The TimberWolf Placement and Routing Package

CARL SECHEN AND ALBERTO SANGIOVANNI-VINCENTELLI, FELLOW, IEEE

Abstract—**TimberWolf is an integrated set of placement and routing optimization programs. The general combinatorial optimization technique known as simulated annealing is used by each program. Programs for standard cell, macro/custom cell, and gate-array placement, as well as standard cell global routing have been developed. Experimental results on industrial circuits show that area savings over existing layout programs ranging from 15 to 62 percent are possible.**

I. INTRODUCTION

TIMBERWOLF is an integrated set of placement and routing optimization programs. Extensions and modifications of the general combinatorial optimization technique known as simulated annealing [1] are used by each program. Four basic optimization programs of the TimberWolf package have been developed.

1) A Standard-Cell Placement Program: This program places standard cells into rows and/or columns in addition to allowing user-specified macro blocks and pads. The program was interfaced to the CIPAR standard cell placement package developed by American Microsystems, Inc. For the largest circuits tested (800 to 2700 cells), TimberWolf reduced total estimated wire lengths by 45 to 66 percent in comparison with CIPAR alone. Furthermore, final chip areas were reduced by 30 to 57 percent as a result of the improved placement. For a circuit of 1000 cells, TimberWolf reduced the final chip area by 31 percent in comparison to CIPAR and by 21 percent over another commercially available standard cell placement program in a benchmark performed at AMI.

2) A Standard Cell Global Router Program: The global router reduced by 10 to 15 percent the number of wiring tracks used by the CIPAR router. This translated to an overall area savings for 6 to 8 percent. Vecchi and Kirkpatrick [2] recently described the use of simulated annealing for global routing.

3) A Macro/Custom Cell Placement Program: This program places cells of any rectilinear shape. Furthermore, the cells may have fixed geometry including pin locations (macro cells) or they may have fixed area with a given aspect ratio range and with pins that need to be placed (custom cells). All rotations and reflections of each cell are considered. TimberWolf also has the ability to place cells among user-defined subregions of the chip. TimberWolf allows multiple chips to be placed simultaneously. This package can also be used to place circuits on one or more printed circuit boards.

The macro/custom cell placement program is currently under test on industrial circuits. However, the program has been tested on a Honeywell Information Systems Italy printed circuit board. The processor board required the placement of 613 variable-sized circuits. TimberWolf reduced the total wire length by 10 percent over the manually placed board.

4) A Generalized Gate-Array Placement Program: This program allows user-specified macros and primary terminals. This program found placement with a 6- to 27-percent reduction in total estimated wire length for several benchmark problems in comparison to the best published results. This program optionally includes in the cost calculation a measure of the local routing congestion.

This paper presents the algorithms used by each of the programs comprising the TimberWolf package and also presents the results that have been obtained. In particular, Section II describes the basic algorithm. In Section III, the standard cell placement optimization algorithm and program are described. Section IV presents details on the standard cell global router. In Section V, the macro/custom placement optimization algorithm and program are described and in Section VI the gate-array placement algorithm and implementation are presented. Finally, Section VII is devoted to concluding remarks and future research.

II. THE BASIC ALGORITHM

Simulated annealing has been proposed by Kirkpatrick *et al.* [1] as an effective method for the determination of global minima of combinatorial optimization problems involving many degrees of freedom. Its basic feature is the possibility of exploring the configuration space of the optimization problem allowing *hill climbing* moves, i.e., the acceptance of new configurations of the problem which increase the cost. These moves are controlled by a parameter, in analogy with temperature in the annealing process, and are less and less likely towards the end of the process.

Manuscript received August 31, 1984; revised December 18, 1984. The algorithmic part of this research has been supported by DARPA under Grant N00039-83-C-0107. The TimberWolf placement and routing package has been supported by a Grant from the MICRO of the State of California.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

0018-9200/85/0400-0510$01.00 ©1985 IEEE

Given a combinatorial optimization problem specified by a finite set of configurations or states S and by a cost function c defined on all the states j in S, the simulated annealing algorithm is characterized by a rule to generate randomly a new state or configuration with a certain probability, and by a random acceptance rule according to which the new configuration is accepted or rejected. A parameter T controls the acceptance rule.

The basic structure of the algorithm is presented in the next subsection. Theoretical investigations of the simulated annealing optimization technique have been reported by our research group [3] and elsewhere [4], [5].

A. Algorithm Structure

The following function gives the general structure of the class of algorithms called *probabilistic hill-climbing algorithms* of which simulated annealing is a special case. This class has been proposed in [3] where a number of different algorithms with the same structure have been introduced.

```
Algorithm Structure (j0, T0){
  /*
  * Given an initial state j0 and an
  * initial value for the parameter T,
  * T0,
  */
  T = T0;
  X = j0;
    while("stopping criterion" is not satisfied){
        while("inner loop criterion" is not satisified){
            j = generate(X);
            /*
            * generate is a function which
            * returns a new state j generated
            * incrementally from the previous state
            * X by a weighted random selection.
            */
            if (accept(c(j), c(X), T){
            X = j;
            }
        }
        T = update(T);
    }
}
```

The acceptance of a new state j is determined by ***accept,*** whose structure is shown below.

```
accept(c(j), c(i), T){
            /*
            * Returns 1 if the cost variation passes a test
            * T is the control parameter
            */
            Δc = c(j) − c(i);
            y = f(Δc, T);
            r = random(0,1);
                /*
                * random is a function which returns a
                * pseudo-random number uniformly
                * distributed on the interval [0,1]
                */
            if(r < y){
              return(1);
            } else {
              return(0);
            }
        }
}
```

The algorithms in the class described above are 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. In the original version of simulated annealing, the acceptance function is governed by the function f shown below

$$f(\Delta c, T) = \min[1, \exp(-\Delta c/T)].$$

It is possible to vary the shape of f by adjusting the control parameter T, called temperature. The updating rule for T is given below.

$$T_{\text{new}} = \alpha(T_{\text{old}}) * T_{\text{old}}, \qquad 0 < \alpha < 1.$$

In the function **accept**, note that new states characterized by $\Delta c \leq 0$ always satisfy the acceptance criterion. However, for the new states characterized by $\Delta c > 0$, the parameter T plays a fundamental role. If T is very large, then r is likely to be less than y and a new state is almost always accepted irrespective of Δc. If T is small, close to 0, then only new states that are characterized by very small $\Delta c > 0$ have any chance of being accepted. In general, all states with $\Delta c > 0$ have smaller chances of satisfying the test for smaller values of T.

The properties of this class of algorithms can be studied using Markov chains as the theoretical models. Theoretical analysis [3] shows that this class generates with probability 1.0 the global optimum of the optimization problem, provided that certain conditions on the number of iterations at each T or a certain updating rule for T is followed. These results are unfortunately asymptotic and provide little information on how to choose the various parameters for the implementation of the algorithm. However, they serve to give confidence in the well posedness of the algorithm and to provide some insight on the reasons why simulated annealing has performed well in practical cases. In the remainder of the paper, attention will be given to the actual implementation of the various functions, the inner loop criterion, and the stopping criterion.

The best results with simulated annealing have been obtained in our experiments by starting with a large value of the parameter T, whereby virtually all proposed new states are accepted. Further, the best results have been obtained when the system is allowed to achieve "equilibrium" at each stage (or value of T) of the annealing process. That is, a sufficient number of iterations are performed in the inner loop such that the probability

distribution of the configuration is "close" to the stationary probability distribution of the Markov chains associated with the algorithm (see [3] for more details on the theory of simulated annealing). This is implemented by the "inner loop criterion" in the simulated annealing algorithm. The "stopping criterion" is satisfied when the cost function's value remains the same after several stages of the annealing process.

