Chapter 4

Mixed Macro/Row-Based Cell Placement

4.1 Introduction

Two prevalent cell-based IC layout design styles are standard cell and macro cell. The standard cell design style uses fixed height preconfigured cells arranged in rows. The routing regions for standard cells occurs in the rectangular channel regions between the rows. In contrast, the macro cell design style arranges more complex rectilinearly shaped cells in a two dimensional plane. Macro cell routing is confined to horizontal and vertical rectilinear regions between the cells. In both methodologies, the core region (the area containing the standard or macro cells) is surrounded by pad cells which make the connection to the outside world. Standard cells are best suited for random logic, such as control logic, and macro cells can be customized to design regular arrays efficiently. The standard cell method has the advantage that a library can be built and shared among many designs increasing the productivity of the IC designer and shortening the time-to-market. However, the transistor density (number of transistors per unit area) of standard cell designs is substantially below the densities achieved in manually designed custom macros. With the recent introduction of module (cell) generators, the design time bottleneck of large regular arrays such as RAM, ROM, and PLAs has been removed. Automatic custom macro cell generation makes it possible to enjoy the advantages of both methodologies - high silicon efficiency in arrays and the design time reduction of standard cells. Figure 4.1 shows an example of a possible mixed style design.

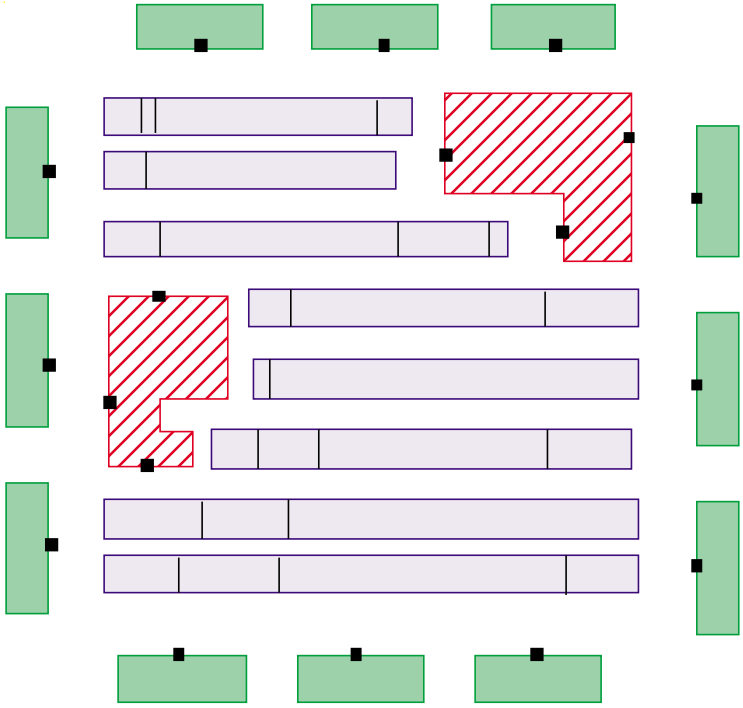

Figure 4.1 An example of a mixed macro / standard cell design.

4.2 Previous Work

Placement and routing methods must adapt to the new constraints imposed by this emerging style. Very little prior work has been reported for the mixed macro/standard cell layout problem. Reingold and Supowit proposed a constructive method which handled macro and standard cells [180]. The original TimberWolf placement and routing package could handle a very limited number of manually placed macro cells [198]. The LTX2 system places the standard cells automatically, but the user must manually preplace the macros [31]. In the VITAL system, macro and standard cells are clustered into groups called domains [174]. The domains are placed relative to one another in a binary tree or slicing structure based on connectivity and area. A simulated annealing algorithm is used to determine only the placement of the standard cells within the domains. The VITAL approach has several problems: Since standard cells cannot be swapped between domains, the simu-

lated annealing algorithm's search space is limited by the quality of the constructive slicing structure placement. The constructive slicing structure placement does not allow rectilinearly shaped macros. In addition, it does not estimate the routing area needed for wiring between domains resulting in the need for substantial placement modification during the detailed routing step. Furthermore, this method does not permit the addition of timing constraints. The Macro Block Placement (MBP) program extended the VITAL approach to allow swapping between domains [229]. However, this method cannot accommodate rectilinear macros and is not timing driven.

To the best of our knowledge, there have been no prior methods which are able to automatically place and route rectilinearly shaped macro cells in the presence of standard cells and meet timing constraints.

