Chip-Planning, Placement, and Global Routing of Macro/Custom Cell Integrated Circuits Using Simulated Annealing

Carl Sechen

Dept. of Electrical Engineering
Yale University
New Haven, CT 06520

Abstract

The algorithms and the implementation of a new macro/custom cell chip-planning, placement, and global routing package are presented. The simulated-annealing-based placement algorithm proceeds in two stages. In the first stage, the interconnect area around the individual cells is determined using a new dynamic interconnect area estimator. The second stage consists of: (1) a channel definition step, using a new channel definition algorithm, (2) a global routing step, using a new global router algorithm, and (3) a placement refinement step. This strategy has produced placements which require very little placement modification during detailed routing. Total interconnect length savings of 8 to 49 percent were achieved in experiments on 9 industrial circuits. Furthermore, circuit-area reductions ranged from 4 to 56 percent versus a variety of other placement methods.

1. Introduction

The algorithms and the implementation of a new macro/custom cell placement and global routing package are presented. This package, named *TimberWolfMC*, makes extensive use of the simulated annealing algorithm. Simulated annealing has been an effective algorithm upon which to base a standard cell placement and global routing package.[1,2,3,4,5] The use of simulated annealing was therefore considered for the generalized macro/custom cell placement problem.

TimberWolfMC is applicable to circuits containing cells of any rectilinear shape. Furthermore, the cells may have fixed geometry including pin locations (*macro cells*) or the cells may have an estimated area with a specified aspect ratio range, and with pins that need to be placed (*custom cells*). TimberWolfMC also permits the custom cells to have aspect ratios in a continuous or discrete range. Furthermore, the cells may have several possible instances, whereby TimberWolfMC is to select the one which is most suitable. The selection of aspect ratio and/or instance is guided by the minimization of the Total Estimated Interconnect Cost (TEIC) and by the geometry of the empty space allotted for the cell as influenced by the neighboring cells. TimberWolfMC places circuits consisting entirely of macro cells as well as circuits consisting entirely of custom cells. Furthermore, the program will place circuits consisting of a combination of macro and custom cells. Consequently, TimberWolfMC is applicable to chip planning problems.

The TEIC calculation is based on the exact pin locations, as opposed to assuming that all of the pins associated with a cell have a location equal to the center of the cell. Consequently, all eight possible orientations are considered for each cell. A new algorithm for accurately estimating the interconnect area around the individual cells is used in TimberWolfMC. This algorithm has resulted in the generation of placements which require very little placement modification during or after detailed routing.

The placement algorithm proceeds in two distinct stages. During the first stage, the interconnect area around the individual cells is determined using the new, *dynamic*, interconnect-area estimator. That is, as each cell is moved, its effective area is adjusted. A simulated annealing algorithm is used to minimize the TEIC. The second stage of TimberWolfMC consists of several executions of a placement-refinement algorithm. Each execution consists of three steps: (1) a channel definition step, using a new channel definition algorithm, (2) a global routing step, using a new global router algorithm, and (3) a placement-refinement step. The information obtained in step 2 is used to compute the density of all of the channels, which then permits accurate interconnect-area determination. The placement of the cells is then refined in step 3 to reflect the required interconnect area. A low-temperature simulated annealing algorithm is used to accomplish step 3. Three such placement refinement steps are sufficient for the final TEIC and the final chip area to converge.

Total interconnect length savings of 8 to 49 percent were achieved in experiments on 9 industrial circuits, in comparison to several automatic and manual placements. Furthermore, for the 9 examples, circuit-area reductions ranged from 4 to 56 percent in comparison to the other placements.

The first report of a macro cell placement program using simulated annealing was by Jepsen and Gelatt.[6] An implementation of simulated annealing for floorplan design was reported by Otten and van Ginneken.[7] More recently, another implementation of simulated annealing was reported by Wong and Liu for floorplan design.[8] None of these previous placement programs were able to handle a combination of macro and custom cells on the same chip. That is, these previous programs could not handle the simultaneous problems of pin placement, instance selection, aspect-ratio selection, orientation selection, rectilinear cells, fixed cell geometry, and placement. Furthermore, none of these programs featured interconnect area estimation, whereas TimberWolfMC features *two* stages of interconnect area estimation. This enables TimberWolfMC to generate placements which require little (if any) placement modification during detailed routing.

The paper is organized as follows: Section 2 presents some details of the general TimberWolfMC methodology. In Section 3, the algorithms for the first stage of TimberWolfMC are presented. The algorithms for the second stage, including the placement-refinement procedure, are described in Section 4. The experimental results are presented in Section 5. Finally, the conclusion is the subject of Section 6.

2. The General TimberWolfMC Methodology

This section covers three aspects of TimberWolfMC which provide flexibility and robustness to the general methodology and algorithms. The first subsection briefly presents the basic simulated annealing algorithm. In the next subsection, the key aspects of the new dynamic interconnect-area estimator are reviewed. The third subsection describes the generation of the initial placement configuration. Finally, the last subsection introduces the custom-cell pin placement methodology.

2.1 The Basic Simulated Annealing Algorithm

Simulated annealing was proposed by Kirkpatrick, et. al., as an effective method for the solution of combinatorial optimization problems involving the minimization of a function over many degrees of freedom.[9] Complete accounts of the structure and terminology of the TimberWolfMC implementation of the simulated annealing algorithm were previously published.[2] The algorithm is characterized by: (1) the generation function *generate*, (2) the acceptance function *accept*, (3) the updating function *update*, (4) the *inner loop criterion* and (5) the *stopping criterion*. TimberWolfMC uses several implementations of simulated annealing.

2.2 A New Dynamic Interconnect-Area Estimator

For macro and custom cells, it is usually the case that there are pins on all of the edges of the cells. It is therefore necessary to allocate interconnect space around each cell. Failure to allocate the correct amount invariably results in significant placement alteration by the global and/or detailed routers. These placement modifications usually result in substantial increases in total interconnect length and chip area, due to the introduction of sizable empty areas on the chip. Furthermore, attempting to find the most suitable cell-placement manipulation is indeed a challenging problem. Solving this type of problem requires either a spacer or a placement refinement algorithm.[10] Since the placement adjustments are local in nature, suboptimal solutions are likely.

Unfortunately, it is very difficult to compute the precise amount of interconnect space needed along each edge of a cell without performing a global routing step. Furthermore, it is even more difficult to compute the interconnect space during an intermediate point of a simulated annealing approach in which a penalty function is used to limit the amount of cell overlapping. During such an intermediate point, it is likely that several cases of cell overlapping are present. This makes a channel definition procedure very difficult to perform. Thus, while it would be best to maintain a global routing of the placement at any given point in the simulated annealing process, such a procedure is very difficult at best, and surely very cpu-time intensive.

Methods have been introduced which appear to give good results for the *static* interconnect-area estimation problem.[11 12] However, these methods appear not to be suitable for the *dynamic* estimation problem, in which it is necessary to update the estimates millions of times during the course of an execution of a simulated annealing algorithm. Furthermore, the previous methods assume that the pin locations on the various cells have a statistical distribution. However, in the problems of interest here, the pin locations for a given cell are either precisely known (macro cells) or approximately known (custom cells).

My new algorithm for the dynamic interconnect area estimation problem bases the estimate on three factors:[13]

(1) The *average net traffic*. This is an estimate of the average number of interconnections passing through a channel. This estimate is used to derive the expected *average channel width*, represented by C_W:

$$C_W = \frac{N_L}{C_L}\, t_s \tag{1}$$

where N_L is an accurate estimate of the final total interconnect length, C_L is the estimate of the total channel length, and t_s is the center-to-center wiring track separation.[14 15]

(2) The position of the channel on the chip. In macro/custom cell layout, it is often the case that the shortest possible route is used for a net. Consequently, it would be expected that the thicknesses of the channels would be greater nearer the center of the core area. This effect is modeled by multiplying C_W by a modulation function, whose value reflects the position of the channel. This modulation function is implemented as the composite of two modulation functions, one reflecting the vertical position of the channel and the other reflecting the horizontal position. The center of the core region is set at $x = 0$ and $y = 0$, with the width of the core represented by W and the height of the core represented by H.