In simulated annealing, the best results have been obtained when the parameter T is slowly reduced when the cost function's value begins to decrease significantly. For each successive step of the annealing process, T is lowered exponentially. The TimberWolf programs currently allow the value of α to be specified for each value of T. The value of α is usually in the range of 0.8–0.95.

B. The TimberWolf Implementation of the Simulated Annealing Algorithm

For the applications of interest here, little difference was noted when using different functions f in the acceptance function **accept**. Hence the standard form for f as proposed by [1] was used. This section presents the TimberWolf implementations of the other functions.

1) Generating New States: The TimberWolf programs begin with a random initial placement or wiring configuration. A new state is generated by either exchanging two fundamental units or moving a unit to another location. For the gate array placement program, the new state is generated by the interchange of two *modules*, where a module refers to a fundamental unit specified in the net list. The standard cell placement program also generates new states by the interchange of cells. However, because standard cells typically vary in width, the interchange of two cells often results in a non-feasible solution because overlaps are not allowed. This is solved by a penalty function approach, first described by Kirkpatrick, Gelatt, and Vecchi [1]. The TimberWolf implementation of this approach will be described in the next section. The penalty function approach is also employed by the macro/custom cell placement program because the cells typically vary in both height and width.

For the standard cell and macro/custom cell problems, new states are also generated by the movement of a cell to a new location. Experimental investigation has revealed that the use of both methods of generating new states is necessary to achieve the best results. Furthermore, orientation changes of standard and macro/custom cells are performed which result in new states. If allowed by the user, new states are also generated for custom cells by assigning a new location to a pin or group of pins and by changing the aspect ratio of the cell.

For the standard cell global router program, new states are generated by assigning a portion of a net to a different channel.

2) Cost Function: The cost function for the placement programs is based on total estimated wire length. The standard cell and macro/custom cell programs also include a penalty function term which penalizes overlaps of the cells. The cost function for the standard cell global router is based on the estimated wiring area which is approximated by the total channel density, that is, the sum over all channels of the channel density.

3) Generating New Values of T: In the current implementation of TimberWolf, the parameter α is user-specified as α versus T data. The best results have been obtained when α is the largest (approximately 0.95) during the stages of the algorithm when the cost function is decreasing rapidly. Furthermore, the value of α is given its lowest values at the initial and latter stages of the algorithm (usually 0.80). The value of α is gradually increased from its lowest value to its highest value, and then gradually decreased back to its lowest value.

4) The Inner Loop Criterion: The inner loop criterion is implemented by the specification of the number of new states generated for each stage of the annealing process. This number is specified as a multiple of the number of fundamental units for the placement or routing problem. For the gate array placement and standard cell global router programs, 20 new states per unit are generated at each stage. The standard cell and macro/custom cell placement problems have many more degrees of freedom (orientation changes, pin location changes, etc.) and hence 100 or more new states are generated per cell at each stage.

5) The Stopping Criterion: The stopping criterion is implemented by recording the cost function's value at the end of each stage of the annealing process. The stopping criterion is satisfied when the cost function's value has not changed for 3 consecutive stages.

III. Standard Cell Placement Optimization Program

A. Introduction

TimberWolf is applicable to standard cell placement problems of the complexity shown in Fig. 1. TimberWolf optimizes the placement of standard cells into row and/or column blocks. Furthermore, the various blocks may have differing heights. The program also optimizes the placement of pads or buffer circuitry, as well as macro blocks. The macro blocks may be positioned anywhere on the chip. The estimation of the wire length for a single net is determined by computing the half-perimeter of the bounding box of the net. The bounding box is defined by the smallest rectangle which encloses all of the pins comprising the net. For the case of a two-pin net, this is the Manhattan distance. Because exact pin locations are used in the wire length calculations, TimberWolf considers all possible orientations for a cell, pad, or macro block. A group of pins which are internally connected within a cell must be given to TimberWolf as a single pin with a location which is the average of the locations of its constituent pins.

The program employs the *exchange class mechanism* for blocks as well as cells, pads and macros. If two blocks have

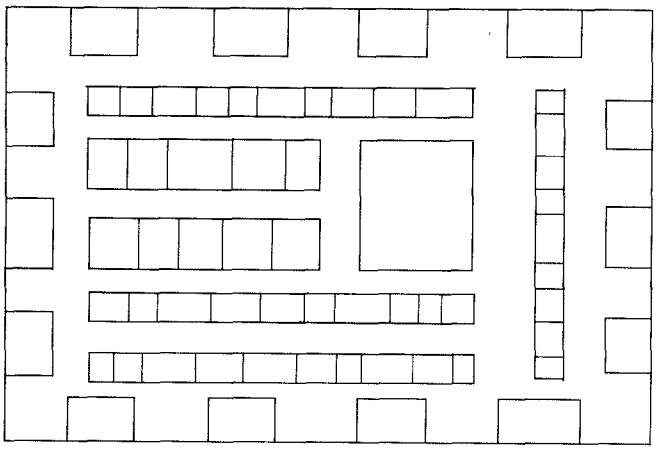


Fig. 1. Example of a general standard cell layout to which TimberWolf is applicable.

the same exchange class, then cells from these blocks are interchangeable. Blocks with differing exchange classes may not have their cells interchanged. Differing exchange classes for blocks are usually employed when blocks have different heights. Furthermore, two cells or two pads may be interchanged only if they belong to the same exchange class.

An additional feature is the net-weighting capability. For any given net, it is possible to weight the *horizontal span* of the net separately from the *vertical span* of the net. The horizontal span of a net is defined as the span of the smallest rectangle which encloses all of the pins comprising the net (bounding box) in the x direction of the x-y coordinate system. Similarly, the vertical span of a net is defined as the span of the bounding box in the y direction of the x-y coordinate system. For critical nets, it is usual to increase both the horizontal weight and the vertical weight, hence ensuring that these nets are kept as short as possible. For double-metal circuits, it is often the case that there are many uncommitted route throughs present in each cell. Consequently, vertical net spans are in some sense *cheaper* than horizontal net spans (which require the allocation of horizontal channel tracks and their associated area). In this case, the best results have been obtained when the vertical weights for the nets are made smaller in comparison to the horizontal weights.

B. Algorithm Details

The cost function for the simulated annealing algorithm consists of two independent portions. The first portion is the total estimated wire length. The second portion is the penalty function which consists of a total sum of overlap penalties. This penalty function was incorporated because of the usual difference in width of the standard cells. Often two cells are selected for interchange which differ in width. Therefore, an exchange of location of these two cells often results in some overlap with one or more of the other cells. Furthermore, the program often selects a single cell for a displacement to a new location. Once again, some overlap may result. The exchange of cells or the displacement of a single cell may also result in a portion of a cell dangling off the end of a row or column block. This is treated as a case of overlap with an *imaginary cell* being located at the ends of each column and row block. This feature increases the number of states in the state space S. Experimental investigation has shown that this results in better placements.

When two standard cells overlap, a penalty is assessed which is proportional to the square of the quantity of the amount of linear overlap plus an offset parameter. The offset parameter is chosen to ensure that when the parameter T approaches zero, then the total amount of overlap approaches zero. A larger value of the offset parameter generally results in more uniform block lengths at the expense of increased total wire length. On the other hand, a smaller value generally results in the smaller values of total wire length with less uniformity of block lengths. Experimentally it has been observed that setting the offset value to 3 yields the best overall results.

