. Note that when all of the pins F_i for net i are merged into just one cluster, the total weight contributed to this cluster (or bin) is always 1.0. This model naturally places a higher emphasis on low fan-out nets than on high fan-out nets since it is much easier to collapse a low fan-out net into a single bin.

We found that the effectiveness of the placement algorithm deteriorated as the aspect ratio of the clusters approached 100. We therefore set the target cluster length to be $10 \cdot \bar{H}$ where $\bar{H}$ is the average cell height. We then set U equal to $30 \cdot \bar{H}$ and L equal to $3 \cdot \bar{H}$. In this way, the variation of cluster sizes is within one order of magnitude, comparable to the normal variation in cell sizes for a flat standard cell netlist. Now the objective is to maximize (7) subject to the cluster capacity constraints.

$$C = \sum_{k=1}^{N} W_k \tag{7}$$

Simulated annealing has proven to be the best approach for maximizing (7). Fig. 11 shows the inner loop of our annealing procedure. First, cell a is randomly selected and we note the bin l where it resides. We obtained much better results when we restricted the move generation process to generating moves which necessarily have the potential to increase the value of (6). This can be accomplished readily by restricting the move of cell a to those bins which contain fan-out cells of cell a. Along these lines, a net i connected to a is randomly selected and a pin j on this net is randomly selected. Next, we note

1. Randomly pick a cell a
2. Locate the bin l where a resides
3. Define the set of nets connected to a as N_a
4. Randomly pick a net i from N_a
5. Define the set of pins for net i as F_i
6. Randomly pick a pin j in F_i
7. Locate the bin k where j resides
8. **If** (moving a to bin k violates capacity constraints) **then**
9. Go to line 1
10. Compute $\Delta W_l = -\sum_{\forall i \in N_a} min(1, |F_i \cap B_l| - 1) \cdot w_i$
11. Compute $\Delta W_k = \sum_{\forall i \in N_a} min(1, |F_i \cap B_k|) \cdot w_i$
12. $\Delta C = \Delta W_l + \Delta W_k$
13. **If**($accept(-\Delta C)$) **then**
14. move a to bin k

Fig. 11. Generating clusters.

the bin k where pin j resides. If cell a can be moved to bin k without violating the capacity constraints for bins l and k, we then propose a move for cell a to bin k. Such a move will only affect the total weights in bins l and k, that is, W_l and W_k. Line 10 and 11 compute the differences in these weights ΔW_l and ΔW_k, derived from (6). Then in line 12 we compute ΔC, the change in cost for this move.

We implemented a hash table for each bin k to store all the nets connected to cells in k and therefore the intersection operation in line 10 and 11 can be done in constant time, as long as the number of nets in k is bounded. This can be inferred as follows. Since we have an upper bound U for the total cell width in a bin, the number of cells in a bin is bounded. For industrial circuits, the number of nets on a cell is also bounded (usually under 10), therefore, the total nets in bin k is also bounded. Therefore the time complexity for the inner loop shown in Fig. 11 is a constant. Our experiments have shown that to generate a good clustering, it is sufficient to set the number of outer loop iterations (that is, the number of calls to the procedure in Fig. 11) to be $100n$, where n is the number of cells. Therefore, our total time complexity is linear. We use the annealing schedule described in [17].

B. The Placement Stages

In the placement stages, the annealing method described in Section II is used to place the different levels of netlist produced by the clustering technique. The first level netlist is placed in the higher temperature regime. This first stage comprises the first 50% of the total annealing schedule as shown in Fig. 12. The second placement stage starts at 50% and ends at 70% of the total annealing schedule as shown in Fig. 12. The constituent cells belonging to a cluster are randomly placed within the confines of the location of the bounding rectangle of the cluster as determined in the previous stage. The third stage starts at 70% of the total annealing schedule. The restarting temperature T of the second (and third) placement stage is given by (8), where $\overline{\Delta W}$ is the