The expression for the horizontal modulation function is given by:

$$f_x(x) = M_x - |x| \left\{ \frac{M_x - B_x}{0.5\, W} \right\}$$

where $-0.5W \le x \le 0.5W$. Note that f_x has a maximum value of M_x for $x = 0$ and a minimum value of B_x for $x = \pm 0.5$. The expression for the vertical modulation function is given by:

$$f_y(y) = M_y - |y| \left\{ \frac{M_y - B_y}{0.5\, H} \right\}$$

where $-0.5H \le y \le 0.5H$.

The functional forms for f_x and f_y were chosen based on the observation of manual layouts of industrial macro cell chips. For two layers of interconnect, the channels near the center of the core usually were approximately twice as wide as channels located near the middle of the sides of the core, and approximately four times as wide as channels located near the corners of the core. Consequently, typical selections are: $M_x = M_y \approx 2$ and $B_x = B_y \approx 1$.

Five channel edges are shown are Figure 1. Edge e^1 is characterized by $f_x(e_x^1) \approx B_x$ and $f_y(e_y^1) \approx B_y$. Edge e^2 is characterized by $f_x(e_x^2)\, f_y(e_y^2) \approx M_x\, M_y$. Edge e^3 is characterized by $f_x(e_x^3)\, f_y(e_y^3) \approx M_x\, B_y$ and edge e^5 is assigned approximately the minimum weight: $f_x(e_x^5)\, f_y(e_y^5) \approx B_x\, B_y$.

(3) The relative pin density of an edge. The *pin density* of a cell edge i, represented by d_p^i, is defined as the total number of pins belonging to edge i divided by the length of edge i. The *average pin density* for the entire circuit is represented by D_p. This is computed by dividing the total number of pins (for all of the cells) by the sum of the perimeters of all the cells. It is then possible to define the *relative pin density* of a cell edge i as d_{rp}^i, where

$$d_{rp}^i = \frac{d_p^i}{D_p}$$

The interconnect area modulation factor associated with the relative pin density of cell edge i has the following form:

$$f_{rp}(i) = \max\{\, 1.0\,,\, d_{rp}^i \}$$

I have defined $f_{rp}(i) \ge 1.0$ for all i, implying that at least some interconnect area needs to be assigned to a cell edge even if the edge has relatively few (if any) pins.

Note that factor (2) above must be updated each time a cell is moved during the simulated annealing process. The cpu time required to update this factor is minimal. Factors (1) and (3) can be determined at the outset and stored.

Each channel edge is bordered by two cell edges. Hence, it is approximated that the interconnect area associated with a cell edge is one half that computed for a channel edge with the same coordinate pairs. Let i now represent a cell edge, and further let e_w^i represent the estimated interconnect area associated with cell edge i. The values of x_i and y_i represent the positions of cell edge i. The complete expression for the interconnect area associated with a cell edge i is given by e_w^i:

$$e_w^i = 0.5\, \alpha\, C_w\, f_x(x_i)\, f_y(y_i)\, f_{rp}(i) \tag{2}$$

From Eqn. 1, the expected value of e_w^i should be $0.5C_W$. Under the assumption that $f_{rp}(i) \approx 1$, the normalization constant α is used to yield the required expected value:

$$\alpha = \frac{1}{HW} \int_{-H/2}^{H/2} \int_{-W/2}^{W/2} f_x(x)\, f_y(y)\, dx\, dy \tag{3}$$

If $M = M_x = M_y$ and $B = B_x = B_y$, then

$$\alpha = \left\{ \frac{M + B}{2} \right\}^2 \tag{4}$$

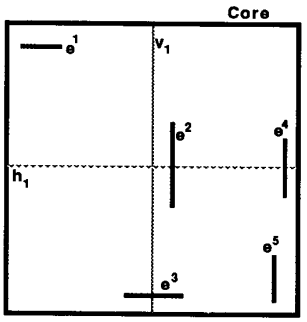

Figure 1.

In the simulated annealing approach used in TimberWolfMC, sufficient interconnect space between the cells is maintained by appending a border around the contours of each cell. The area occupied by each rectilinear cell is represented as a set of one or more non-overlapping rectangular tiles. Each time the amount of overlap (if any) is updated for two cells, the edges of each tile for a cell are expanded outward by an amount given by Eqn. 2.

The respective outward expansions of the tile edges are updated each time a cell participates in the generation of a new state in the simulated annealing approach. Hence the estimation of the interconnect area attributable to a given edge of a cell is a *dynamic* quantity, depending on the location of the edge at any given time. It follows that if a cell is moved from a corner towards the center of the core area, then the effective area of the cell will increase. Similarly, if a cell is moved from the center towards a corner of the core area, then the effective area of the cell will decrease.

2.3 Determining the Core Area

The determination of the desired (or, target) core area is difficult since the amount of wiring area needed cannot be determined prior to the completion of the simulated annealing placement algorithm. During the course of the simulated annealing algorithm, the cells will be discouraged from extending beyond the core boundaries.[16] If the target core area is excessive, the final aspect ratio generated by the placement algorithm may be quite far from the desired value. On the other hand, if the target core area is too small, either insufficient interconnection space will have been allotted between some of the cells, or cases of cell overlapping will be present at the conclusion of the algorithm.

TimberWolfMC uses the dynamic interconnect-area estimation algorithm to determine the expected interconnect area required for the chip. For the purposes of computing the initial core area, in which the positions of the cell edges are not known, Eqn. 2 is approximated by:

$$e_w^i = 0.5\, \alpha\, C_w\, f_x(0)\, f_y(0) \tag{5}$$

The modulation functions, f_x and f_y, are given their maximum values. The relative pin density for a cell edge, represented by f_{rp} in Eqn. 2, is assumed to have to be unity for all cell edges.

2.4 Custom-Cell Pin Placement

This subsection presents some details on pin placement with regard to custom cells.[17] Pins on custom cells may be specified in several possible ways: (1) A pin may be given a particular fixed location. (2) A pin may be assigned to a particular edge or edges of a cell. (3) A pin may belong to a group of pins which may be assigned to a particular edge or edges of a cell. Finally, (4) a pin may belong to a group of pins which is assigned a particular sequence ordering as well as a particular edge or edges of a cell. The placement of a single pin, a group of pins, or a sequence of pins may be specified as being restricted to either one cell edge, two cell edges, or any of the edges.

Changes to the placement of the *uncommitted* pins, that is, those specified according to (2), (3), and (4) above, occur throughout the course of the simulated annealing algorithm. All possible pin locations on the edges of a custom cell require storage for each of the eight possible orientations. Since the number of possible pin locations on a custom cell can easily number into the thousands, the amount of storage needed would be excessive. Also, during the earlier portion of the simulated annealing run, when T is relatively high, approximating the location of the pins is sufficient.

For these reasons, a strategy is used in which a specific number of pin *sites* are defined for each edge of a custom cell. The pin sites are approximately evenly spaced along each edge. Even though the locations of the sites are stored for each possible cell orientation, the amount of storage required is modest due to the limited number of pin sites per edge. Each pin site is assigned a *capacity*, in accordance with the number of pin locations encompassed by the site. During the course of the simulated annealing algorithm, a penalty function approach is used to discourage more than this number of pins from occupying a site.

3. The Algorithm for Stage 1 of TimberWolfMC

This section presents the algorithmic details for the first stage of TimberWolfMC. A simulated annealing algorithm is used to find a placement of the macro/custom cells such that sufficient interconnect area is allotted between the cells and such that the TEIC is minimized. If all of the net-weighting factors have a value of 1.0,

the total estimated interconnect length (TEIL) is identically equal to the TEIC. During this first stage, the interconnect area around the individual cells is determined using the dynamic interconnect-area estimator described in Section 2.2.

3.1 The Cost Function

The cost function for the simulated annealing algorithm of stage 1 consists of three independent terms.