The overlap penalty function has an additional term which controls block lengths. The sum of the lengths of the cells in a particular block is compared to the actual block length. A penalty is assessed which is equal to the absolute value of the difference times a parameter value. As an example, consider the movement of a cell from a block whose total cell length is greater than the actual length of the block to another block whose total cell length is less than its actual length. The penalty term is reduced in this case. On the other hand, moving a cell from a block whose total cell length is less than its actual length to a block whose total cell length is greater than its actual length increases the penalty term. It has been experimentally observed that a parameter value of 5 results in very uniform block lengths with no compromise in the final total wire length.

The alternative to the aforementioned overlap concept is of course to not allow overlaps. For example, when inserting a cell into a row block, if insufficient space is available then the cells to the right are all shifted farther to the right as necessary. This has the obvious disadvantage of destroying the relationships between the shifted cells and the cells on the neighboring rows. The overlap concept was employed so as to not disturb the placement of the remaining cells when performing an interchange of cells or a displacement of a single cell.

The selection of new states is based on the following considerations: 1) A random number between one and the total number of cells, pads and macro blocks is generated. The cells are numbered from one to the total number of cells, and the pads and macro blocks are numbered starting from the number of cells plus one. If the random number is less than or equal to the number of cells, then a cell is selected. Otherwise, a pad or macro block is selected. 2) A second random number is selected between 1 and the total number of cells, pads, and macro blocks. 3) If the two numbers selected both represent cells, then the pair of cells are interchanged to generate a new state. 4) Similarly, if two pads or two macro blocks were selected, then an interchange constitutes the new state. 5) If the two numbers selected do not represent the same unit (that is, cell,

pad, or macro block) then the first unit selected governs the generation of a new state. If this first unit was a pad or macro block, then an orientation change of the respective unit is attempted. If the first unit was a cell, then this cell is displaced to a new location. If this new state is rejected, then the next state generated is an orientation change for the cell.

The ratio of single cell displacements to cell interchanges has a pronounced effect on the quality of the final placement. Experimental investigation has revealed that a ratio of about 5 to 1 yields the best results. Hence, if the first unit selected was a cell, the generation of the second random number is weighted to produce the desired ratio. This is implemented by generating a random number between one and the number of cells multiplied by 5.

The displacement of a cell to a new location is controlled by a *range limiter*, which limits the range of the displacement of a cell. For example, in the latter stages of the algorithm when the value of T approaches zero, the displacement of a cell has very little chance of being accepted unless the displacement is very local. By limiting the range of the cell displacements in the latter stages of the algorithm, the cells undergo many small displacements while gradually eliminating overlaps and reducing wire length.

The implementation of the range limiter is as follows. A rectangular *window* is centered at the center of the cell to be displaced and this window has a particular horizontal span and a particular vertical span. At the beginning of the algorithm, when T is at its maximum value, the horizontal span of the window is equal to twice the horizontal span of the chip and similarly the vertical span of the window is equal to twice the vertical span of the chip. The horizontal and vertical window spans are proportional to the logarithm of the value of T. Hence, when the value of T is reduced, the size of the window is correspondingly reduced. When a cell is to be displaced, a randomly-selected location within the window is chosen as the new location for the cell. That is, a block (row or column) is randomly selected which intersects the window and then a random position is selected within that block and within the window.

Pairwise interchanges of cells are also controlled by the range limiter. An interchange of two cells is attempted only if the window can be positioned such that it contains the centers of both cells.

As T is reduced, eventually the size of the range-limiter window has been reduced such that inter-block cell displacements or interchanges are no longer attempted. At this point, all residual cell overlaps are removed and the blocks are compacted. The generation of new states then takes on a different form as follows: 1) A standard cell is randomly selected and its left and right neighbors (if any) for the case of a row block or its bottom and top neighbors (if any) in the case of a column block are noted. 2) An interchange of the randomly selected cell is performed with either its left (bottom) neighbor and/or its right (top) neighbor for row (column) blocks. For example, in the case of a cell belonging to a row block, if the cell has both left and right neighbors, then one of the neighbors is randomly selected and an interchange of the cell with the selected neighbor is attempted. If the interchange is not accepted, then an interchange is attempted with the neighbor not previously selected. If the cell has only one neighbor, then only that interchange is attempted. 3) An orientation change of the selected cell is attempted if permitted by the user.

The user may also request that TimberWolf is to insert *route-through cells* as necessary if the standard cell circuit contains only row blocks. A route-through cell has two internally connected pins, one on the top and one on the bottom. If a portion of a net must connect two cells which are not on the same row and are not on neighboring rows, then this net must be routed through the rows between those containing the cells. A route-through cell must be inserted to accomplish this for the case of two levels of interconnect. Once the size of the range-limiter window has been reduced such that inter-block cell displacements or interchanges are no longer attempted, TimberWolf will then insert route-through cells as necessary. The route-through cells participate in the generation of new states as described above. That is, they are positioned in their respective rows such that the total wire length objective is minimized.

For standard cell circuits comprised solely of row blocks of cells and pads around the periphery of the blocks, the user may request that TimberWolf is to configure the rows in the most advantegeous manner. The user inputs the number of rows desired and the estimated row separation. For example, in anticipation of the fact that most of the route-through cells are concentrated toward the center-most rows, TimberWolf will restrict the total cell length allowable in these rows. The user supplies an *indent factor* which is the ratio of the total cell length allowed in the center-most row divided by the total cell length allowed in the outermost row. The total cell length allowed in the other rows increases linearly from the center row toward each of the top and bottom rows. TimberWolf also queries the user for the expected number of route-through cells. This can either be a guess or the user may try a short TimberWolf run (that is, with relatively few new states generated at each T) and note the number required. TimberWolf uses this information to increase the actual row lengths. Note that when the final placement is determined by TimberWolf and the route-through cells have been added, the final row lengths will tend to be close to the actual row lengths given to TimberWolf. Having the actual row lengths greater than the total allowable cell length for each row increases the cardinality of the state space of the problem and has been shown to yield the best results.

Of major concern to all implementations of the simulated annealing algorithm is CPU time. The TimberWolf standard cell program was designed to reduce computation time while sacrificing storage. One of the features of the program is that computation time per iteration is constant (that is, it is invariant with the number of cells). The iteration time is defined to be the time required to generate

a new configuration, evaluate the new value of the cost function, and then decide to accept or reject the new configuration. Two key features make this possible. 1) The cells in a block are hashed into bins that partition the block's coordinate system. Hence overlap calculations require a constant amount of time. 2) The possible orientations for a cell, including the pin locations for each orientation, are computed at the outset and are stored. Thus to change a cell orientation, only a pointer change is required rather than recomputing the cell boundaries and pin locations.

Additional reductions in CPU time were achieved by employing a table look-up technique for the computation of the exponential function [6]. This technique requires only 3 table look-ups and 2 floating multiplies to achieve excellent accuracy (it has been observed that the least significant decimal digit is at most plus or minus one in comparison to the exact value of the exponential function). This technique reduced the time per call to the exponential function from 107 to 44 μs on a VAX-780 system and from 75 to 2.5 μs on an IBM-3081/UTS system. Because on the order of several hundred million calls to the exponential function are made for a large standard cell problem, substantial CPU-time reductions were achieved.

Many current standard cell optimization programs attempt to first perform an inter-row optimization and then an intra-row optimization. That is, each cell is first assigned to a row and then in a second step, the cells are placed within their respective row. Note that the method employed by TimberWolf simultaneously considers both optimizations and hence better results should be obtained.