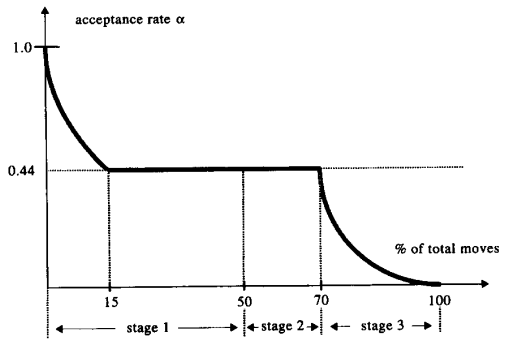


Fig. 12. Annealing schedule in hierarchical mode.

TABLE II
TEST CIRCUITS

Circuit	# modules	# nets	# pins	# rows
Primary 1	752	904	5526	16
Primary 2	2907	3029	18407	28
Biomed	6417	5742	26947	46
Industry 2	12142	13419	125555	72
Industry 3	15059	21940	176584	54
Avqsmall	21854	22124	82601	80
Avqlarge	25114	25384	82751	86
Golem3	99932	143379	336299	192

average net length and α is the target acceptance rate.

$$T = -\frac{\overline{\Delta W}}{\log(\alpha)} \tag{8}$$

In all placement stages, timing requirements are satisfied using the precise timing driven method in [16].

IV. RESULTS

Table II shows the circuit parameters of our test circuits. All seven circuits are from the 1993 MCNC layout benchmark sets [9]. All reported computation times are CPU seconds for a DEC station 5000/200.

A. The Wire Estimation Model

Our new implementation of simulated annealing constitutes *TimberWolfSC v7.0*. In an effort to test the effectiveness of the two wire length estimation models, we also developed a several orders of magnitude slower routine which exactly updates ΔW_S, the change in the net lengths for all the nets connected to the cells being shifted. We then compared the final total wire lengths achieved from *TimberWolfSC v7.0* with the exact updating routine (the *Exact* column in Table III) with the two models which estimate ΔW_S (column *model A* and *model B* in Table III). That is, in Table III, the wire length shown under *Exact* means that ΔW_S in (2) was computed exactly by updating all the nets involved during the shifting operation. The columns under *model A* and *model B* imply that ΔW_S is estimated using model A and model B, respectively, described in Section II.4. In the *Exact* column, we were only

TABLE III
WIRE LENGTH COMPARISON, EXACT VERSUS ESTIMATED

Circuit	W (*Exact*)	$\overline{SD}$	W (*model A*)	$\overline{SD}$	W (*model B*)	$\overline{SD}$
Primary 1	0.84	0.38	0.83	0.50	0.83	0.69
Primary 2	3.67	0.36	3.66	0.46	3.53	0.58
Biomed	3.61	0.35	3.43	0.55	3.55	0.65
Avqlarge	-	-	6.36	0.59	6.50	0.69

TABLE IV
RUN TIME COMPARISON, EXACT, MODEL A AND MODEL B

Circuit	Time (*Exact*)	Time (*model A*)	Time (*model B*)
Primary 1	2265	1307	822
Primary 2	33647	13320	7301
Biomed	110735	36915	20513
Avqlarge	-	305624	164322

able to obtain the results for the first three circuits due to the slowness of the exact routine. The $\overline{SD}$ columns are the average shift distances in terms of the average standard cell width. Table IV shows the run time comparison.

The results show that both of our estimation models yield comparable or better results compared with exactly updating ΔW_S. The computation time for model A is about two times that for model B because of the complexity of model A (Fig. 5). In comparing model A and model B, we also find $\overline{SD}$, the average shift distance, is about half the average width of standard cells in the circuit. That $\overline{SD}$ is small is crucial for the success of model B, since it is based on the assumption that the shift distance is small. The results show that model B performs comparably to model A while using about half of the computation time. Therefore we use model B in *TimberWolfSC v7.0*.