3.1.1 The first term in the cost function: C_1

The first term is a function C_1 returning the TEIC. The Stage 1 algorithm seeks to minimize this term while driving the next two terms of the cost function to zero. The following terminology is used: (1) N_n is the total number of nets, (2) $x(n)$ returns the span of net n in the x, or horizontal direction, (3) $y(n)$ returns the span of net n in the y, or vertical direction, (4) $h(n)$ returns the weighting factor for net n for the horizontal direction, and (5) $v(n)$ returns the weighting factor for net n for the vertical direction. The function C_1 is then given by:

$$C_1 = \sum_{n=1}^{N_n} (x(n)\, h(n) + y(n)\, v(n)) \tag{6}$$

3.1.2 The second term in the cost function: C_2

The second term is the overlap penalty function C_2. The value returned by this function is given by:

$$C_2 = p_2 \sum_{i<j} O(i,j) \tag{7}$$

where p_2 is a normalization constant. The function $O(i, j)$ returns the total amount of overlap area between cells i and j. A rectilinear cell is stored as a union of non-overlapping rectangular *tiles*. Formally, the function $O(i, j)$ is given by:

$$O(i,j) = \sum_{t_i=1}^{T_i} \sum_{t_j=1}^{T_j} O_t(t_i, t_j) \tag{8}$$

where the rectangular tiles comprising the area of cell i are represented by $t_i \in \{1, 2, \cdots, T_i\}$ and similarly for the tiles comprising the area of cell j. The function $O_t(i,j)$ returns the common area between the two tiles specified as its arguments.

The normalization parameter p_2 is required since function C_1 scales linearly with the grid size of the net list data, while function C_2 is quadratic with respect to the grid size. The normalization is performed when T is at its maximum value, that is, for $T \rightarrow \infty$, designated as $T = T_\infty$. If the normalization is not performed and the initial average values of the function C_2 are substantially greater than the initial average values of the function C_1, then the simulated annealing algorithm will be concerned primarily with the minimization of the overlap penalty function. That is, relatively little attention will be paid to the minimization of the TEIC. On the other hand, if the initial average values of C_2 are substantially smaller than the average values of C_1, then the simulated annealing algorithm will be concerned primarily with the minimization of the TEIC. That is, relatively little attention will be paid to the elimination of the cell overlaps, until T is relatively small. At this point, most of the attention will then be focused on the removal of the infeasibilities, with the accompanying increase in the TEIC.

Nine industrial circuits were tested to determine the value of p_2 which would, on average, result in the lowest values of the final TEIC. The results showed that the lowest final values of the TEIC were obtained when

$$p_2\, C_2 = \eta\, C_1 \tag{9}$$

at the highest temperatures ($T = T_\infty$), where $\eta \approx 0.5$. The performance of TimberWolfMC was not very sensitive to the value of η. In fact, a degradation in average performance was not noted until η was reduced below 0.25 or beyond 1.0.

3.1.3 The third term in the cost function: C_3

The third part of the overall TimberWolfMC cost function, C_3, is a penalty function which serves to limit the number of pins in each pin site to the capacity of that site. The following notation is used: The custom cells are represented by j, where $j \in \{1, 2, \cdots, N_c\}$. For each custom cell j, the pin sites are represented by s_j, where $s_j \in \{1, 2, \cdots, N_s^j\}$. The function C_3 returns the contents of a given pin site, that is, the number of pins contained by the site. Further, the function C_3 returns the capacity of a given pin site.

The function $E(s_j)$ returns the penalty which is assigned to site s_j on custom cell j. A nonzero penalty is assigned to any site containing a number of pins which is greater than the specified site capacity. The expression for E is given below:

$$E(s_j) = \begin{cases} 0 & \text{if } C_t(s_j) \le C_p(s_j) \\ C_t(s_j) - C_p(s_j) + \kappa & \text{if } C_t(s_j) < C_p(s_j) \end{cases} \tag{10}$$

The constant κ serves to ensure that the number of pin sites containing more pins than specified by the capacity goes to zero just prior to the end of Stage 1 of TimberWolfMC. In the actual implementation, $\kappa = 5$. With reference to Eqn. 10, the penalty function C_3 is given by:

$$C_3 = \sum_{j=1}^{N_c} \sum_{s_j=1}^{N_s^j} (E(s_j))^2 \tag{11}$$

3.2 The Generation of New States

3.2.1 The *generate* function

Tests have corroborated the theoretical results which have shown that the initial state has no influence on the final value of the TEIC.[18] Hence it is common to begin with a random initial placement of the macro/custom cells. Each iteration of the inner loop initiates a call to the generation of new states function *generate*. The implementation of this function is now presented.

```
generate( ) {
    move_type = R_i(1, 2, p) ;
        /* R_i(1, 2, p) returns a random integer between 1 and 2, such that the
        probability of returning 1 is p.  r = p / (1-p) represents the ratio of single-
        cell displacements to pairwise interchanges. */
    if ( move_type == 1 ) {   /* Single cell displacement */
            i = R( 1, N_c ) ;  /* Nc is the number of cells */
            x = R( c_l, c_r ) ;
            y = R( c_b, c_t ) ;
              /* R(k,l) returns a random integer between k and l, with uniform dis-
              tribution.  A cell i is randomly selected for displacement to a ran-
              domly selected new location, represented by the coordinate pair (x,y)
              in the core area (bounded on the left by c_l, on the right by c_r, below
              by c_b, and above by c_t). The coordinate pair (x,y) represents the new
              location for the center of cell i. */
            if( A_1( i, x, y ) == YES ) {
                  /* Displacement of cell i to (x,y) was accepted. */
            } else if( A_1'( i, x, y ) == YES ) {
              /* Cell i is the focus of another new state attempt; this time its ori-
              entation is changed to effect an aspect ratio inversion. In Figure 2, if
              cell C_2 is displaced to location (x,y) with its current orientation, a
              great deal of overlap will result with the core boundary and with cell
              C_3. If the orientation of cell C_2 is changed such that its aspect ratio
              is inverted, no overlap will result in the new location. */
            } else {
                  /* The displacement failed for both aspect ratios; the next at-
                  tempt is a randomly-chosen orientation change of cell i. */
                  A_0( i ) ;
            }
            if( cell i is a custom cell ) {
                  for( k = 1 ; k ≤ P_U(i) ; k++ ) {
                        A_p(i) ;
                          /* A_p(i) returns the number of uncommitted pins
                          on cell i. This value dictates the number of times a
                          new pin group (or sequence) is selected for dis-
                          placement to a new set of pin sites. */
                  }
                  A_r(i) ;
                  /* Attempt an aspect ratio change, within specified bounds. */
            }
    } else {  /* Two cells will be selected for interchange. */
            j = R( 1, N_c ) ;
            if( A_2( i, j ) == YES ) {
              /* A_2 returns YES if the interchange of cells i and j was accepted. */
            } else {
                  A_2'( i, j ) ;
                  /* An interchange of cells i and j will be attempted again;
                  this time the orientations of the cells are each changed to
                  effect aspect ratio inversions. For example, in Figure 2,
                  if cells C_5 and C_7 are interchanged, a great deal of overlap
                  will result with the core boundary and with cells C_4 and
                  C_6. However, if the orientation of each cell is changed
                  such that its respective aspect ratio is inverted, no overlap
                  will result in their new locations. */
            }
    }
}
```

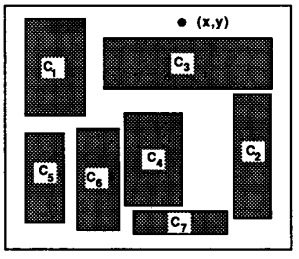

Figure 2

The ratio r of single cell displacements to cell interchanges can have an important effect on the final TEIL. The normalized final TEIL for a variety of circuits versus the ratio r is shown in Figure 3. The circuits contained an average of about 25 macro cells. Each execution of the *inner loop* consisted of $A_c = 200$ calls to the *generate* func-