C. Results

The program was interfaced to the CIPAR standard cell placement package developed by American Microsystems, Inc. For the larger circuits tested (800 to 2700 cells), TimberWolf achieved total estimated wire length reductions ranging from 45 to 66 percent in comparison with CIPAR. Furthermore, final chip area reductions ranged from 15 to 57 percent. For a circuit of 1000 cells, TimberWolf reduced the final chip area by 31 percent in comparison to CIPAR and by 21 percent over another commercially available standard cell placement and routing package in a benchmark performed at AMI.

For the largest circuit tested (2700 cells), 75 million iterations were performed. The computation time was 300 μs per iteration (IBM 3081 running UTS), implying nearly 6.5 h of CPU time. TimberWolf runs 12.2 times faster on the IBM/UTS system in comparison to the VAX-780/VMS and VAX-780/UNIX systems.

The memory requirement is linearly related to the number of cells. For the 2700-cell circuit, the memory requirement was 4 Mbytes (32-bit integers are used). The results are summarized in Table I.

The layout of CktA1 using the TimberWolf placement was also compared to the manual layout of the same circuit. A team of designers from AMI worked approximately 4 months on the layout after which time the effort was abandoned for two reasons. First, the projected manual layout was 10-percent larger than the layout produced by CIPAR with TimberWolf, and second, the tape-out deadline had been reached. Manual layouts of circuits CktF and CktG were not attempted by AMI because of the rapid turnaround required by their customer.

TABLE I
TIMBERWOLF STANDARD CELL PLACEMENT OPTIMIZATION PROGRAM

Circuit	# Cells	Total Wire Length Reduction	Final Chip Area Reduction	CPU Time in Hours VAX 780
CktF	2700	66%	57%	84
CktG	1500	**	40%	36
CktA1	1500	45%	30%	20
CktA2	1500	37%	25%	10
CktB	1000	57%	31%	8
CktC	200	41%	15%*	2
CktD	100	37%	15%*	0.5

* pad-limited ** not recorded

CktF and CktG were double-metal circuits. Consequently there were many uncommitted route throughs present in each cell. By weighting the vertical net spans approximately one half as much as the horizontal net spans, almost 20-percent additional area reductions were achieved over equal-weighting results.

The CktC and CktD circuits could not have their areas reduced more than 15 percent due to pad limitation. There were two versions of the CktA circuit. The second version had very many of its cells specified to occur in fixed sequences. Hence the number of states in the state space S is significantly reduced. It has been experimentally observed that the wiring area reduction achieved by TimberWolf is less if the cardinality of the state space is reduced.

The effect of the TimberWolf placement optimization can be further demonstrated by the number of route-through cells which were required. For the CktD circuit, the number of route-through cells was reduced from 50 to 14. Furthermore, the number of route throughs was reduced from 51 to zero for the CktC circuit. For the CktB circuit, more than 1000 route-through cells were eliminated. All of the approximately 300 route-through cells were eliminated for the 1500-cell CktA circuit.

The TimberWolf standard cell program was also interfaced to the Zymos placement and routing package (ZYPAR). For a 1000-cell circuit, TimberWolf reduced the total estimated wire length by 44 percent in comparison to ZYPAR. The chip area reduction was limited to 8 percent as a result of using the TimberWolf placement. The smaller-than-expected area reduction was a result of the ZYPAR post-placement row-compaction routine which greatly altered the TimberWolf placement. Modification of the compaction algorithm is under way and much greater area reductions are expected as a result of using TimberWolf.

An interface to TimberWolf was also developed by Intel Corp. Two 1000-cell circuits were used for comparison to their standard cell placement and routing package. The first circuit was manually placed while the second circuit was placed automatically. The result of the TimberWolf placement was a 10-percent final chip area reduction for the first (manually-placed) circuit with a 30-percent reduction in the number of route throughs required. The TimberWolf placement resulted in a 25-percent final chip area reduction for the second circuit.

Furthermore, Hughes Aircraft Company developed an interface to TimberWolf. A 1000-cell circuit was chosen for comparison with their manual placement methodology. The result of the TimberWolf placement was a 6-percent area reduction and a 26-percent reduction in the number of route-through cells that were required for the 1000-cell circuit.

IV. Standard Cell Global Router Program

A. Introduction

The layout of a standard cell circuit often consists of rows of cells bordered by pads and/or buffer circuitry. In order to minimize the need for route-through cells (which increase the area of a circuit), the cells are typically designed with electrically equivalent (internally connected) pins on both the top and bottom side. Thus a net from above can be connected to the top pin while the same net from below can be connected to the bottom pin. The internally connected pins are referred to as a *pin cluster*. A portion of a net which must connect two pin clusters is referred to as a *net segment*.

It often arises that a pin cluster from one cell must be connected to a pin cluster from another cell on the same row. If each such cluster has a top pin and a bottom pin, then this net segment is defined as being *switchable*. A decision must be made as to whether to route the switchable net segment in the channel above or below the row. The TimberWolf global router assigns switchable net segments to channels based on the minimization of the *total channel density*. The total channel density is defined to be the sum of the channel densities for all of the channels.

The TimberWolf global router is applicable to standard cell circuits consisting of rows of cells bordered by pads and/or buffer circuitry. The global router assumes that all necessary route-through cells have been inserted into the proper rows. The global router routes all nets and considers all pins except those nets and pins which route power and ground. It is often the case (as with CIPAR) that separate routines are used to route power and ground. The global router takes into consideration pins on the outer pads or buffer cells.

Some standard cell place and route systems (for example, CIPAR) do not employ a global router. Instead, only a channel router is used and it routes as many connections as possible for each channel. Thus the order in which the channels are routed can have a substantial effect on the total number of wiring tracks required (and thus the area of the circuit). In contrast, after using the TimberWolf global router, specific pins have been identified for interconnection. Thus the number of wiring tracks required is independent of the order in which the channels are routed.

B. Global Router Algorithm

The TimberWolf global router performs the optimization in two stages. The first stage examines each net separately. Two basic steps are applied to each net. 1) The first step identifies which pairs of pin clusters are to be connected based on the minimization of the Manhattan interconnection distance. This results in the identification of the net segments. 2) The second step considers each net segment and selects a pin from each cluster such that the Manhattan length of the segment is minimized. Two pairs of pins are selected for each switchable net segment.

The second stage results in the assignment of a channel for each switchable net segment. The two stages are detailed below.

1) First Stage of the Global Router Algorithm: The first stage consists of applying the two steps detailed below to each net separately.

Step 1

For a given net, the pin clusters that need to be connected are determined. A graph is formed in which the clusters are represented by the nodes and connections between the nodes (the formation of potential net segments) are represented by edges. An edge connects two nodes if a net segment could possibly connect the two clusters. For example, two clusters can be connected only if one of the following two conditions is true. 1) They lie on the same row, with no intervening cluster occupying the same row. This is the case of a potential switchable net segment. The net segment is switchable if each cluster has a pin on the top and on the bottom of the row. That is, the net segment could be routed either in the channel above the row or in the channel below the row. 2) They lie on neighboring rows. Furthermore, there cannot be another cluster lying between the two clusters which occupies either of the rows occupied by the two clusters.

The result of conditions 1) and 2) above is that the maximum degree of a node is 4. Further, this maximum degree is achieved when a given cluster is to be connected to two clusters in the row above (one to the left and one to the right) and to two clusters in the row below (also one to the left and one to the right).