In this paper, we present a new system for placement and routing of the mixed macro/standard cell style. The system consists of a sequence of programs that perform placement and global and detail routing of mixed cell designs. In addition, timing driven placement is used.

4.3 TimberWolf Mixed Macro / Standard Cell Flow Algorithm

The flowchart of Figure 4.2 corresponds to the TimberWolf mixed macro/standard cell placement and routing flow. The first program object is *edit_files*, a shell script which allows the user to edit the input parameter files upon entering the system. The user can, of course, change the shell script to call a translator, or alternatively change the flow file to directly call the translator as a program object.

4.3.1 Error Checking: Syntax

The first program necessary for correct operation of the TimberWolf system is the *syntax* program. Its function is to check for errors in the user's input. Implemented using a

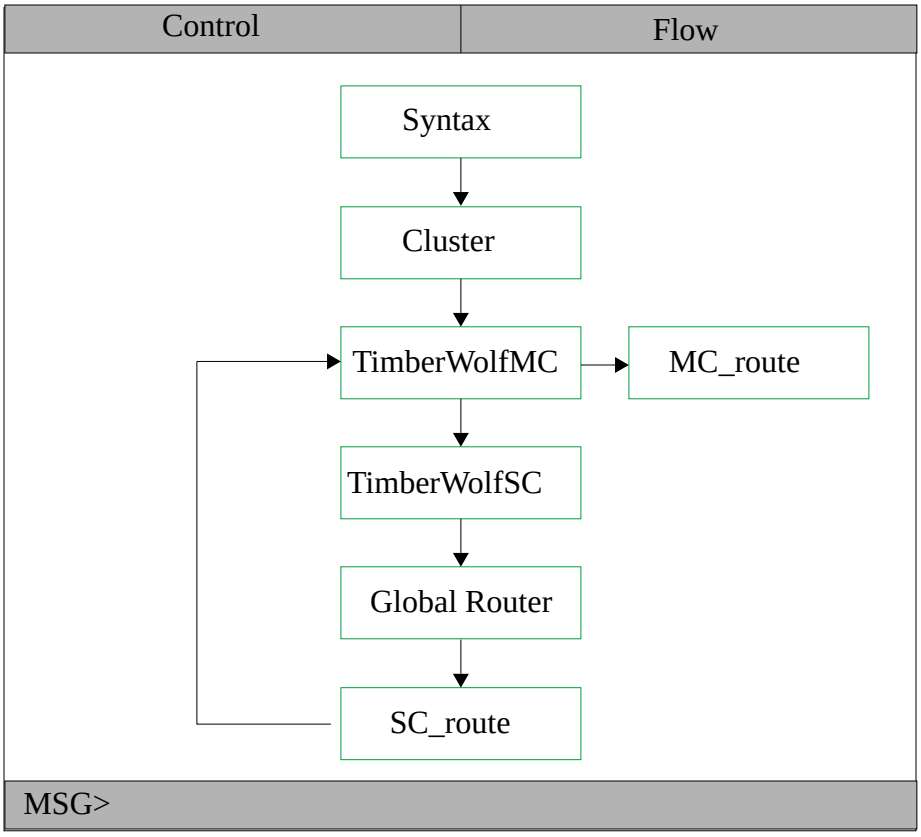

Figure 4.2 Flow chart of TimberWolf mixed macro/standard cell placement and routing algorithm.

yacc/lex parser, this program also compiles vital statistics on the design such as number of standard cells, number of macro cells, etc. to make it easier for downstream programs to allocate resources.

4.3.2 Standard Cell/Macro Cell Clustering

The next step in the flow is the *Cluster* program which groups the standard cells in the netlist into a set of standard cell clusters for floorplanning. These clusters have variable aspect ratios, and in addition, the signal pin locations are not fixed but may be adjusted in order to minimize the wirelength.

By default, all standard cells are grouped into a single standard cell core region. The standard cell core region is specified as multiple versions as shown in Figure 4.3. Here the