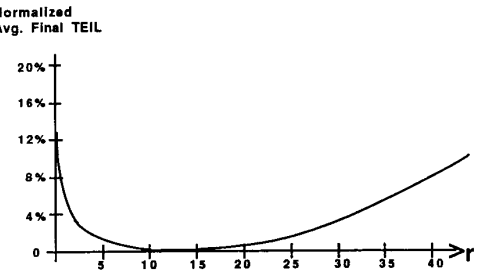

Figure 3

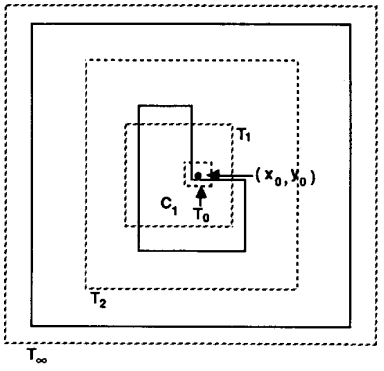

Figure 4

tion per cell. The initial value of $T = T_\infty$ was such that almost every new state was accepted. For each successive iteration of the *inner loop*, the new value of T was determined by: $T_{new} = \alpha\, T_{old}$, where $\alpha = 0.90$. Figure 3 indicates that values of r in the range from 7 through 15 yield TEIL values which are within one percent of the minimum. This wide range gives robustness to the algorithm.

3.2.2 The range limiter

The *generate* function is controlled by a *range limiter*. Large-distance moves usually imply large values of ΔC. At low temperatures, only moves which approximately satisfy $\Delta C \leq 0$ have a reasonable chance of being accepted. Hence, at low temperatures, the large-distance moves are almost invariably rejected. In order to generate moves which have a reasonable probability of acceptance, these large-distance moves are prohibited by the use of a range-limiter window. When a cell is selected for displacement, such as cell C_1 in Figure 4, the range-limiter window is centered at (x_0, y_0), corresponding to the center of C_1. The randomly-selected new location for C_1 must lie within the range-limiter window. In the current implementation of TimberWolfMC, cell interchanges are not controlled by the range-limiter window.

At the beginning of stage 1, when T is at its maximum value (indicated as T_∞ in Figure 4), the window extends beyond the core area, thereby allowing moves of maximum distance. Values of T are considered over a range of approximately six decades. As T is lowered (for example, to T_2 and subsequently to T_1), the window span in each direction is reduced as a function of the logarithm of T. In Figure 4, the span of the window is indicated for two intermediate values of T, namely T_1 and T_2. As T approaches the value $T = T_0$, the span of the window reaches its minimum value (as indicated in Figure 4) and this condition marks the end of stage 1 of TimberWolfMC.

Several range-limiter functions were tested, in which each function reduced the span of the window as a function of $\log_{10}(T)$.[19] In particular, tests were made for range-limiter functions of the following form, where $1 \leq \rho \leq 10$:

$$W_x(T) = W_x^\infty \left\{ \frac{\rho^{\log_{10}(T)}}{\lambda} \right\} \tag{12}$$

$$W_y(T) = W_y^\infty \left\{ \frac{\rho^{\log_{10}(T)}}{\lambda} \right\} \tag{13}$$

W_x^∞ represents the window span in the x-direction at $T = T_\infty$, that is, $W_x(T_\infty)$. Furthermore, W_y^∞ represents the window span in the y-direction at $T = T_\infty$, that is, $W_y(T_\infty)$. The value of λ was chosen such that for $T = T_\infty$, the term in the braces on the right hand sides of Eqns. 12 and 13 are normalized to 1.0. That is,

$$\lambda = \rho^{\log_{10}(T_\infty)}$$

A nominal value of T_∞ is 10^5.

The smallest values of the final TEIL were achieved with ρ in the range $1 \leq \rho \leq 4$. For this range, there was no observable difference in the performance with respect to the final TEIL. However, noticeably smaller values of the residual cell overlapping after the completion of the Stage 1 algorithm were achieved with the larger values of ρ. The residual cell overlapping is the value of the penalty function C_2 when $T \rightarrow T_0 \approx 0$. For a given value of T, as ρ increases, the window size is smaller. This implies that more local cell moves will be attempted, thus enhancing the probability of the removal of instances of cell overlapping. Consequently, the choice of $\rho = 4$ was made so as to obtain the lowest possible values of both the final TEIL and the residual cell overlapping.

3.2.3 Single-cell displacement-point selection

In the function *generate*, when a particular cell is selected for displacement, the new target location is a randomly-selected coordinate pair within the range-limiter window. The process of selecting the new target location is performed by the *displacement-point selection* function D_s. This function selects only from a very limited number of evenly-dispersed points lying within the range-limiter window. Let $W_x(T)$ and $W_y(T)$ represent the horizontal and vertical spans of the range limiter window, respectively. The value of $0.5W_x(T)$ defines the maximum distance in x that a cell's center may be moved in either direction. The function D_s restricts the step sizes in each direction to a set of 3 equally-separated amounts. The step sizes in each direction are multiples of s_x, defined as follows:

$$s_x = \frac{W_x(T)}{6} \tag{15}$$

$$s_y = \frac{W_y(T)}{6} \tag{16}$$

The only possible integers that can be multiplied by s_x (or s_y) to yield the actual step distance are contained in the set $I_x = I_y = \{-3, -2, -1, 0, 1, 2, 3\}$. Excluding the selection $I_x = 0$ and $I_y = 0$, there are 48 possible points within the range-limiter window which could be selected as the displacement points (that is, the points to which the center of the selected cell could be placed). As T approaches the value $T = T_0$, the span of the window reaches its minimum value of 6 units (in terms of the grid size inherent in specification of the cell geometry and pin locations in the net list) and this condition marks the end of stage 1 of TimberWolfMC. Thus, the minimum value of the step sizes s_x and s_y are one.

The function D_s has the following two characteristics: (1) With far fewer points to be considered, it is likely that a displacement of a cell to most of the 48 points could be considered for each value of T. (2) Much more emphasis is placed on the cells making large moves during large values of T and on the cells making comparatively smaller moves for smaller values of T. That is, small refinement moves are not attempted at high values of T, nor are large moves attempted at low values of T.

The function D_s was compared with a function D_r, which selects displacement points *randomly* from any of the points lying within the range-limiter window. The function D_s yielded only slightly better values for the final TEIL, however, the average residual cell overlapping after the completion of the Stage 1 was 22 percent lower when D_s was used.[19]

3.3 Additional Stage 1 Simulated Annealing Algorithmic Details

The *inner loop criterion* in TimberWolfMC is specified in terms of the number of attempted new states per cell per value of T, designated as A_c). The number of iterations of the inner loop for a given value of T is then given by:

$$A = A_c\, N_c \tag{17}$$

where N_c is the number of macro/custom cells. The effect of the inner loop criterion was noted for several circuits containing 30 to 60 macro cells. In Figure 5, a plot of the normalized average final TEIL versus A_c for these circuits is shown. Note that for circuits of this size, $A_c \approx 400$ is sufficient to yield the best results. In Figure 6, a plot of the relative final chip area (following TimberWolfMC global routing and placement refinement) versus A_c for these circuits is shown. Again, $A_c \approx 400$ is sufficient to yield the best results. Also note that reasonably good results can be obtained for substantially smaller values of A_c. The execution time for stage 1 of TimberWolfMC is directly proportional to A_c. The cpu time for a typical $A_c = 400$ run was about 4 hours on a VAX 8650. Note that for $A_c = 25$, the reduction in TEIL was 13 percent less than the best results. However, the cpu time in this latter case was 16 times less than that for the $A_c = 400$ tests. While the $A_c = 400$ case represents a substantial amount of cpu time, the additional 13 percent reduction in TEIL often yields an additional final chip area reduction of as much as 10 to 15 percent. From this standpoint, in the latter stages of the circuit

