Floorplanning with Pin Assignment*

Massoud Pedram Malgorzata Marek-Sadowska Ernest S. Kuh

Electronics Research Laboratory
University of California, Berkeley, CA 94720, USA

Abstract

We present a hierarchical technique for floorplanning and pin assignment of the general cell layouts. Given a set of cells with their shape lists, a layout aspect ratio, relative positions of the external I/O pads and upper bound delay constraints for a set of critical nets, we determine shapes and positions of the cells, locations of the floating pins on cells and a global routing solution such that a linear combination of the layout area, the total interconnection length and constraint violations for critical nets is minimized. Floorplanning, pin assignment and global routing influence one another during the hierarchical steps of the algorithm. The pin assignment algorithm is flexible and allows various user specified constraints such as pre-specified pin locations, feedthrough pins, length-critical nets and planar net topologies. Placement, timing and floorplanning results for *Xerox* general cell benchmark are reported.

1 Introduction

The layout synthesis problem is very complex and, hence, is often divided into a sequence of simpler tasks. Initially, positions and shapes of cells are determined. This step is followed by pin assignment which obtains positions for *floating* pins on the cell boundaries and by global routing which assigns connection paths to the nets. Finally, routing area is broken into smaller regions which are sequentially processed by appropriate detailed routers. The reason for this decomposition is computational. In fact, floorplanning, pin assignment and routing steps must be combined in order to generate high quality layouts. (Performing global routing is usually sufficient since it makes accurate estimation of the layout area and the interconnection length possible.) In addition, a hierarchical approach which appropriately prunes the solution space and reduces the design objects to manageable sizes is advantageous. In this paper we present a hierarchical technique for simultaneous floorplanning, pin assignment and global routing.

Previous works on pin assignment assume that shapes and positions of cells are given as input data. These algorithms can be classified into three categories:

1 Those which assign pins on a cell by cell basis [1, 2];

2 Those which assign pins on a net by net basis [3, 4];

3 Those which sequentially process edges of a supergraph containing the global route solutions for all nets, finding a coarse pin assignment and global routing solution followed by a local pin assignment optimization for that global routing [5].

Among these approaches, only [5] correlates pin assignment with global routing. However, the quality of the global routing with coarse pin assignment depends on the ordering in which the 'non-essential' edges are eliminated and there is no way to determine a 'good' ordering for edge deletion. It is not clear to us how any of these approaches could be extended to include the floorplanning task.

The technique proposed in this paper avoids cell or net ordering problems. Floorplanning and pin assignment influence each other during the hierarchical steps of the algorithm. Floorplanning determines positions and shapes of hierarchical cells, sets the channel topology and assigns capacities to the routing regions. Pin assignment and global routing operate on the hierarchical floorplanning solution and are weaved in order to produce assignments for floating pins which minimize the layout area as well as the total interconnection length. The initial pin assignment sets the stage for the global routing step by assigning positions to the floating pins. Global routing, then, determines connection patterns and defines channel densities. This information is subsequently used to adjust the pin positions. Global spacing is also performed in order to guarantee routing success.

We formulate the problem and give an overview of our solution technique in Section 2. We discuss our hierarchical floorplanning and pin assignment techniques in detail in Sections 3 and 4. Sections 5 and 6 contain our experimental results and conclusions.

2 Overview of the Algorithm

We assume that each cell is characterized by a *shape function* which defines the cell's height as a function of its width. We further assume that area of a cell is not sensitive to the arrangement of pins on its boundary. We consider rectangular cells and measure quality of the layout by the area of the smallest rectangle which completely covers all cells and routing regions. The user may specify a set of critical nets with propagation delay constraints, impose constraints on the relative positions of the pins, or require planar topology for *Vdd* / *Gnd* nets. Some pins may be fixed while others are floating.

We generate a hierarchical representation of the circuit in the form of a multi-way *cluster tree*. Each leaf in the tree corresponds to an actual cell, and each internal node (which we will alternately refer to as a *cluster node* or a *hierarchical cell*) represents a collection of highly connected cells (or clusters of cells). After building the cluster tree, we traverse it from the bottom up and compute lower bounds on the area of each cluster node as a function of its aspect ratio (i.e, the shape function for the node). At the same time, we estimate the total length of interconnections which lie entirely within the node as a function of node area (i.e., the interconnection length function for the node). These functions are used during the top down floorplanning to guide the search for a good floorplanning solution. (See [6] for details.)

In the top down phase we start from the root of the cluster tree and floorplan the nodes in a breadth first manner. As a result of floorplanning a cluster node, we assign shapes and positions to its child nodes and update the current partial floorplan solution. Next, we enter the pin assignment and global routing phase for the node. We do an initial pin

*This project is supported in part by Semiconductor Research Corporation under grant number 89-DC-008 and National Science Foundation under grant number MIP-88-3711.