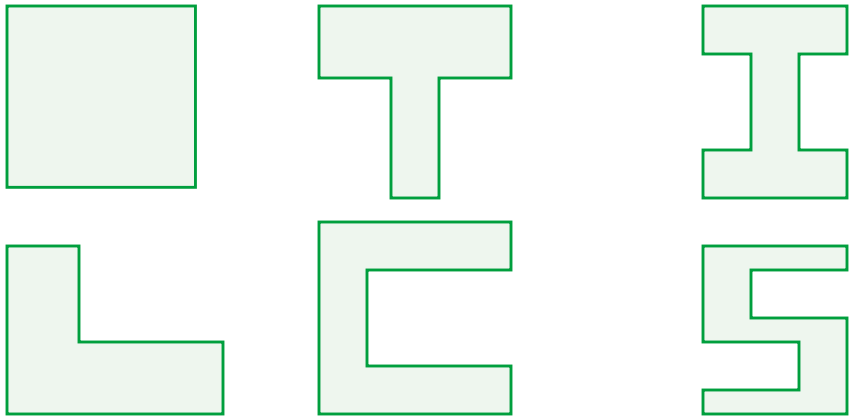

Figure 4.3 Multiple versions of the standard cell core. User may specify additional versions.

multiple version capability of the floorplanning tool is exploited. The floorplanning tool will pick the version of the standard cell core which minimizes area and timing delay.

If desired, the standard cell core may be partitioned into multiple core regions using mincut bipartitioning. The *Mincut* program features a new objective function which better enables the program to find natural partitions of the standard cells [197]. The *Mincut* program partitions the standard cells into standard cell clusters which are approximately equal to the average macro cell area. This insures that both macro cells and standard cell clusters have equal importance in the determination of the core region topology. Empirically, we found that mincut partitioning reduces the quality of the final solution.

In all cases, the area of a standard cell cluster *s* is given by:

$$A_s = (r+1)\sum_{i=1}^{n_s} A_c(i) \tag{4.1}$$

where A_s is the area of the standard cell cluster *s*, *r* is the ratio of the row separation distance to the row height, n_s is the number of standard cells in cluster *s*, and $A_c(i)$ is the

area of standard cell *i*. The estimated row separation factor *r* accounts for the standard cell routing area needed between the rows. This parameter is user supplied.

After partitioning, the *Cluster* program outputs a netlist for the floorplanner containing the macros and the standard cell clusters. An example of a mixed design after partitioning is shown in Figure 4.4. The light shaded cells are standard cell clusters. The macro cells (dark shaded cells) are arbitrarily placed at the origin, and the pads (hatched regions) are placed outside the estimated core region.

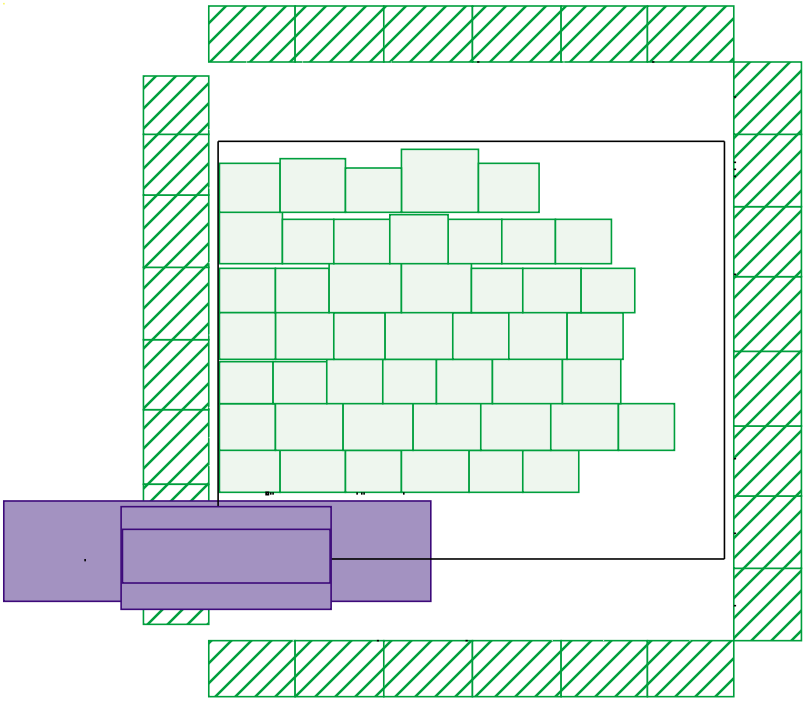

Figure 4.4 Result of a *Mincut* partition for a mixed macro/standard cell example.

4.3.3 Floorplanning: TimberWolfMC

TimberWolfMC is a timing driven floorplanner which handles cells of any rectilinear shape. The cells may have fixed geometries such as pin locations (hard macros), or the cells may have an estimated area with a specified aspect ratio range and pins that need to be placed (soft macros). In addition, the cells may have several possible versions. *Timber-WolfMC* is to select the one which is most suitable. Cells may also be grouped hierarchi-

cally. Cells and/or cell groups may be restricted to subregions within the core region. Through the use of the X11R3-R5 graphics interface, the user may interrupt the automatic execution at any time to add region restrictions or to place macros at specific locations. *TimberWolfMC* also features a sophisticated I/O placement algorithm which places the pad cells so as to minimize wirelength subject to side, spacing, and grouping constraints.