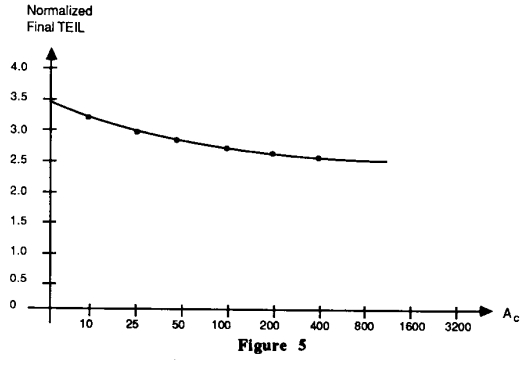

Figure 5

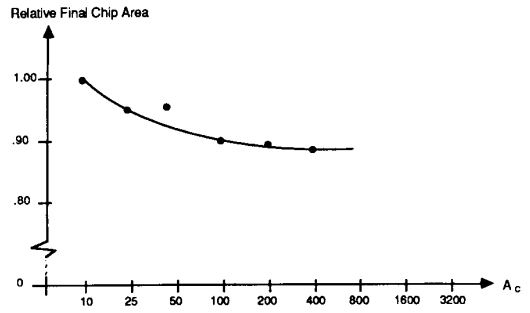

Figure 6

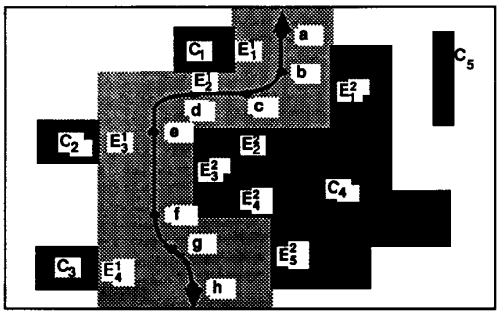

Figure 7

design process, the additional cpu time is usually justified. On the other hand, in the early stages of the design, a smaller value of A_c is more appropriate.

The function *update(T)* is expressed by

$$T_{new} = update(T_{old}) = T_{old} \cdot \alpha(T_{old}), \qquad 0 < \alpha(T_{old}) < 1 \qquad (18)$$

The cooling schedule used in TimberWolfMC was determined experimentally, with the following observations: (1) Three to five iterations are performed in which virtually every new state was accepted and where T is reduced rapidly from iteration to iteration. (2) After having left the high T regime, T was reduced such that the reduction in the average TEIC from one T step to another is approximately the same. (3) When the average value of the TEIC is only slightly reduced from one T step to another, the control of T enters the final regime where T is reduced more rapidly so that the value of the cost function firmly converges.

The initial value of $T = T_\infty$ is chosen such that virtually every new state proposed by *generate* is accepted. Tests on 9 industrial circuits showed that the average cell area $\bar{c}_a$, including the estimated interconnect area, was approximately proportional to the value of $T = T_\infty$ necessary to obtain an initial acceptance rate of nearly 100 percent. For a 25-cell circuit characterized by $\bar{c}_a = 10^4$, the smallest value of T yielding an initial acceptance rate of nearly 100 percent was $T_\infty = 10^5$. Therefore, to normalize for different circuit sizes and different grid sizes, the following expression is used:

$$T_\infty = T_\infty^* \frac{\bar{c}_a}{\bar{c}_a^*} \qquad (19)$$

where the selections of $\bar{c}_a = 10^4$ and $T_\infty^* = 10^5$ were made based on 25-cell industrial circuits. Defining

$$S_T = \frac{\bar{c}_a}{\bar{c}_a^*} \qquad (20)$$

it is then possible to rewrite Eqn. 19 as:

$$T_\infty = S_T T_\infty^* \qquad (21)$$

The value of S_T is a scale factor for the simulated annealing temperature profile, taking into account the circuit and grid sizes.

Table 1 contains the data for $\alpha(T_{old})$ as a function of T_{old} for which the best results were obtained with TimberWolfMC. The entries in the first column indicate the smallest value of $(S_T \cdot T_{old})$ for which T_{old} returns the corresponding entry in the second column. For example, for $T_{old} \geq (S_T \cdot 7000)$, $\alpha(T_{old})$ returns 0.85. It was arbitrarily determined that approximately 120 temperature values were to be considered in a typical execution of the simulated annealing algorithm. Based on extensive experimentation for 9 industrial circuits, the range of T from $(S_T \cdot 200)$ to $(S_T \cdot 7000)$ most strongly influenced the performance of TimberWolfMC. The *stopping criterion* is satisfied following an iteration of the *inner loop* in which the range-limiter window has achieved its minimum span.

For $T_{old} \geq$	$\alpha(T_{old})$
$(S_T \cdot 7000)$	0.85
$(S_T \cdot 200)$	0.92
$(S_T \cdot 10)$	0.85
0	0.80

Table 1

4. The Algorithms for Stage 2 of TimberWolfMC

The second stage of TimberWolfMC consists of several executions of a placement refinement algorithm. A placement refinement step is used to correct any (usually small) inaccuracies produced by the dynamic interconnect area estimator. That is, if insufficient space was allocated between a pair of cells, then additional space is provided as required. Or, if excessive space was allocated, then the cells are compacted as much as possible.

Each execution of the placement refinement algorithm consists of three steps: (1) a channel definition step, (2) a global routing step, and (3) a placement-refinement step. The information obtained in step 2 is used to compute the density of all of the channels, which then permits more accurate interconnect-area determination. The placement of the cells is then refined in step 3 to reflect the required interconnect area. A low-temperature simulated annealing algorithm is used. Three such placement-refinement steps have proven to be sufficient for the final TEIL and the final chip area to converge. The placement of the cells requires very little modification during detailed routing as a result of this strategy. This section presents the three steps of the placement-refinement algorithm.

4.1 Channel Generation

A new channel generation algorithm was developed for rectilinear cell layouts emphasizing the creation of channels which are more amenable to detailed monitoring of the local routing congestion.[20] This algorithm enables accurate determination of the space necessary between nearby cells. Traditional routing-channel generators have been designed only to facilitate detailed routing of the chip, and are less able to monitor local routing congestion.

Consider the routing channel defined by the dotted region in Figure 7. Suppose that a global router assigned some number of net segments to this channel, which is understood to follow the contour from a to h. With respect to this channel, the following observations are made: (1) Compaction (or expansion) of the channel is performed by moving the group of cells G_1 independently from the group of cells G_2, where $G_1 = \{ c_1, c_2, c_3 \}$ and $G_2 = \{ c_4, c_5 \}$. (2) The cell edges bordering the channel are labeled E_j^i, where i is the group (either 1 or 2) and where j identifies a particular cell edge in the group. Note that there are four cell edges bordering the channel which belong to group 1 and five cell edges bordering the channel which belong to group 2. (3) Note that defining the *channel density* (the usual congestion metric) is not possible since the channel does not have exactly two parallel, opposite sides. Hence it is not possible to specify a single parameter which gives the expected width of the channel.

A channel such as that of Figure 7 is suitable for detailed routing. However, several disadvantages arise if such a channel is present during the placement refinement phase. First, since there is no single parameter expressing a dimension to the channel, this channel must be analyzed in greater detail. That is, a multitude of parameters must be derived which specify the required spacing between edges: E_1^1 and E_1^2, E_1^1 and E_2^2, E_2^1 and E_3^2, E_3^1 etc. Second, if this additional analysis reveals that the spacing between cell edges E_1^1 and E_1^2 requires adjustment(either expansion or compaction), such a change may or may not subsequently necessitate an adjustment of the spacing between edges E_3^1 and E_3^2, and perhaps between edges E_4^1 and E_5^2 as well.

In summary, complex channels such as that represented in Figure 7 require additional *local* congestion analysis, well beyond a simple channel density calculation. Furthermore, responses to spacing changes may well trigger additional local analyses and/or spacing adjustments.

On the other hand, if a channel is bordered by only two cell edges, then it is possible to specify a single parameter which yields the expected width of a channel. Defining the *track spacing* t_s to be the minimum center-to-center spacing of the interconnect along the length of the channel, the expected width w (for two layers of interconnect) is given by:

$$w = (d + 2)\, t_s \qquad (22)$$

where d is the density of the channel. This expression is based on the fact that channel routers are currently available which routinely route a channel in a number of tracks t such that $t \leq (d + 1)$.[21] It is therefore possible to specify the required spacing between any two cell edges provided that every routing channel is bordered by exactly two cell edges, or bordered by one cell edge and the border of the chip.