The minimum spanning tree is generated for the graph via Kruskal's algorithm [7]. This portion of the algorithm effectively generates a Steiner tree [8] for the interconnection of the clusters. When the minimum spanning tree has been generated, pairs of pin clusters have been identified which are to be connected by a net segment.

Step 2

In this step, each edge of the minimum spanning tree is examined, and one pin from each cluster is selected to form

the actual net segment. In the case of an edge connecting two clusters on the same row, it is determined if this is a switchable net segment. If the segment is switchable, then two pairs of pins are selected. One pair is for the segment routed in the channel above the row and another pair is for the segment routed in the channel below the row.

Pin selection proceeds as follows. 1) For the case of two clusters on neighboring rows, the bottom pin of the top cluster and the top pin of the bottom cluster are selected based on the minimization of the Manhattan distance between the two points. 2) For the case of two clusters on the same row: a) If the edge is determined to be switchable, the top pin from each cluster is selected based on the minimization of the distance between the two points. Also, the bottom pin from each cluster is similarly selected. b) If the edge is not switchable, either the pair of top pins (if the segment must be routed in the channel above the row) or the pair of bottom pins (if the segment must be routed in the channel below the row) are selected. The pin selection is again based on the minimization of the segment length.

2) Second Stage of the Global Router Algorithm: This step employs a simulated annealing algorithm. The net segments (for all of the nets) with their respective pins are supplied as input. One half of the minimum contact-to-contact spacing is added to each end of the horizontal span of each segment. For each switchable segment, an arbitrary initial selection (of above or below the row) is made. Each channel is examined sequentially to determine its density. The densities of the channels are summed, and this sum is the initial value of the cost function. A new state of the configuration is generated by the random selection of a switchable segment and then routing it on the opposite side of the row from its current position. As a result of the new state, the cost function either increases by 1, decreases by 1, or remains the same. That is, the total channel density changes by at most 1.

The case of no change in the cost is treated further. This is the case in which the net segment switch has no effect on the total channel density. A second *cost function* is introduced in this case. This cost function is a measure of the congestion in a channel between the two points defining the span of a net segment. The cost function is evaluated by taking the difference between the overall channel density and the density between the two points defining the span. The cost function is first evaluated for the span of the net segment in the original channel. Next, the cost function is evaluated for the net segment span in the new channel. The difference in cost (Δc) is determined by subtracting the second cost function value from the first. A negative value of Δc indicates that switching the net segment to the new channel places the segment in a channel of less congestion.

C. Results

The global router was also interfaced to the CIPAR placement and routing package developed by AMI. The global router reduced the number of wiring tracks used by the CIPAR router by 10 to 20 percent. Because routing typically occupies one half of the chip area, this translated to an overall area savings of 6 to 8 percent.

TABLE II
TIMBERWOLF STANDARD CELL OPTIMIZATION PROGRAMS

Circuit	# Cells	Global Router Area Reduction	Final Chip Area Reduction	CPU Time VAX 780 in Hours
CktF	2700	8%	62%	1
CktG	1500	8%	45%	0.5
CktA	1500	6.1%	34%	0.5
CktB	1000	6%	35%	0.3

For the largest circuit (2700-cell CktF), the global router reduced the area by an additional 8 percent. A total area savings of 62 percent was achieved for CktF when both TimberWolf placement optimization and the global router were applied. The results are summarized in Table II, showing the additional area reductions due to use of the global router and also the overall area reductions as a result of using both TimberWolf placement and global routing.

Simulation results for CktG revealed that all interconnections had capacitance values below the specifications, and hence that the circuit should operate properly at the specific clock rate. Simulation results for CktF were not available at this time.

Fig. 2 depicts the layout of a 1500 cell circuit which was produced by CIPAR. The layout as a result of using TimberWolf for placement and global routing is shown in Fig. 3. Note that the TimberWolf layout was pad limited and hence the area reduction achieved was limited to 11 percent. However, the core size (the area inside the pad ring) was reduced by 22 percent in area.

V. MACRO/CUSTOM PLACEMENT OPTIMIZATION PROGRAM

A. Introduction

This program optimizes the placement of macro cells and custom cells, as well as pads. The term *macro cell* will be used to refer to a cell contained in a cell library. That is, the dimensions of the cell are known, as are the pin locations. The term *custom cell* will be used to refer to a block of circuitry known only to occupy an estimated area and to possess a list of pins.

The program places circuits comprised solely of macro cells as well as circuits comprised entirely of custom cells. Furthermore, the program will place circuits consisting of a combination of macro and custom cells. The macro cells and custom cells may be of any rectilinear shape.

TimberWolf allows the specification of lower and upper bounds for the aspect ratio of a custom cell. If a range of aspect ratios is given for a custom cell, TimberWolf will try to select the shape of the cell which minimizes chip area.

Wire length calculations are based on the exact pin locations. Thus all possible orientations are considered for each cell.

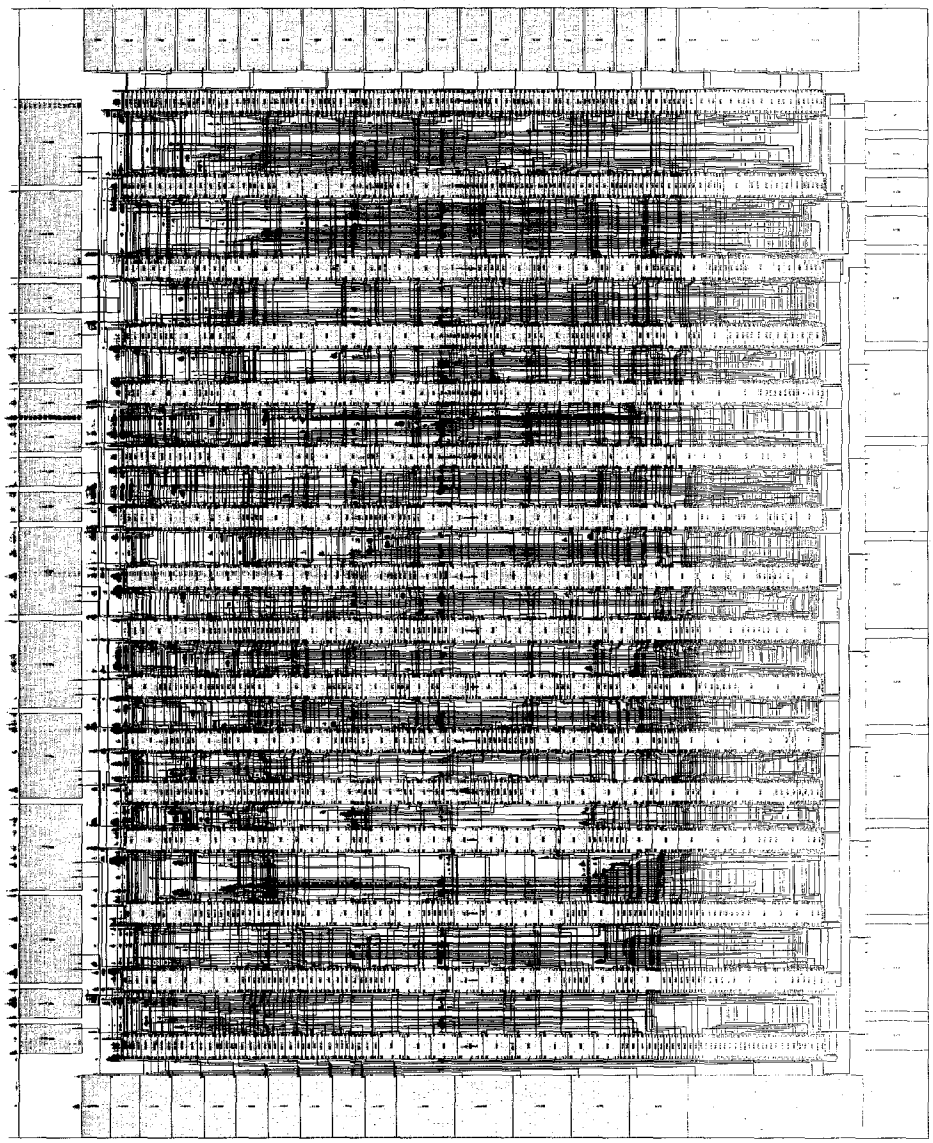


Fig. 2. CIPAR layout of 1500-cell circuit without TimberWolf.