B. TimberWolfSC v7.0 Flat Mode

Tables V–IX show the average results of *TimberWolfSC v7.0* in flat mode compared with TimberWolfSC v6.0 [13], [14]. Table V shows the reduction in total wire length. *TimberWolfSC v7.0* outperforms v6.0 for all benchmark circuits. The average wire length reduction is 8%. Table VI shows the reduction in the length of the longest row. This is important because the chip width is determined by the size of the longest row. For all of the benchmarks, we see tremendous reductions both in total wire length and chip width. Because our algorithm is based on simulated-annealing, a different seed for the random number generator will produce slightly different results. The results usually vary in a small range, within a few percentage points. Therefore the results we have shown are not produced by a single run, but are rather the average of three runs. Other than the random seed, there are no user-tunable parameters in our implementation.

Tables VII and VIII show the results after global routing. The global router we used is described in [18]. Table VII shows the number of tracks after global routing. The average track reduction is 5%. Table VIII shows the final chip area comparison and the average reduction is 7%. The computation

TABLE V
WIRE LENGTH COMPARISON, TIMBERWOLF V6 VERSUS V7 FLAT

Circuit	Wire length *TW v6.0*	Wire length *TW v7.0*	Wire length Reduction
Primary 1	0.90	0.82	9%
Primary 2	3.71	3.50	6%
Biomed	3.94	3.49	11%
Industry 2	14.69	13.69	7%
Industry 3	48.38	44.02	9%
Avqsmall	6.72	6.15	8%
Avqlarge	6.93	6.50	6%
average	-	-	8%

TABLE VI
CHIP WIDTH COMPARISON, TIMBERWOLF V6 VERSUS V7 FLAT

Circuit	Chip width *TW v6.0*	Chip width *TW v7.0*	Chip width Reduction
Primary 1	5260	5100	3%
Primary 2	8420	8210	3%
Biomed	10328	9856	5%
Industry 2	14896	13976	6%
Industry 3	29784	26224	12%
Avqsmall	9560	9072	5%
Avqlarge	9824	9344	5%
average	-	-	6%

TABLE VII
TRACKS COMPARISON, TIMBERWOLF V6 VERSUS V7 FLAT

Circuit	# Tracks *TW v6.0*	# Tracks *TW 7.0*	Track Reduction
Primary 1	164	155	6%
Primary 2	434	430	1%
Biomed	850	825	3%
Industry 2	1133	1080	5%
Industry 3	1909	1747	9%
Avqsmall	1009	985	2%
Avqlarge	1035	988	5%
average	-	-	5%

time spent in flat mode is longer than that of TimberWolfSC v6.0 due to the overhead of the shifting operation (Section II.2) and due to the recomputation of the old net lengths for the cells involved in an exchange (Section II.3). The increases in CPU time ranged from 5% to 80% for the test circuits. Note that these kinds of reductions in chip area can not be achieved in TimberWolfSC v6.0, even if it is run much longer.

C. TimberWolfSC v7.0 Hierarchical Mode

Tables X–XIII summarize the results of our new hierarchical placement approach in *TimberWolfSC v7.0*, compared with TimberWolfSC v6.0. Table X shows the reduction in wire length, which is 12% on average. Tables XI and XII show the track counts and chip area after global routing. Up to a 21% area reduction is achieved, with the average area reduction

TABLE VIII
AREA COMPARISON, TIMBERWOLF V6 VERSUS V7 FLAT

Circuit	Area *TW v6.0*	Area *TW v7.0*	Area Reduction
Primary 1	20.4	19.4	5%
Primary 2	68.3	66.3	3%
Biomed	115	109	5%
Industry 2	212	195	8%
Industry 3	753	637	15%
Avqsmall	108	104	4%
Avqlarge	119	111	7%
average	-	-	7%