Consequently, in my new channel definition algorithm a channel, or *critical region*, is created between every pair of parallel cell edges belonging to different cells such that: (1) The span of the two edges overlap in one (of the two) dimensions, in which a rectangular region of *empty space* is bounded on two sides by these two edges. The extent of the *empty space* region equals the common span of the two edges. (2) No other cell edges intersect the rectangular, *empty space* region between the two edges.

In Figure 8, the placement of five cells, C_1 through C_5, is shown. Note that C_4 is a rectilinear cell having 12 edges. The cross-hatched regions in this figure indicate the *critical regions*, that is, those rectangular regions bordered by exactly two cell (or core boundary) edges. The distance between the two edges determines the thickness (or capacity) of the vertical or horizontal channel. Each of the *empty space* regions in Figure 8, labeled r_i for $i \in \{1, 2, \cdots, 20\}$ is a node in the channel graph. In Figure 9, the edges for the final channel graph are shown as dashed lines. The channel graph nodes are represented by n_i where $i \in \{1, 2, \cdots, 20\}$.

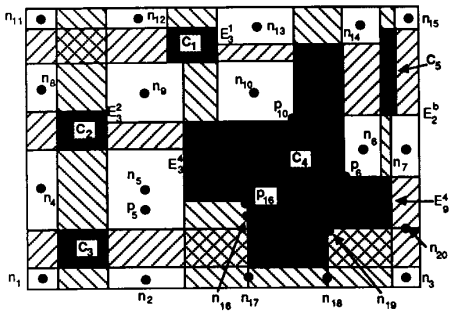

Figure 8

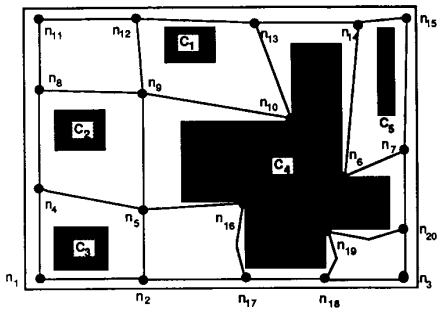

Figure 9

The pins on each edge of each cell are then mapped onto the corresponding adjacent channel edge. This mapping is done by projecting the pin location perpendicular to the cell edge until intersecting the corresponding channel edge. In Figure 9, pin P_1 on cell C_2 is projected to location P_1' on the channel edge defined by nodes n_4 and n_5. Similarly, pin P_0 on cell C_4 is projected to location P_0' on the channel edge defined by nodes n_5 and n_9. The channel graph nodes are positioned (for example, node n_{10} in Figure 11) such that every pin on a cell edge is projected to lie between the two nodes of the corresponding channel edge.

The concept of the *bottleneck*, introduced by N.P. Chen, is quite similar to the definition of a critical region.[22] My new algorithm is somewhat more general in that every possible critical region is identified, and used, during the placement refinement step. Chen's method assumes that two critical regions, or bottlenecks, will not overlap. Overlapping can occur if a critical region is created both by a pair of vertical cell edges and a pair of horizontal cell edges. An example of this exists in the upper left corner of Figure 9, where nodes n_8, n_9, n_{11}, and n_{12} enclose two overlapping critical regions. In Chen's method, either the placement is modified to remove the overlapping critical regions, or one of the critical regions is ignored. The ignored critical region is that which is the widest, based on the estimate that this critical region will probably be less congested after global routing. Since global routing has not been performed yet, this can only be an estimate. In my new algorithm, all of the critical regions are identified and used.

4.2 Global Routing

A good review of global routing strategies has been published.[23] A new general-purpose global routing algorithm which avoids the classical routing-order dependence problem was also developed.[24] The global router is independent of the layout style since the only inputs to the algorithm are a net list and a channel graph (such as that generated by the algorithm of Section 4.1). In the input to the global router, each pin in the net list has been assigned to a specific position on a channel edge in the graph, including electrically equivalent pins. The global router makes full use of equivalent pins to minimize the routing length of a net. The global router minimizes the sum of the routing lengths of all of the nets subject to the satisfaction of the *capacity constraints* of the edges. The constraints result from the fixed widths of the channel edges.

4.2.1 Phase one of the global router

The new global router algorithm has two basic phases. The first phase generates (and then stores) M alternative routes for *each* net. The parameter M is typically on the order of 20 or more. The algorithm attempts to find the M *shortest* routes for each net, a task which can be accomplished readily for two-pin nets using Lawler's algorithm.[25] This is an efficient algorithm for finding the M-shortest paths between two vertices on a graph.

For nets consisting of more than two pins, I developed an algorithm which generalizes Lawler's approach. This algorithm finds the approximately M-shortest paths on a graph for an n-pin ($n \geq 2$) net, with the ability to handle electrically-equivalent pins. Results have shown that for nearly all nets of fewer than 20 pins, this new algorithm appears to find the minimal Steiner length route among the M alternatives.

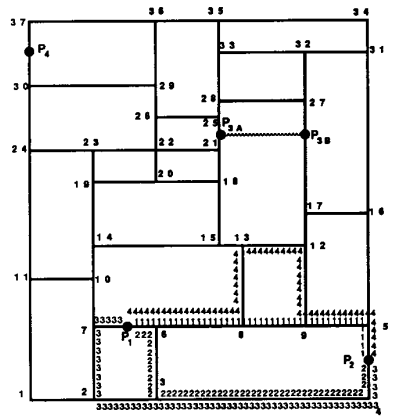

Figure 10

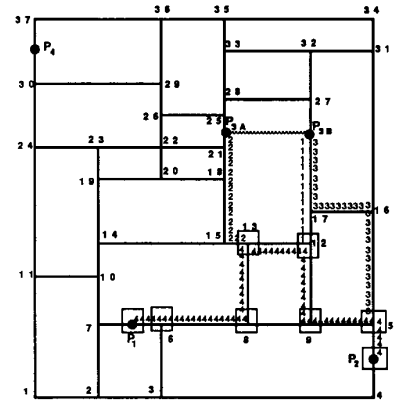

Figure 11

The phase one algorithm will be explained using Figures 10, 11, and 12. In Figure 10, there is a five-pin net to be globally routed on the indicated graph. There are only four distinct pin groups, as pins P_{3A} and P_{3B} are electrically equivalent. The algorithm arbitrarily selects a starting pin and then interconnects subsequent pins in an order essentially determined by following Prim's algorithm for finding a minimum spanning tree on a graph.[26] There is one main difference, however. Each time a pin is to be added to the list of interconnected pins, the algorithm generates and stores the M-shortest paths connecting the new pin with the existing interconnected pins. Referring to Figure 10, P_2 was selected as the starting pin. The first step of the recursive process selects P_1 as the nearest next pin.[27] The M-shortest paths between nodes P_1 and P_2 are generated and stored. In the figure, the four shortest such paths are indicated.

The algorithm then sequentially selects one of the stored paths and recursively interconnects an additional pin. For example, in Figure 11, the algorithm has proceeded to where path number 4 is under consideration. At this level of recursion, nodes P_1, P_2, 6, 8, 13, 12, 9, and 5 are all target nodes. The next pin to be interconnected is either of the electrically equivalent pair P_{3A} and P_{3B}. The M-shortest paths between the target nodes and nodes P_{3A} or P_{3B} are generated and stored. In the figure, the three shortest such paths are indicated.

The algorithm again selects one of the stored paths and recursively interconnects pin P_4. In Figure 12, the recursion has proceeded to where path 2 of the previous step is being considered. At this point, nodes P_1, P_2, 6, 8, 13, 12, 9, 5, P_{3A}, P_{3B}, 21, 18, and 15 are all target nodes. The M-shortest paths between the target nodes and node P_4 are generated and stored. In the figure, paths 1, 2, and 5 are shown. Each time a connection has been made to the final pin, the completed route is stored along with its length. The overall M-shortest such routes are retained for consideration in phase two.