Both *TimberWolfMC* and the standard cell placement program *TimberWolfSC* support timing driven placement. The timing driven algorithm is presented in Chapter 6.

TimberWolfMC uses an adaptive dynamic interconnect-area estimator in order to allocate the necessary interconnect space around each cell. Failure to account for the correct amount of routing area during floorplanning results in placements that require significant modification during detailed routing. In this algorithm, we are using the floorplanner to determine the relative placement of the macro cells to the standard cell clusters. If large amounts of placement modifications are required during detailed routing, this relative placement will be destroyed. *TimberWolfMC* updates its estimate of the core region periodically during the course of an execution of the simulated annealing algorithm. The core area estimate in this case is the sum of the macro cell area, the estimated macro cell routing area, and the area of the standard cell clusters. Standard cell clusters already have routing area added to them as noted previously.

At the completion of the simulated annealing run, the floorplanner has determined the placement of the macro cells. The result of floorplanning is shown in Figure 4.5.

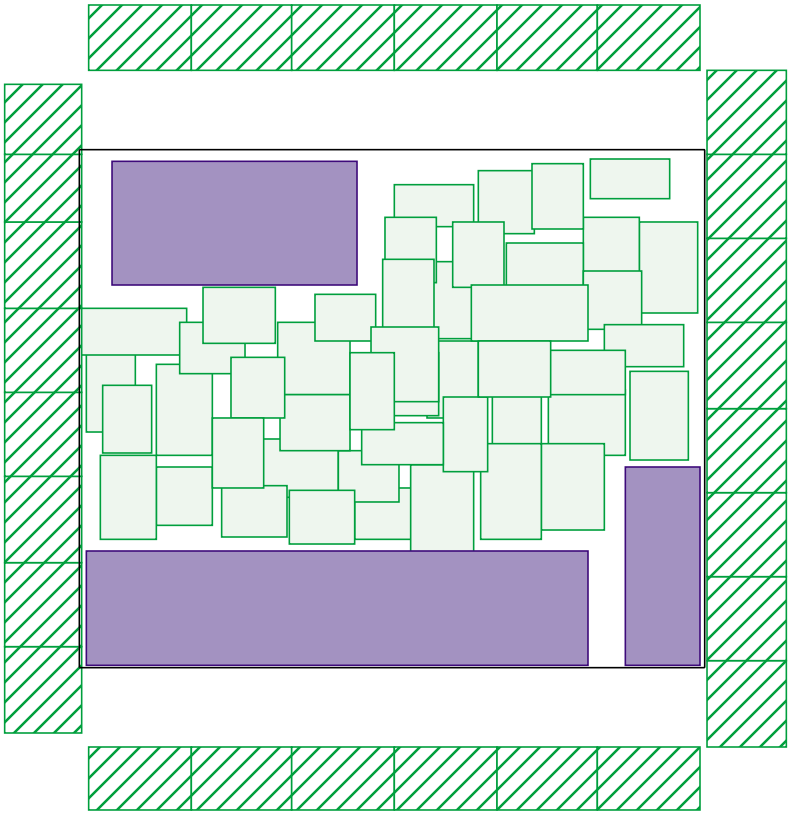

Figure 4.5 Result of floorplanning.

4.3.4 Standard Cell Row Generation: Genrows

We need to determine a configuration of standard cell rows which will satisfy the following constraints: the standard cell rows must span the core but must not overlap the macro cells or the macro cell routing area, the total row length should be equal to the sum of the standard cell widths, and the standard cell rows should be placed at the specified row separation.

The standard cell row topology generation is performed by the program *genrows* which is spawned as a child process of the *TimberWolfMC* floorplanner. The floorplanner outputs the size of the core region deemed necessary and the vertex lists of all the macro

cell blocks. The standard cell clusters, having served their purpose, are then thrown away. The algorithm is as follows: First the entire core region is broken into tiles. The tiles are constructed from the spaces unoccupied by the macros and the macro cell routing. Vertically adjacent tiles are merged if their horizontal extents match. Figure 4.6 shows an example of the evolution of the development of the tiles. Note the original tiles 4 and 5, and also 6 and 7 are merged.