TABLE IX
RUN TIME COMPARISON, TIMBERWOLF V6 VERSUS V7 FLAT

Circuit	Run time *TW v6.0*	Run time *TW v7.0*
Primary 1	794	830
Primary 2	5316	7322
Biomed	14631	20782
Industry 2	37521	69725
Industry 3	65652	120485
Avqsmall	92959	133023
Avqlarge	96564	165941

TABLE X
WIRE LENGTH COMPARISON, TIMBERWOLF V6 VERSUS V7 HIERARCHICAL

Circuit	Wire *TW v6.0*	Wire *TW v7.0*	Wire Reduction	Wire *TW v7.0**	Wire Reduction
Primary 1	0.90	0.84	7%	0.83	8%
Primary 2	3.71	3.57	4%	3.53	5%
Biomed	3.94	3.24	18%	3.22	18%
Industry 2	14.69	13.53	8%	13.30	10%
Industry 3	48.38	42.84	12%	41.53	14%
Avqsmall	6.72	5.41	19%	5.08	24%
Avqlarge	6.93	5.86	16%	5.65	19%
Golem3	107.69	93.10	14%	88.98	17%
average	-	-	12%	-	14%

TABLE XI
TRACKS COMPARISON, TIMBERWOLF V6 VERSUS V7 HIERARCHICAL

Circuit	# Tracks *TW v6.0*	# Tracks *TW v7.0*	Track Reduction	# Tracks *TW v7.0**	Track Reduction
Primary 1	164	161	2%	152	7%
Primary 2	434	425	2%	412	5%
Biomed	850	773	9%	773	9%
Industry 2	1133	1030	9%	1004	11%
Industry 3	1909	1478	23%	1436	25%
Avqsmall	1009	873	13%	858	15%
Avqlarge	1035	882	15%	882	15%
Golem3	4049	3141	22%	2928	28%
average	-	-	12%	-	14%

TABLE XII
AREA COMPARISON, TIMBERWOLF V6 VERSUS V7 HIERARCHICAL

Circuit	Area *TW v6.0*	Area *TW v7.0*	Area Reduction	Area *TW v7.0**	Area Reduction
Primary 1	20.4	19.6	4%	19.2	6%
Primary 2	68.3	66.0	3%	65.0	5%
Biomed	115	104	10%	104	10%
Industry 2	212	191	10%	189	11%
Industry 3	753	595	21%	588	22%
Avqsmall	106	96.1	9%	94.7	11%
Avqlarge	119	101	15%	101	15%
Golem3	851	689	19%	672	21%
average	-	-	11%	-	13%

TABLE XIII
RUN TIME COMPARISON, TIMBERWOLF V6 VERSUS V7 HIERARCHICAL

Circuit	Run time *TW v6.0*	Run time *TW v7.0*	Speed Up
Primary 1	794	221	3.59
Primary 2	5316	1252	4.25
Biomed	14631	2164	6.76
Industry 2	37521	9252	4.06
Industry 3	65652	8766	7.48
Avqsmall	92959	13018	7.14
Avqlarge	96564	15597	6.19
Golem3	896487	141954	6.32

being 11%. These MCNC benchmark results are the best results ever reported. In Tables X–XII, the column labeled TW *v7.0** are the results of *doubling* the total number of moves in our annealing schedule. Therefore the run time is twice that of the regular *TW v7.0* as shown in Table XIII. This trade-off yields a 2% lower wire length on average in exchange for a doubling of the run time. Table XIII shows the speed-up achieved with our new hierarchical placement algorithm. The hierarchical algorithm yields much better results compared with TimberWolfSC v6.0 and runs up to 7.5 times faster.

D. Gordian/Domino versus TimberWolfSC v7.0

In Tables XIV–XVII, we compare the Gordian/Domino placement package [2], [3], [8] with *TimberWolfSC v7.0* in hierarchical placement mode. Table XIV compares the total wire lengths (in meters) for the seven MCNC benchmark circuits. On average, *TimberWolfSC v7.0* produced placements having 8% less total wire length. We use the same router [18] to produce the routing results in Tables XV and XVI. Table XV shows that up to a 16% track reduction is obtained, with the average being 9%. Table XVI shows the chip area comparison. In Tables XIV–XVII the column labeled TW *v7.0** are again the results due to a doubling of the computation time. This produced an additional 2% reduction in wire length, on average. This shows the trade-off between run time and the quality of the final result.