4.2.2 Phase two of the global router

The second phase of the algorithm selects a single route from among the $M_i \leq M$ alternatives for each net i, where $i \in \{1, 2, \cdots, N\}$ and N is the number of nets. Note that M_i is specified as $M_i \leq M$ since cases arise in which fewer than M alternative routes are possible for a given net. Let i_k represent the k-th alternative route for net i, where $k \in \{1, 2, \cdots, M_i\}$. A random interchange algorithm is then used to select alternative i_k for each net while seeking to minimize the total routing length subject to the channel-edge capacity constraints. This approach enables the global router to avoid the net-routing-order dependence problem.

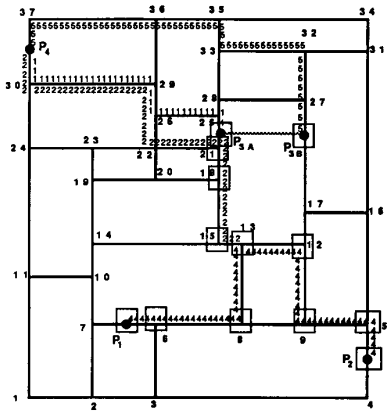

Figure 12

This section describes the cost and *generate* functions for the random interchange algorithm. The cost function is represented by:

$$L = \sum_{i=1}^{N} \lambda(i_k) \tag{23}$$

where $\lambda(i_k)$ represents the length of alternative route i_k.

Let E_j represent a channel edge, where $j \in \{1, 2, , N_{ch}\}$ and where N_{ch} represents the total number of channel edges for a circuit. The density of channel edge j is represented by D_j and the capacity of the edge is represented by C_j.

A second cost function is defined by:

$$X = \sum_{j=1}^{N_{ch}} \{ (D_j - C_j) \mid D_j > C_j \} \tag{24}$$

X represents the total number of excess tracks required over all channel edges.

The M_i possible routes for each net i are enumerated such that the $k = 1$ route is the shortest. That is, i_1 is the shortest route for each net i. The first state of the random interchange algorithm is such that $k = 1$ for each net i. If this first state is such that no capacity-constraint violations are observed, that is, if there does not exist $j \in \{1, 2, \cdots, N_{ch}\}$ such that $D_j > C_j$, then the algorithm terminates since each net has been given its shortest route with no capacity violations. If capacity-constraint violations are observed, the random interchange algorithm visits other states while attempting to eliminate the excess congestion.

New state generation begins with the random selection of a channel edge E_j characterized by $D_j > C_j$. A net i containing a segment using E_j is then randomly selected. An alternative route for net i is selected randomly from among those characterized by $\Delta X \leq 0$. If $\Delta X = 0$ and $\Delta L \leq 0$, or if $\Delta X < 0$, then the new route is accepted; otherwise it is rejected. The *stopping criterion* is satisfied when either: (1) each net has been assigned the k=1 route and X=0, or (2) the values of L and X have not changed for $M \cdot N$ new state attempts.

4.3 Placement Refinement

A low-temperature simulated annealing algorithm is used to accomplish this step. The placement of the cells is refined to reflect the required interconnect area, as determined by the channel definition and global routing steps. These steps determined the required interconnect area of every channel. Since exactly two cell edges border each channel, the spacing requirement between the two cell edges is immediately established. One half of the interconnect space is associated with each of the two cell edges. That is, each respective edge is expanded outward (from the interior of the cell) by an amount equal to one-half of the required channel width. These expanded cell edges are used each time the amount of overlap (if any) is updated for two cells. Note that in contrast to the Stage 1 dynamic interconnect-area estimator, the amount of outward expansion of the cell edges is a static quantity for each execution of the three placement refinement steps.

The same objective function as in stage 1 of TimberWolfMC, described in Section 3.1, is used. The function *generate* has a much simpler form for Stage 2. New states can only be generated by attempting single cell displacements and by alterations of the pin placement. The orientations of the cells are not changed during Stage 2. The aspect ratios of custom cells also remain fixed during the second stage. These latter two methods of new state generation tend to invalidate the calculation of the interconnect area associated with a particular cell edge. More placement refinement steps are required in order to enable the possibility of a convergence in the final chip area and the final TEIL. Furthermore, even if convergence is achieved, the final values of the chip area and TEIL tend not to be better than the case in which orientation and aspect-ratio changes are not permitted.

Since the cell moves are very local in Stage 2, the range-limiter window must have an initial size which is some fraction μ of the span of the core area. Equation 12 dictates the control of the window size for the x direction. It is desired that:

$$\mu = \frac{W_x(T)}{W_x^{\infty}} \tag{25}$$

The initial value of T for Stage 2 is selected which satisfies Eqn. 25. That is, a solution for T' is desired from the following expression:

$$\mu = \frac{4^{\log_{10}(T')}}{\lambda} \tag{26}$$

where λ is defined by Eqn. 14. That is,

$$\mu = \frac{4^{\log_{10}(T')}}{4^{\log_{10}(T_\infty)}} \tag{27}$$

Solving this expression for T' then yields:

$$T' = \mu^{\log_4 10} T_\infty \tag{28}$$

In TimberWolfMC, $\mu = 0.03$ is used. This implies that the cells are initially permitted to move 3 percent of the span of the core in any direction. Based on tests for 9 industrial circuits, on the average it was found that larger values of μ yielded equally good final values of the TEIL, but not better. Of course, the cpu time required to achieve a convergence in the TEIL is larger for larger values of μ. The value of μ had to be reduced somewhat below 0.03 before the average performance showed a degradation.

The temperature is updated by the function *update* expressed by Eqn. 18. Table 2 contains the data for $\alpha(T_{old})$ as a function of T_{old} which is used during Stage 2. The parameter S_T is defined by Eqn. 20.

For $T_{old} \geq$	$\alpha(T_{old})$
$(S_T \cdot 10)$	0.82
0	0.70

Table 2

For the first two iterations of the three placement-refinement steps, the stopping criterion is satisfied once an iteration of the *inner loop* has been performed with the range-limiter window at its minimum span. For the third (final) iteration of the placement-refinement step, the *stopping criterion* is satisfied if the value of the cost function is unchanged for 3 consecutive iterations of the *inner loop*.

5. Results

5.1 Dynamic Interconnect Area Estimator Results

In order to determine the accuracy of the dynamic interconnect area estimator, the final TEIL and the final core area for 9 industrial circuits were compared at the end of each of the two stages. A large change in the final TEIL or in the core area would indicate substantial cell movement during the second stage of TimberWolfMC, thereby implying inaccurate interconnect area estimation. The results for the 9 circuits are shown in Table 3. The last two columns represent, respectively, the TEIL and the core area at the end of stage 2 divided by the values obtained at the end of stage 1, expressed in terms of a percentage change. For the 9 circuits tested, on average there was negligible change in core area between the two stages of TimberWolfMC. Furthermore, there was a minimal decrease in TEIL after the second stage.

Circuit	No. Cells	No. Nets	No. Pins	No. Trials	Avg. TEIL Red. (%)	Avg. Area Red. (%)
i1	33	121	452	5	5.8	3.0
p1	11	83	309	6	2.0	-9.2
x1	10	267	762	4	4.0	2.5
i2	23	127	577	5	-1.0	-3.8
i3	18	38	102	2	10.5	-0.5
l1	62	570	4309	4	2.5	-0.5
d2	20	656	1776	4	12.7	8.5
d1	17	288	837	4	0.5	8.25
d3	17	136	665	2	0.5	-1.0
Avg.					4.4	4.1

Table 3

5.2 TimberWolfMC Results

The performance of TimberWolfMC was compared versus various industrial, university, and manual placement methods. The two criteria were the TEIL and the final chip area. The results for 9 industrial circuits of various sizes are summarized in Table 4.