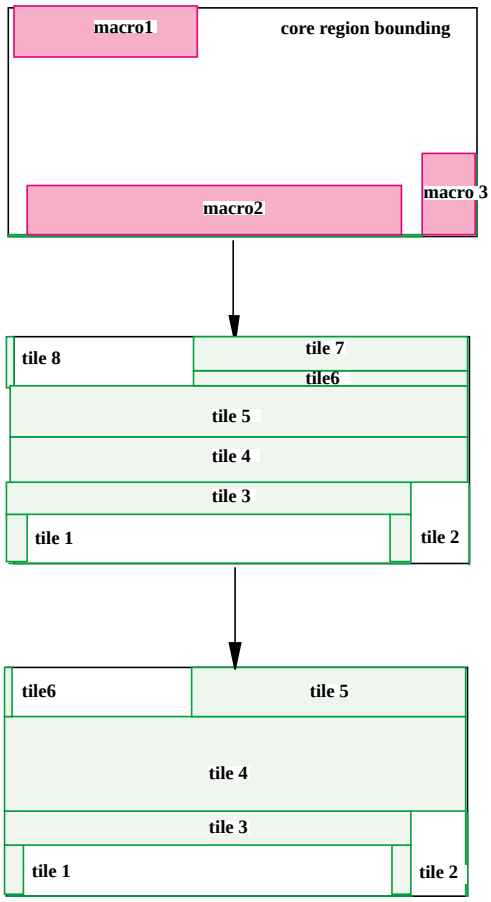

Figure 4.6 Tile construction and merging.

Next, the placement of the standard cell rows is determined. Rows are first proposed for each tile. Additional rows across the top of the core are added if necessary. All of the

rows are extended slightly to the left and right to make the total row length exactly equal to the total cell width. The final result of *genrows* is shown in Figure 4.7.

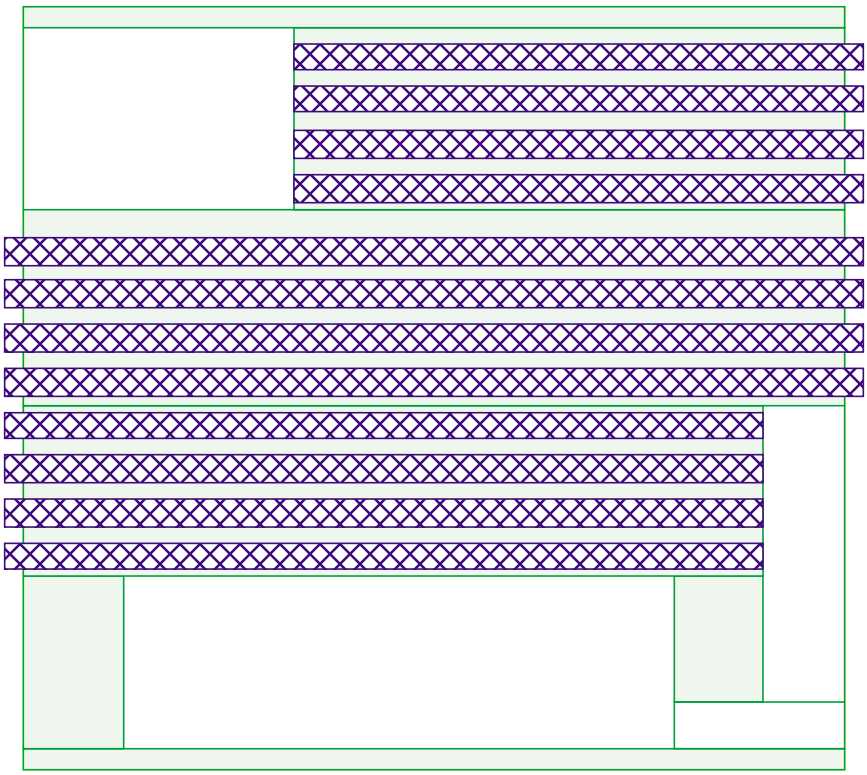

Figure 4.7 The final standard cell row topology.

4.3.5 Standard Cell Placement: TimberWolfSC

After the configuration of the standard cell rows, the placement of the standard cells occurs with the invocation of the *TimberWolfSC* program [199][201]]. *TimberWolfSC* uses simulated annealing to place the standard cells by minimizing the total interconnect length subject to timing constraints. Now that *TimberWolfMC* has determined the placement of the macro cells relative to the standard cell clusters, we can observe the positions of the macro cell pins as fixed. The task remains to place the standard cells minimizing the total interconnect length.