Table XVII compares the *TimberWolfSC v7.0* run time with Gordian/Domino (CPU times are in seconds). For circuits with more than 5 000 cells (our circuit size range of interest), *TimberWolfSC v7.0* uses up to 26% less computation time.

E. Ritual/Tiger versus TimberWolfSC v7.0

In this section, we compare the results of Ritual/Tiger [15] and *TimberWolfSC v7.0* in hierarchical mode. Table XVIII shows the characteristics of the three test circuits.

TABLE XIV
WIRE LENGTH COMPARISON, GORDIAN/DOMINO VERSUS TimberWolf v7 HIERARCHICAL

Circuit	Wire *Gordian/ Domino*	Wire *TW v7.0*	Wire Reduction	Wire *TW v7.0**	Wire Reduction
Primary 1	0.88	0.84	5%	0.83	6%
Primary 2	3.68	3.57	3%	3.53	4%
Biomed	3.91	3.24	17%	3.22	18%
Industry 2	15.80	13.53	14%	13.30	16%
Industry 3	44.97	42.84	5%	41.53	8%
Avqsmall	5.68	5.41	5%	5.08	11%
Avqlarge	6.21	5.86	6%	5.65	9%
average	-	-	8%	-	10%

TABLE XV
TRACKS COMPARISON, GORDIAN/DOMINO VERSUS TimberWolf v7 HIERARCHICAL

Circuit	# Tracks *TW v6.0*	# Tracks *TW 7.0*	Track Reduction
Primary 1	164	155	6%
Primary 2	434	430	1%
Biomed	850	825	3%
Industry 2	1133	1080	5%
Industry 3	1909	1747	9%
Avqsmall	1009	985	2%
Avqlarge	1035	988	5%
average	-	-	5%

TABLE XVI
AREA COMPARISON, GORDIAN/DOMINO VERSUS TimberWolf v7 HIERARCHICAL

Circuit	Area *Gordian/ Domino*	Area *TW v7.0*	Area Reduction	Area *TW v7.0**	Area Reduction
Primary 1	19.6	19.6	0%	19.2	2%
Primary 2	69.2	66.0	5%	65.0	6%
Biomed	112	104	7%	104	7%
Industry 2	205	191	7%	189	8%
Industry 3	609	595	2%	588	3%
Avqsmall	99.3	96.1	3%	94.7	5%
Avqlarge	110	101	8%	101	8%
average	-	-	5%	-	6%

TABLE XVII
RUN TIME COMPARISON, GORDIAN/DOMINO VERSUS TimberWolf v7.0 HIERARCHICAL

Circuit	Run Time *Gordian/ Domino*	Run Time *TW v7.0*	Reduction
Primary 1	168	221	-
Primary 2	922	1252	-
Biomed	2640	2164	18%
Industry 2	9587	9252	3%
Industry 3	10349	8766	15%
Avqsmall	17444	13018	25%
Avqlarge	21086	15597	26%

Table XIX shows the wire length comparison. *TimberWolfSC v7.0* produces on average 11% lower wire length than that of RITUAL. Tables XX and XXI show the results after global routing. In Table XX, up to a 19% reduction in the track count is obtained, and the average track reduction is 11%. In Table XXI, up to a 13% chip area reduction is obtained, and the average area reduction is 8%. In Tables XIX–XXI, the column labeled TW *v7.0** are the results due to a doubling of the computation time. On average, this produces an additional 2% reduction in chip area.

Tables XXII and XXIII shows the average performance improvement of *TimberWolfSC v7.0* over TimberWolfSC v6.0, Ritual, and Gordian/Domino in three categories: wire length, number of tracks after global routing, and final chip area.