Another feature of TimberWolf is the multiple region capability. This feature incorporates either a division of the chip into regions or the placement of multiple chips simultaneously. Interchanges of cells from different regions are permitted only if the regions belong to the same exchange class. The exchange class mechanism is extended to individual cells as well.

Pins are specified in several possible ways. 1) A pin may be given a particular fixed location. 2) A pin may be assigned to a particular side or sides of the cell. 3) A group of pins may be assigned to a particular side or sides of a cell. 4) A group of pins may be assigned to a particular sequence as well as a particular side or sides.

B. Macro/Custom Cell Placement Algorithm

For macro and custom cells, there are often pins on all of the sides of the cells. Consequently, wiring space must be allocated around each cell. If insufficient space is allocated during TimberWolf placement, the global and detailed routers will have to (perhaps substantially) alter the placement. The strategy employed by TimberWolf to ensure routability with a minimum amount of placement alteration during routing consists of the following: TimberWolf (by default) computes the expected wiring area required along each side of each cell based on the number of pins on that side. Appropriate borders are then appended around the enclosed area of the cell. This prevents cells from abutting in the final placement and hence allows approximately sufficient wiring space around each cell. Furthermore, TimberWolf allows the user to override the default border values.

The number of possible locations at which an *uncommitted* pin could be placed on a custom cell can often number into the thousands. Execution time considerations (as in the standard cell program) require that the pin locations be stored for each orientation of the cell. Clearly the amount of storage required can become excessively large. This potential problem is averted by defining a

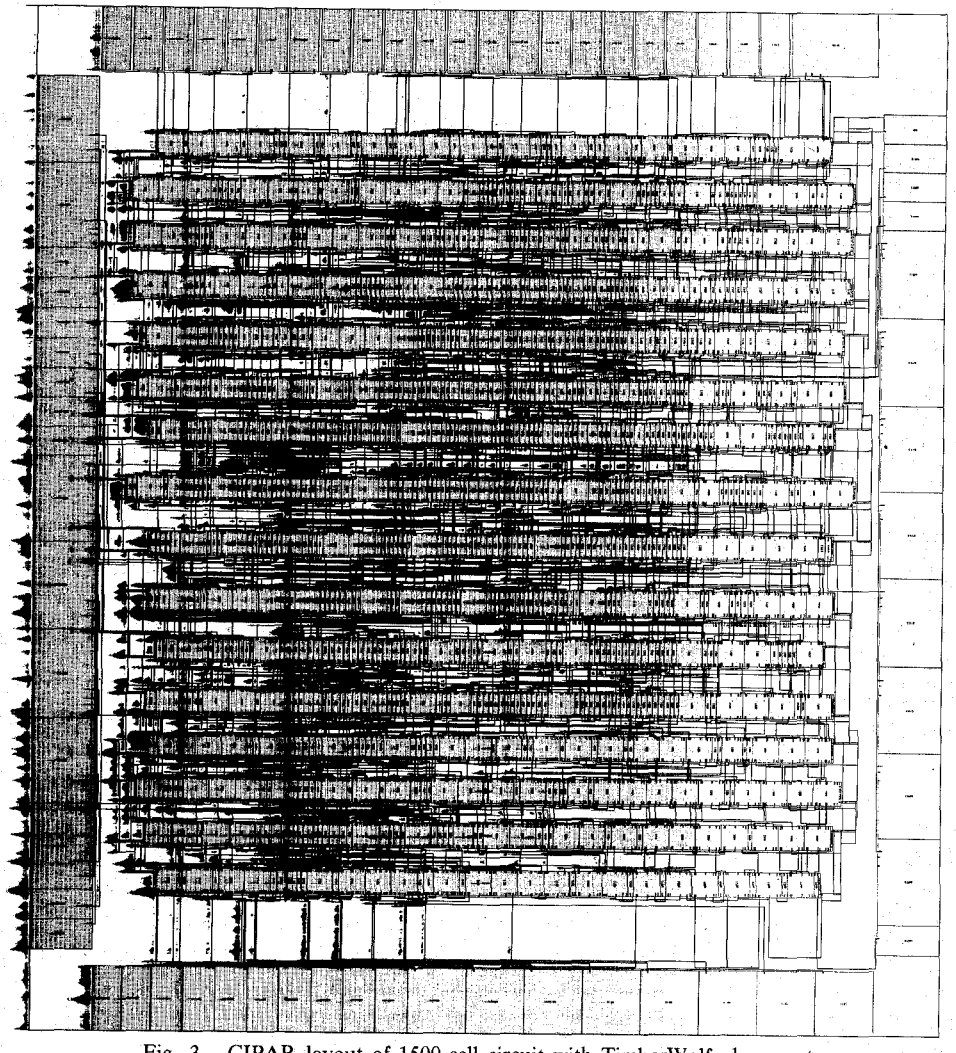


Fig. 3. CIPAR layout of 1500-cell circuit with TimberWolf placement and global routing.

specified number of pin *sites* approximately evenly spaced along the periphery of a cell. Furthermore, each site is assigned a *capacity*. The capacity is a function of the number of pin locations encompassed by the site. During the annealing stages, pins are assigned to sites. Upon completion of the annealing algorithm, the pins for a given site are assigned to locations within the scope of the site based on the minimization of wire length. For accuracy considerations, the number of pin sites that are declared for a given placement problem is usually limited only by memory capacity.

The location of the pins on a macro cell are taken exactly. That is, their location is not approximated by the pin-site mechanism. The same is true for fixed-location pins on custom cells (if any are so specified). The capacity for a site in the vicinity of a fixed-location pin is correspondingly reduced.

The cost function consists of two independent parts. The first part is the total estimated wire length which is based on the sum over all nets of the half-perimeter of a net's bounding box. The second is the penalty function. The penalty function consists of two parts. 1) The first part is the sum of the overlap penalties for the cells. This penalty function was incorporated because of the usual difference in the size and shape of the cells. Often two cells are selected for interchange which differ in size and/or shape. Therefore, an exchange of location of these two cells often results in some overlap with one or more of the cells. Furthermore, the program often selects a single cell for a displacement to a new location or an aspect ratio change (in the case of custom cells). Once again, some overlap may result. The penalty assessed for an overlap of two cells is equal to the square of the quantity of the area of overlap (including cell borders) plus an offset value. The offset parameter is selected to ensure that as the parameter T approaches zero, then the total overlap approaches zero. 2) The second part is the sum of the penalties assessed for the contents of a pin site exceeding its capacity. When a pin is displaced from an original site to a new site, the contents of the old site is reduced by 1 and the contents of the new site is increased by 1. The penalty assessed for a site is a product of the square of the amount by which the contents exceed the capacity, times a factor inversely related to the capacity of the site. This factor reflects the fact that exceeding the capacity by a given amount is a more serious violation for the sites with smaller capacities.