4.3.6 Row - Based Global Routing

After placement of the standard cells, global routing is performed. In this step, each net segment (pin-to-pin interconnection) is assigned to a particular channel on the chip subject to interconnect length and congestion constraints. The global routing algorithm is presented in Chapter 7.

4.3.7 Standard Cell Detailed Routing: Sc_route

After the global routing of the standard cell core, the detailed routing of the standard cell channels is performed under the direction of the *sc_route* program. This program make a function call to *TimberWolfDR*, our derivative of the *Mighty* maze router [207] for each channel within the core region. All of the channels between the standard cell rows are routed by *sc_route*. The channels above the top row and below the bottom row will be routed during the inter-macro routing step.

The *sc_route* program achieves design rule correct routing without the need for compaction if the design rules are simple (i.e. if the vertical track pitch equals the horizontal track pitch and together, they equal the via pitch). A data transformation feature was added to *TimberWolfDR* to make the input specification easier; *TimberWolfDR* translates to and from the actual chip coordinates and the integer grids (comprising the original Mighty input format) of the internal maze data structures. The user only needs to specify the vertical and horizontal track pitch and the pin coordinates. If a pin's coordinates do not fall exactly on a multiple of the grid, *TimberWolfDR* rounds the coordinates to the closest internal grid based on the specified horizontal and vertical track pitch. Maze routing proceeds from the internal grid. A post-processing step in *TimberWolfDR* connects the internal grid locations to the user-specified locations. *TimberWolfDR* will inform the user if two pins on the same layer map to the same internal grid implying a design rule violation exists between the pins.

Routing of the standard cell core proceeds by routing each interior channel from the bottom as follows: *sc_route* first directs *TimberWolfDR* to give an estimate of the number of tracks necessary for a successful route of the current channel. Using this information, it outputs the channel and the pin locations to accommodate this space. If TimberWolfDR is successful, we continue on to the next channel; otherwise, we increase the channel separation by one track and try again. We continue increasing the separation until we successfully route the channel. From experience, we have found that *TimberWolfDR* routes most channels in density, and almost all channels in density plus one tracks.