TABLE XVIII
RITUAL TEST CIRCUITS

Circuit	# Cells	# Nets	# Pins	# Pads	# Rows
c2	590	823	2060	373	9
c7	2150	2357	6380	315	18
s13207	4267	4967	13175	1490	24

TABLE XIX
WIRE LENGTH COMPARISON, RITUAL VERSUS TimberWolf v7 HIERARCHICAL

Circuit	Wire *Ritual*	Wire *TW v7.0*	Wire Reduction	Wire *TW v7.0**	Wire Reduction
c2	0.31	0.29	6%	0.28	10%
c7	1.27	1.07	16%	1.02	20%
s13207	5.75	5.17	10%	5.07	12%
average	-	-	11%	-	14%

TABLE XX
TRACKS COMPARISON, RITUAL VERSUS TimberWolf v7 HIERARCHICAL

Circuit	# Tracks *Ritual*	# Tracks *TW v7.0*	Tracks Reduction	# Tracks *TW v7.0**	Tracks Reduction
c2	134	127	5%	122	9%
c7	298	242	19%	234	21%
s13207	811	738	9%	717	12%
average	-	-	11%	-	14%

TABLE XXI
AREA COMPARISON, RITUAL VERSUS TimberWolf v7 HIERARCHICAL

Circuit	Area *Ritual*	Area *TW v7.0*	Area Reduction	Area *TW v7.0**	Area Reduction
c2	3.96	3.83	3%	3.74	6%
c7	14.8	12.9	13%	12.6	15%
s13207	50.1	45.9	8%	45.3	10%
average	-	-	8%	-	10%

TABLE XXII
PERFORMANCE COMPARISON, TimberWolf v7.0 VERSUS OTHERS

TW v7.0	TW v6.0	Ritual	Gordian/ Domino
Wire length	12%	11%	8%
Tracks	12%	11%	9%
Area	11%	8%	5%

V. CONCLUSION

We presented a new approach to simulated annealing for row-based placement, and a hierarchical placement algorithm.

TABLE XXIII
PERFORMANCE COMPARISON, TIMBERWOLF V7.0* VERSUS OTHERS

TW v7.0*	TW v6.0	Ritual	Gordian/ Domino
Wire length	14%	14%	10%
Tracks	14%	14%	11%
Area	13%	10%	6%

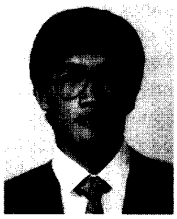
We have obtained the best results ever reported for a large set of MCNC benchmark circuits. Our new hierarchical annealing-based placement algorithm yields chip area reductions of up to 21% while consuming up to 7.5 times less CPU time in comparison to TimberWolfSC v6.0. Furthermore, *TimberWolfSC v7.0* produces lower total wire length by an average of 8% than Gordian/Domino, 11% lower wire length than Ritual, while using comparable run time. *TimberWolfSC v7.0* supports precise timing driven placement [16].

ACKNOWLEDGMENT

The authors would like to acknowledge K. Doll of the Technical University of Munich for providing the latest version of the Gordian and Domino package and Prof. E. Kuh of the University of California, Berkeley, for providing the Ritual/Tiger test circuits and results.