New states can be generated in several possible ways. 1) A pair of cells (either could be a macro cell or a custom cell) are selected for interchange. 2) A single cell is selected for a displacement to a new location. 3) A single cell is selected for an orientation change. 4) A custom cell is selected for an aspect ratio change. 5) An uncommitted pin (or sequence of pins) is assigned to a new site (or sites).

The ratio of single cell displacements to cell interchanges has a significant effect on the quality of the final placement. Initial experimental investigation has revealed that the best results are obtained when the ratio is about 10 to 1.

The strategy for generating new states is based on the following: 1) A random number between one and the number of cells is generated. The cells are numbered sequentially from one. 2) A second random number is generated between 1 and the number of cells times 10. 3) If the two numbers both represent cells, then the pair of cells are interchanged to generate a new state. 4) If only the first number represents a cell, then the new state is generated by the displacement of the cell to a randomly selected location. If this new state was rejected, the next state generated is an orientation change for the cell. Similarly, if this new state was rejected and if the cell is a custom cell, then the next state is an aspect ratio change. Finally, if this new state was rejected, then a new state is generated by the selection of an uncommitted pin or group of uncommitted pins for transfer to a new pin site or sites.

As in the case of standard cell placement, the displacement of a cell to a new location is controlled by a *range limiter*, which limits the range of the displacement of a cell. For example, in the latter stages of the algorithm when the value of T approaches zero, the displacement of a cell has very little chance of being accepted unless the displacement is very local. By limiting the range of the cell displacements in the latter stages of the algorithm, the cells undergo many small displacements while gradually eliminating overlaps and reducing wire length.

The implementation of the range limiter is as follows. A rectangular *window* is centered at the center of the cell to be displaced and this window has a particular horizontal span and a particular vertical span. At the beginning of the algorithm, when T is at its maximum value, the horizontal span of the window is equal to twice the horizontal span of the chip and similarly the vertical span of the window is equal to twice the vertical span of the chip. The horizontal and vertical window spans are proportional to the logarithm of the value of T. Hence, when the value of T is reduced, the size of the window is correspondingly reduced. When a cell is to be displaced, a randomly selected location within the window is chosen as the new location for the cell. That is, a region is randomly selected which intersects the window and then a random position is selected within that region and within the window.

Pairwise interchanges of cells are also controlled by the range limiter. An interchange of two cells is attempted only if the window can be positioned such that it contains the centers of both cells.

This program is also applicable to printed circuit board placement problems. The circuits to be placed are handled in the same manner as macro cells, that is, cells with fixed geometry and fixed pin locations. In printed circuit board layouts, total wire length and maximum wire length minimization are important objectives per se in addition to their correlation with ease of routing. In fact, signal crosstalk due to long wires may cause signal degradation and limit the speed of operation much more severely than in integrated circuits.

C. Results

The TimberWolf macro/custom cell placement optimization program is currently being interfaced to CIPAR for testing purposes. In addition, testing is in progress on some macro cell circuits designed at UC Berkeley.

This program was applied to a Honeywell Information Systems Italy printed circuit board problem in which the circuits had variable size. The processor board required the placement of 613 circuits, each of which had from 2 to 64 pins. The circuits had to be placed on a 14.4×16 in printed circuit board. The processor board had 900 nets, 4000 pins, and contained 3 microprocessors.

TimberWolf used 18 h of CPU time on a VAX-780/UNIX system to place the circuits. The placement obtained was routed by the HONDA (Honeywell Design Automation) printed circuit router and 96-percent routing completion was achieved.

For comparison, the manual placement of the same printed circuit board was considered. The total estimated wire length of the TimberWolf placement was 21-percent less than the manual placement. HONDA was also run on the manual placement resulting in 99-percent routing completion. The TimberWolf placement resulted in a 10-percent reduction in actual total wire length. Furthermore, the manual placement required approximately 4 months of effort on the part of the design team.

These results are preliminary since a few constraints deriving from the automatic insertion of components on the printed circuit board were neglected by TimberWolf. Furthermore, the HONDA router is specifically tuned to a particular layout style, and hence is not fully compatible with an automatic layout program such as TimberWolf. Minor modifications to the router should produce improvements in the final results.

VI. Gate-Array Placement Optimization Program

A. Introduction

This section describes the generalized gate-array placement program. Each fundamental unit in a gate array will be referred to as a *cell*. Hence, a 50 by 50 gate array is said to have 2500 cells. Some gate array designs allow additional flexibility and hence greater gate utilization by creating functionally independent units within a cell. For

example, Tektronix gate arrays widely utilize functional units which are half-cell sized. TimberWolf allows the functional units to be half-cell sized or quarter-cell sized. The term *module* will refer to a fundamental unit specified in the net list. A module may be the size of: 1) a full cell, 2) a half cell, or 3) a quarter cell. Additionally, macro modules may be specified. A macro module consists of a prewired, arbitrarily shaped collection of cells.

TimberWolf has other features which provide additional flexibility. For example, a module (or macro module) may be designated as unmoveable (that is, preplaced) or as belonging to an exchange class of modules. The modules in such a class may only be interchanged among themselves. This feature is often desirable when a group of modules on the edge of the gate array are to be considered as primary terminals. Often the exact location of a given primary terminal is not important, only that it lie on a given edge.

It is often the case that gate arrays have wider channels in the center of the array. This is in anticipation of the greatest wiring congestion occurring in this region. Because prewired macro modules usually have a fixed cell-to-cell spacing, certain macros may not be placed in the center region (or the outer regions). TimberWolf allows the designation of cell locations as either suitable or unsuitable for a particular set of macro modules.

B. Gate-Array Placement Algorithm

The TimberWolf gate array placement program can be used with either of two cost functions. The first cost function is based on the computation of net crossing *histograms* for each horizontal and vertical channel of the placement region. The histograms are computed by considering the bounding box of each net and adding 1 to the histogram for each channel intersecting the bounding box. The sum of the histogram values for each horizontal and vertical channel is equivalent to summing the half perimeters of the bounding boxes of each net. Further, a net-crossing *threshold* value is assigned to each channel. If the number of nets crossing a channel exceeds the specified threshold value, a penalty is assessed proportional to the square of the number of net crossings exceeding the threshold. The threshold mechanism has the effect of evening out the wiring congestion during the earlier stages of the annealing. This has shown to result in a lower value of the total wire length. A partitioning effect may be produced by setting the threshold of a particular channel to zero or a negative value. In this case, nets crossing this channel will be severely penalized. The formulation of the cost function in terms of net-crossing histograms and threshold values was first introduced by Kirkpatrick, Gelatt, and Vecchi [1].

A second cost function for this program examines the local routing congestion more closely. For this cost function, each *channel segment* is assigned a threshold value. A channel segment is a portion of a horizontal or vertical channel with a length equal to the cell-center to cell-center spacing in that region of the array. For example, if the bounding box of a net encompasses 2 cells in the horizontal direction and 3 cells in the vertical direction, then a total of 17 segments are enclosed by the bounding box. The congestion per channel segment introduced by this net is approximated as the half perimeter of the bounding box 5 divided by the total number of segments enclosed 17. The factor of 5/17 is the estimated probability of occupancy for the given net in each of the 17 segments. The given net contributes zero to all other segments. The summation of the occupancy probabilities over all nets for a given segment is an estimate of the number of wiring tracks required. The cost function is then the sum of the expected occupancy of each segment plus a penalty assessed for each segment which has occupancy exceeding the corresponding threshold. Specifying a threshold value for each channel segment which reflects the actual fixed channel width increases the likelihood that the final placement will be routable. Furthermore, the total wire length will be minimized within the limits of these constraints.