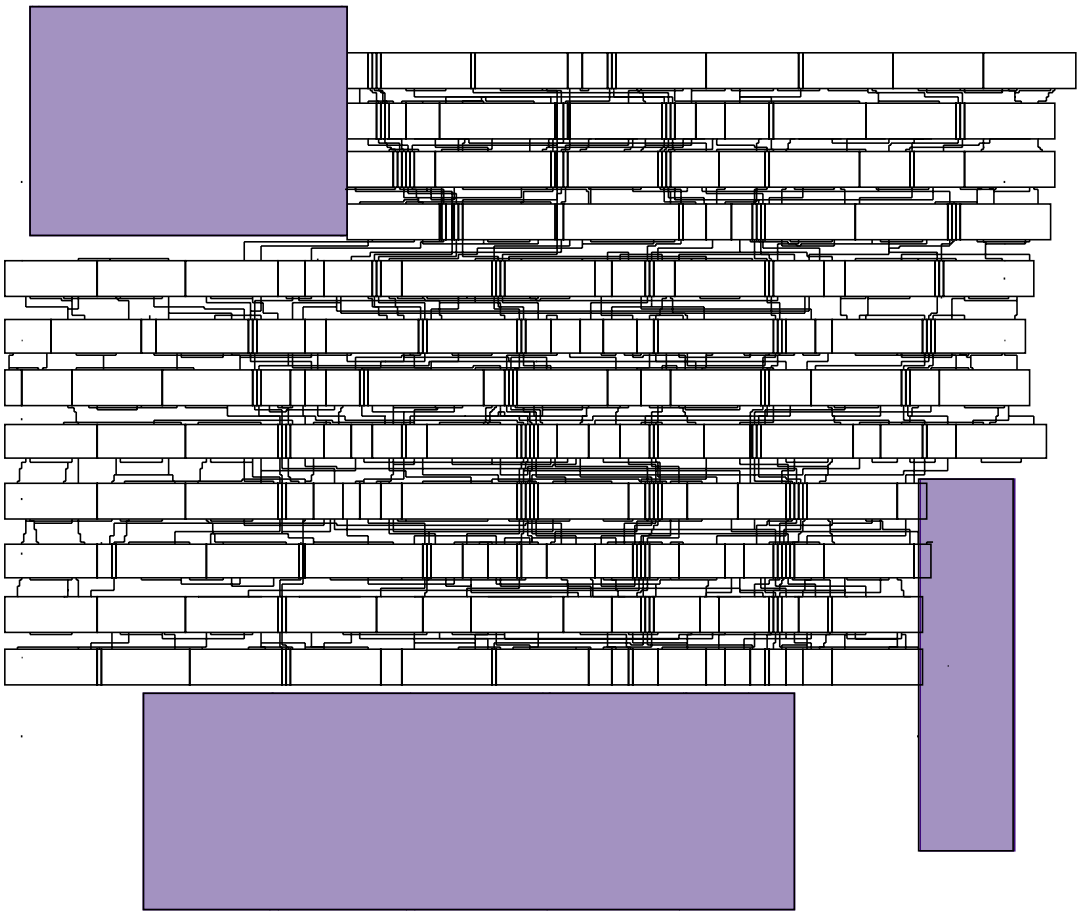

Figure 4.8 The core region after standard cell routing.

4.3.8 Macro Cell Placement Refinement: TimberWolfMC

After the execution of *sc_route*, we return control to *TimberWolfMC* since we now need to complete the interconnections between the pads, macros, and the routed standard cell core. We also need to remove any possible overlaps between the macros and standard cell core and eliminate any unneeded empty space on the chip. Here we use *TimberWolfMC's* placement refinement phase to space the pads and macros treating the standard cell core as simply another rectilinear macro cell. Hence, the problem has now been reduced to a macro-only place and route problem.

Using the placement refinement algorithm presented in Chapter 3, we obtain the result shown in Figure 4.9.

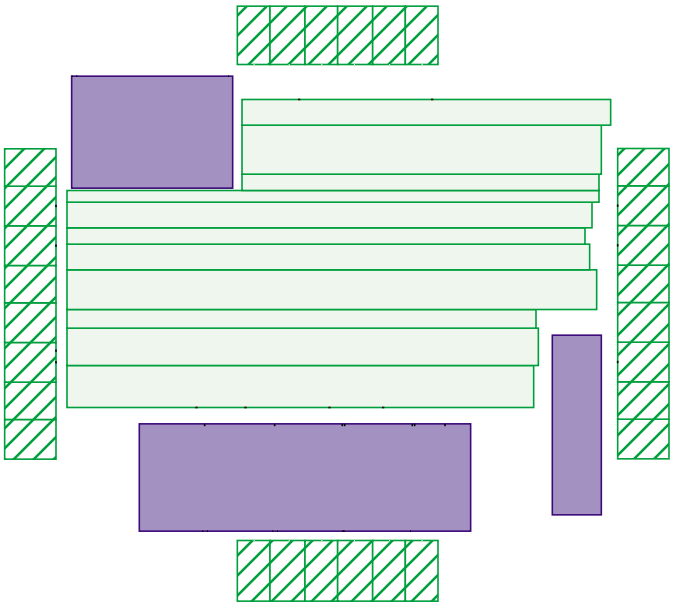

Figure 4.9 The example circuit after placement refinement.

4.3.9 Macro Cell Detailed Routing: Mc_route

Next, we call *mc_route* to direct *TimberWolfDR*, our derivative of the *Mighty* maze router [207]. As presented in Chapter 3, *TimberWolfDR* has been modified to allow routing to cells within a switchbox by using *macro objects* to model the rectilinear cells. This

eliminates the channel definition problem. The routing segments are coerced into the routing regions determined by the global router through the use of pseudopins. These pseudopins are ordered at the end of the regions using a modification of Groeneveld's algorithm [78]. In addition, the memory requirements are reduced by one order of magnitude using local grid lines as shown in Table 4.1.

Approach	No. of *x* grids	No. of *y* grids	Percentage
local grid	328	81	12.7
full	531	393	100.0

Table 4.1 The results of using a local grid for the example circuit.

4.3.10 Results

We have executed TimberWolf on a recent industrial circuit from Texas Instruments, Incorporated. Figure 4.10 shows the results of the mixed macro/standard cell placement. Table 4.2 displays the statistics for this circuit.

Number of standard cells	180
Number of macro cells	3
Number of nets	281
Wire length (after detailed routing)	97102 μm^2
Area	248 $\mu m \times 3144$ μm

Table 4.2 Statistics for the Texas Instruments circuit.