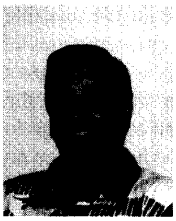
REFERENCES

[1] J. Cong, L. Hagen, and A. Kahng, "Random walks for circuit clustering," *Proc. IEEE Intl. Conf. on ASIC*, June 1991, pp. 14.2.1–14.2.4.
[2] K. Doll, F. M. Johannes, and G. Sigl, "Accurate net models for placement improvement by network flow methods," *Proc. Int. Conf. on Computer-Aided Design*, 1992, pp. 594–597.
[3] ——, "Domino: deterministic placement improvement with hill-climbing capabilities," *Proc. VLSI*, 1991, pp. 3b.1.1–3b.1.10.
[4] J. Garbers, H. J. Promel, and A. Steger, "Finding clusters in VLSI circuits," *Proc. Int. Conf. on Computer-Aided Design*, 1990, pp. 520–523.
[5] L. K. Grover, "A new simulated annealing algorithm for standard cell placement," *Proc. Int. Conf. on Computer-Aided Design*, 1986, pp. 378–380.
[6] L. Hagen and A. B. Kahng, "A new approach to effective circuit clustering," *Proc. Int. Conf. on Computer-Aided Design*, 1992, pp. 422–427.
[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Sci.*, vol. 220, no. 4598, pp. 671–680, May 13, 1983.
[8] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 3, 1991, pp 356–365.
[9] K. Kozminski, "Benchmarks for layout synthesis," *Proc. 28th Design Automation Conf.*, 1991, pp. 265–270.
[10] J. Lam and J. M. Delosme, "Performance of a new annealing schedule," *Proc. 25th Design Automation Conf.*, 1988, pp. 306–311.
[11] S. Mallela and L. K. Grover, "Clustering based simulated annealing for standard cell placement," *Proc. 25th Design Automation Conf.*, 1988, pp. 312–317.
[12] G. Sigl, K. Doll and F. M. Johannes, "Analytical placement: a linear or a quadratic objective function?" *Proc. Design Automation Conf.*, 1991, pp. 427–432.
[13] C. Sechen, *VLSI placement and global routing using simulated annealing*. Boston: Kluwer Academic, 1988.
[14] C. Sechen and K. W. Lee, "An improved simulated annealing algorithm for row-based placement," *Proc. Int. Conf. on Computer-Aided Design*, 1987, pp. 478–481.
[15] A. Srinivasan, K. Chaudhary, and E. S. Kuh, "RITUAL: A performance driven placement algorithm for small cell IC's," *Proc. Int. Conf. on Computer-Aided Design*, 1991, pp. 48–51.
[16] W. Swartz, "Automatic layout of analog and digital mixed macro/standard cell integrated circuits," Ph.D. dissertation, Yale University, CT, 1993.
[17] W. Swartz and C. Sechen, "New algorithms for the placement and routing of macro cells," *Proc. 27th Design Automation Conf.*, 1988, pp. 336–339.
[18] ——, "A new generalized row-based global router," *Proc. Intl. Conf. on Computer-Aided Design*, 1993, pp. 491–498.
[19] C. W. Yeh, C. K. Cheng, and T. T. Lin, "A probabilistic multicommodity-flow solution to circuit clustering problems," *Proc. Int. Conf. on Computer-Aided Design*, 1992, pp. 428–431.



Wern-Jieh Sun received the B.S.E.E. degree from National Taiwan University in 1987 and the M.S. degree from Yale University, New Haven, CT, in 1992. In 1994, he received the Ph.D. degree from University of Washington.

In 1989, he was a full time teaching assistant with the Department of Electrical Engineering at National Taiwan University. From 1990 to 1994, he was research assistant and Ph.D candidate both at Yale University and University of Washington. He has worked at Cadence, National Semiconductor, Intel, and AMD during the summers of 1991 to 1994. His main research interests are in the area of VLSI design and CAD.



Carl Sechen received the B.E.E. degree from Minnesota and the M.S. degree from M.I.T. In 1987 he received the Ph.D. degree from the University of California, Berkeley.

From July 1986 through June 1992 he was an Assistant and then an Associate Professor at Yale University, New Have, CT. Since July 1992 he has been an Associate Professor with the Department of Electrical Engineering at the University of Washington. His primary research interests are the design and computer-aided design of analog and digital integrated circuits. He is the principal developer of the TimberWolf placement and routing package. He is a consultant for DEC, Intel, National Semiconductor, Motorola, AMD, IBM, and Cadence.