TABLE III
TIMBERWOLF GATE ARRAY PLACEMENT PROGRAM

Circuit (# modules)	Stevens	Goto and Kuh	TimberWolf	CPU Time in Mins.
151	2181	2098	1731	15
108	untested	1242	909	10
67	700	618	580	5

C. Results

Experiments are currently being initiated on large gate array problems. To test the program and compare it with existing placement techniques, a set of standard benchmarks have been considered. These benchmarks are the ILLIAC IV computer boards reported by Stevens [9]. Note that the printed circuit board problem as stated for these examples is a particular case of the general gate array placement problem described in the previous subsection.

Wire length for a net was estimated by computing one half of the perimeter of the net's bounding box. The figure of merit is the sum of the estimated wire lengths for each net.

Three of the ILLIAC IV computer boards were tested. 1) The largest example required the placement of 151 modules on an 11×15 board. TimberWolf reduced the total wire length by 21 percent over Stevens' result and by 17 percent over the result published by Goto and Kuh [10]. 2) The second example required the placement of 108 modules on an 8×15 board. TimberWolf reduced the total wire length by 27 percent over the result published by Goto and Kuh. 3) The third example required the placement of 67 modules on a 5×15 board. TimberWolf reduced the total wire length by 17 percent over Stevens' result and 6 percent over the result published by Goto and Kuh.

The value of α remained at a constant value of 0.90 for each of the examples. The results are summarized in Table III. CPU times are for a VAX 11/780 running UNIX.

VII. Conclusions

The TimberWolf placement and routing package has been shown to provide substantial chip area savings in comparison to existing standard cell layout programs. Substantial wire length reductions were also achieved for the gate array placement program for some benchmark examples. The TimberWolf macro/custom program is applicable to placement problems as complex as a multichip design employing a combination of macro cells and custom cells. The macro/custom program was applied to an industrial circuit board problem and improved the manual placement by 10 percent in terms of total (exact) wire length.

The TimberWolf placement and routing package is written in the C programming language. The package currently runs under both the VAX/UNIX and VAX/VMS operating systems as well as the IBM/UTS system. The package is easily convertible to other systems supporting the C language.

Acknowledgment

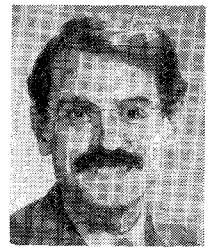
The authors would like to thank American Microsystems, Inc. for allowing the interface of TimberWolf to the CIPAR system and for providing the test circuits for the standard cell package. The authors would also like to thank Honeywell Information Systems Italy for providing the test case for the printed circuit board package. Special thanks are also extended to Intel Corp. for providing computer time on an IBM 3081 for TimberWolf testing.

The support of J. Tobias and B. Kirk of AMI, S. Nachtsheim of Intel, and D. Cesa Bianchi, L. Fezzi and M. Vinsani of HISI is gratefully acknowledged. Further, the authors deeply appreciate the efforts of T. Young of Zymos and C. P. Hsu of Hughes Aircraft Co. in developing TimberWolf interfaces and for providing standard cell test circuits.

The authors wish to thank F. Romeo and K. Keller for stimulating discussions. C. Sechen wishes to thank P. Moore, T. Quarles, R. Spickelmier, and M. Hofmann for their significant contributions to his knowledge of the C programming language and the UNIX operating system.

References

[1] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by simulated annealing," IBM Computer Science/Engineering Technology Watson Res. Center, Yorktown Heights, NY, Tech. Rep., 1982.
[2] M. Vecchi and S. Kirkpatrick, Global Wiring by Simulated Annealing, *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 215–222, 1983.
[3] F. Romeo and A. Sangiovanni-Vincentelli, Probabilistic Hill Climbing Algorithms: Properties and Applications, ERL, College of Engineering, Univ. California, Berkeley, CA, Mar. 1984.
[4] D. Geman and S. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," unpublished.
[5] M. Lundy and A. Mees, "Convergence of the annealing algorithm," unpublished.
[6] J. Deutsch, private communication, UC-Berkeley, 1984.
[7] J. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Amer. Math Soc.*, vol. 7, no. 1 pp. 48–50, 1956.
[8] M. Hanan, "Net wiring for large scale integrated circuits," IBM Res. Rep. RC-1375, IBM, Feb. 1965.
[9] J. Stevens, "Fast heuristic techniques for placing and wiring printed circuit boards," Ph.D dissertation, Univ. of Illinois, Urbana, IL, 1972.
[10] S. Goto and E. Kuh, "An approach to the two-dimensional placement problem in circuit layout," *IEEE Trans. Circuits Syst.*, vol. 25, p. 208,. Apr. 1978.

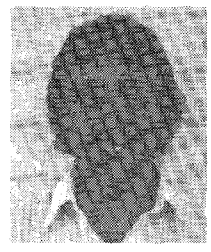


Alberto Sangiovanni-Vincentelli (M'74–SM'81–F'83) received the Dr. Eng. degree (summa cum Laude) from the Politecnico di Milano, Italy, in 1971.

From 1971 to 1977, he was with the Istituto di Elettrotecnica ed Elettronica, Politecnico di Milano, Italy. In 1976 he joined the Department of Electrical Engineering and Computer Sciences of the University of California at Berkeley, where he is presently Professor and Vice Chairman. He is a Consultant in the area of computer-aided design to several industries including IBM, AT&T Bell Laboratories, Harris Semiconductors, and General Electric. His research interests are in various aspects of computer-aided design of integrated circuits, with particular emphasis on VLSI simulation and optimization. He was Associate Editor of the IEEE Transactions on Circuits and Systems, Associate Editor of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. He is an Associate Editor of the IEEE Design and Test Magazine, a member of the Large-Scale Systems Committee of the IEEE Circuits and Systems Society and of the Computer-Aided Network Design (CANDE) Committee. He was the Guest Editor of a Special Issue of the IEEE Transactions on Circuits and Systems on CAD for VLSI. He has been elected Executive Vice-President of the IEEE Circuits and Systems Society in 1983 in charge of Membership Development. He is also Division I Representative in the IEEE Membership Development Committee.

In 1981 he received the Distinguished Teaching Award of the University of California. At the 1982 IEEE-ACM Design Automation Conference he was given a Best Paper and a Best Presentation Award. In 1983 he received the Guillemin–Cauer Award for the best paper published in the Transactions on CAS and CAD in 1981–1982. At the 1983 Design Automation Conference he received a Best Paper Award.

Dr. Sangiovanni-Vincentelli is member of ACM and Eta Kappa Nu.



Carl Sechen received the B.E.E. degree (with highest honors) from the University of Minnesota in 1975. He received the M.S. degree from M.I.T. in 1977. In 1977 and 1978 he was with Honeywell Systems and Research Center. Since 1979 he has been a graduate student at the University of California at Berkeley where he is completing the requirements for the Ph.D. degree.

His research interests are in many aspects of computer-aided design of VLSI circuits, with emphasis on automated layout. He is working at several companies including Intel, American Microsystems, Inc., Hughes Aircraft Co., Harris Semiconductors, and Zymos Corp.

Mr. Sechen is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.