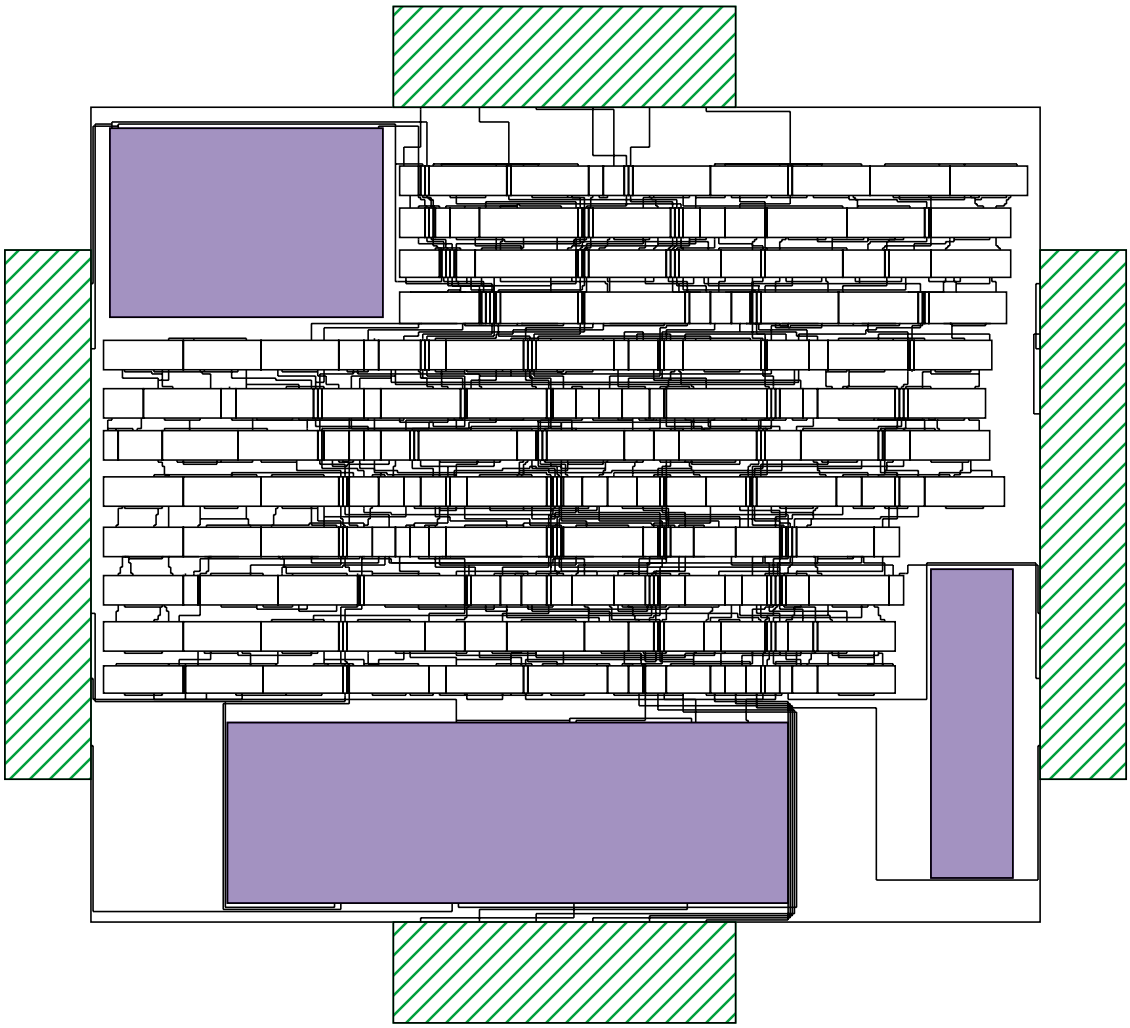

Figure 4.10 The final routing for the Texas Instruments circuit.

4.4 Conclusions

A new fully automatic placement and routing system has been developed for mixed macro/standard cell designs. The system uses a simple yet effective X11R3-5 graphics interface to allow user interaction if desired. At both the standard cell and macro cell level, placement is driven by timing constraints. Based on the results of floorplanning, a new robust method for generating the standard cell row topology has been developed. The floorplanner handles cells of any rectilinear shape and accurately estimates the area neces-

sary for routing. In addition, a new detailed routing method at the macro cell level has been presented. Detailed routing is performed all at once relieving the need for channel definition and eliminating the channel-routing order problem. A local grid approach is utilized to decrease the memory requirements of the detailed router for large mazes. Finally, pseudopins are introduced to take advantage of the global routing optimization. We have obtained results for industrial circuits.