Circuit	No. Cells	No. Nets	No. Pins	TEIL	Area ($x \times y$)	TEIL Red. (%)	Area Red. (%)
i1	33	121	452	7431	236 × 223	26	14
p1	11	83	309	12306	293 × 294	8	18
x1	10	267	762	60326	875 × 744	11	15
i2	23	127	577	121386	2873 × 2751	49	†
i3	18	38	102	7043	644 × 699	46	56
l1	62	570	4309	254063	1084 × 1042	19	50
d2	20	656	1776	419608	1355 × 1433	13	4
d1	17	288	837	37365	245 × 305	23	†
d3	17	136	665	325457	3398 × 3298	29	31
Avg.						24.9	26.9

† comparative result was not available

Table 4

The 26 percent reduction in TEIL and the 14 percent chip area reduction achieved for circuit i1 was in comparison to a placement method based on resistive network optimization.[28] Circuits i2 and i3 were in comparison to the CIPAR placement and routing package developed by Gould-AMI. The CIPAR router was not able to route its own placement for circuit i2, hence an area comparison was not possible.

The 19 percent total interconnect length reduction and the 50 percent chip area reduction was in comparison to a manual layout performed at Intel Corp. Circuit p1 was in comparison to a manual layout at Hewlett-Packard. Further, circuits d1, d2, and d3 were in comparison to manual layouts at Advanced Micro Devices. For circuits l1, p1, d1, d2, and d3, the widths of the power and ground lines were not supplied. It was assumed that, on average, the widths of these lines were about twice a normal wire width, and that these lines were present in every channel. The cpu time ranged from 15 minutes for the smallest circuits to 4 hours for the largest circuits on a DEC MicroVAX II.

6. Conclusion

A macro/custom cell chip-planning, placement, and global routing package based on the simulated annealing algorithm has been developed. A new algorithm for accurately estimating the interconnect area around the individual cells is used in TimberWolfMC. This dynamic interconnect-area estimation algorithm has resulted in the generation of placements which require virtually no placement modification during or after detailed routing.

The placement algorithm proceeds in two distinct stages. During the first stage, the interconnect area around the individual cells is determined using the dynamic interconnect area estimator. A simulated annealing algorithm is used to minimize the TEIC. The second stage of TimberWolfMC consists of three executions of a placement refinement algorithm. Each execution consists of three steps: (1) a channel definition step, in which a new channel definition algorithm is used, (2) a global routing step, using a new global routing algorithm, and (3) a placement refinement step.

The performance of the package was compared for 9 industrial circuits versus a variety of other placement algorithms. The average total interconnect length reduction was over 24 percent and the average chip area reduction was nearly 27 percent. Furthermore, for the 9 examples, circuit-area reductions ranged from 4 to 56 percent versus a variety of other placement algorithms.

7. Acknowledgements

I am very grateful for the financial support provided by Texas Instruments, Inc., Digital Equipment Corp., and Lawrence Livermore National Laboratory. AMD, Intel, Hewlett Packard, Gould-AMI, and H. Chen graciously supplied test cases. I would like to thank Prof. Alberto Sangiovanni-Vincentelli for his encouragement and support.

8. References

1. C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *Proc. 1984 Custom Integrated Circuits Conference*, Rochester, NY, May 1984.

2. C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE J. of Solid-State Circuits*, vol 20, n. 2, April 1985, p. 510.

3. C. Sechen and A. Sangiovanni-Vincentelli, "TimberWolf3.2: A New Standard Cell Placement and Global Routing Package," *Proc. 1986 Design Automation Conference*, Las Vegas, NV, June 29 - July 2, 1986, p. 432.

4. C. Sechen, D. Braun, and A. Sangiovanni-Vincentelli, "ThunderBird: A Complete Standard Cell Layout Package," *IEEE J. of Solid-State Circuits*, April 1988.

5. C. Sechen and K. W. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement," *Proc. 1987 Int. Conf. on Comp. Aided Design*, Nov. 9-12, 1987, Santa Clara, CA.

6. D. Jepsen and C. Gelatt, "Macro Placement by Monte Carlo Annealing," *Proc. Int. Conf. on Comp. Aided Design*, Sept. 1983, Santa Clara, CA, p. 495.

7. R. Otten and L. van Ginneken, "Floorplan Design using Simulated Annealing," *Proc. Int. Conf. on Comp. Aided Design*, November 1984, Santa Clara, CA, p. 96.

8. D. Wong and C. Liu, "A New Algorithm for Floorplan Design," *Proc. 23rd Design Automation Conference*, Las Vegas, NV, p. 101.

9. S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, n. 4598, May 13, 1983, p. 671.

10. J. Burns and R. Newton, "SPARCS: A New Constraint-Based IC Symbolic Layout Spacer," *Proc. 1986 Custom Integrated Circuits Conference*, p. 534.

11. A. El Gamal and Z. Syed, "A Stochastic Model for Interconnections in Custom Integrated Circuits," *IEEE Transactions on Circuits and Systems*, v. 28, September 1981.

12. W. Heller, W. Mikhail, and W. Donath, "Prediction of Wiring Space Requirements for LSI," *J. Design Automation and Fault-Tolerant Computing*, 1978, p. 117.

13. C. Sechen, *Placement and Global Routing of Integrated Circuits Using Simulated Annealing*, Ph.D. Dissertation, U. of California, Berkeley, 1987, Chapter 6.

14. A complete derivation can be found in: C. Sechen, *Placement and Global Routing of Integrated Circuits Using Simulated Annealing*, Ph.D. Dissertation, U. of California, Berkeley, 1987, Chapter 5.

15. C. Sechen, "Average Interconnection Length Estimation for Random and Optimized Placement," *Proc. 1987 Int. Conf. on Comp.-Aided Design*, Nov. 9-12, Santa Clara, CA.

16. This is done by placing four dummy cells, one bordering each side of the core, which (in effect) extend infinitely outward. If a macro/custom cell edge extends beyond a core boundary, the cell would then overlap with the dummy cell, leading to an increase in the value of the overlap penalty function.

17. Recall that pins on macro cells have fixed locations.

18. For example: Mitra, Romeo, and Sangiovanni-Vincentelli, "Convergence and Finite-Time Behavior of Simulated Annealing," *Proc. 24th Conf. on Decision and Control*, Ft. Lauderdale, FL, December 1985.

19. C. Sechen, *Placement and Global Routing of Integrated Circuits Using Simulated Annealing*, Ph.D. Dissertation, U. of California, Berkeley, 1987, Chapter 4.

20. C. Sechen, *Placement and Global Routing of Integrated Circuits Using Simulated Annealing*, Ph.D. Dissertation, U. of California, Berkeley, 1987, Chapter 7.

21. Reed, Sangiovanni-Vincentelli, and Santamauro, "A New Symbolic Channel Router: YACR2," *IEEE Trans. on CAD*, v. 4, July 1985, p. 208.

22. N. P. Chen, *Routing System for Building Block Layout*, Ph.D. Dissertation, Dept. of EECS, U. of California, Berkeley, 1983.

23. E. S. Kuh and M. Marek-Sadowska, "Global Routing," in *Advances in CAD for VLSI*, v. 4, North Holland, 1986.

24. C. Sechen, *Placement and Global Routing of Integrated Circuits Using Simulated Annealing*, Ph.D. Dissertation, U. of California, Berkeley, 1987, Chapter 8.

25. E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, 1976, p. 102-104.

26. R. Prim, "Shortest Connecting Networks and Some Generalizations," *Bell System Technical Journal*, v. 36, 1957, pp. 1389-1401.

27. In its most general form, the algorithm considers not only the closest unconnected pin, but up to k additional unconnected pins as well, where k is a parameter. For more information, refer to: C. Sechen, *Placement and Global Routing of Integrated Circuits Using Simulated Annealing*, Ph.D. Dissertation, U. of California, Berkeley, 1987, Chapter 8.

28. C. K. Cheng and E. Kuh, "Module Placement Based on Resistive Network Optimization," *IEEE Trans. on CAD*, v. 3, n. 3, July 1984, p. 218.