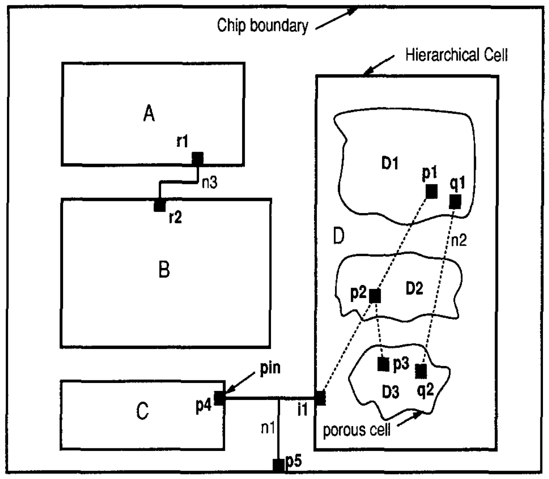

Figure 1: Partial floorplan solution prior to floorplanning D.

assignment which produces a pin assignment solution consistent with relative pin position, net topology and channel capacity constraints and minimizes the sum over all nets of the half perimeter length of the net bounding boxes. Then, we do global routing which produces shortest connection paths for all nets. After global routing, if capacity constraints for some channels are violated, we perform a final pin assignment which re-positions the floating pins to reduce congestions in the over-subscribed channels. The process of the top down traversal of the cluster tree continues until the leaf level is reached.

3 Floorplanning

We define some terminology and notations. A k-room *floorplan pattern* is a floorplan structure with exactly k rooms. An *orientation* of a pattern is a clockwise rotation of the pattern with respect to the external I/O pin locations. A *labeling* of a pattern is an assignment of nodes of the cluster tree to individual rooms. A *topological possibility* refers to a particular choice of floorplan pattern, pattern labeling and pattern orientation [7, 6].

The user has specified the chip aspect ratio, the *relative* locations of the external I/O pads and upper bound delay constraints on a few critical nets. Initially, we convert these timing constraints to upper bound constraints on the length of critical nets using a simple delay model. (See [8] for details.) Next, we floorplan the root of the cluster tree by enumerating all topological possibilities and picking the possibility that minimizes area, interconnection length and critical net-length constraint violations. In the process we assign shapes, positions and pin locations to the children of the root. Then, we floorplan these child nodes in the order of decreasing area. However, prior to floorplanning a child node, we update the global net list to include cells and connections inside the node. Updating the cell list means that we delete the node and insert its children into the list. Updating the pin lists consists of deleting pins on the node boundary and adding pins on the cells inside the node. Referring to Figure 1, D is about to be floorplanned. The current net list consists of cells A, B, C, D and nets $n_1 = (p_4, p_5, i_1)$ and $n_3 = (r_1, r_2)$. We update the global net list to include cells A, B, C, $D1$, $D2$ and $D3$ and new nets $n_1 = (p_1, p_2, p_3, p_4, p_5)$, $n_2 = (q_1, q_2)$ and $n_3 = (r_1, r_2)$. This updating is beneficial since it provides a global view of the layout plane and the connections during floorplanning and pin assignment of D.

To measure the quality of a proposed floorplan solution for a cluster node, we proceed as follows. For an enumerated topological possibility, we add areas and interconnection lengths contributed by the children of the node as given by their shape and interconnection length functions. These functions act as estimates of the expected layout cost starting from the partial layout solution where the child nodes are not yet floorplanned to the complete layout. Next, we add the area and the interconnection length which are required to combine the child nodes into the enumerated possibility.

4 Pin Assignment

We enter the pin assignment phase after a node (e.g., D) has been floorplanned. Thus, we know shapes and positions of its child nodes ($D1$, $D2$ and $D3$), shapes, positions and pin locations for other nodes (A, B and C), the estimated routing area around the cells (channel capacities) and the global net list as depicted in Figure 2. Our goal is to assign locations to the I/O pins on the child cells such that the channel capacity constraints are satisfied while the total interconnection length and the critical net length violations are minimized .

In order to avoid necessity for the sequential processing of cells or nets, we transform the pin assignment problem into a linear sum assignment problem as follows. We build a cost matrix whose rows correspond to the floating pins on the child cells and its columns correspond to the *pin slots* (feasible pin locations) on the child cells. By solving the linear assignment problem, we determine locations for the floating pins. For that assignment we perform global routing and calculate channel densities. If some channels are over-subscribed, we repeat the pin assignment procedure. The initial and final assignments differ only in the way we set-up and calculate the linear assignment cost matrix.

Routing area may be very irregular. Therefore, in order to store the routing information, we use the general approach of [9], which was also adopted in the *BEAR* layout system [10, 7]. The entire area of a layout is covered with rectangles referred to as tiles. There are two kinds of tiles: *solid* tiles which represent cells and *space* tiles which represent empty space for routing between the cells. Given a placement of rectangular shaped general cells, we define two tile planes: the horizontal tile plane where all space tiles are maximal horizontal strips and the vertical tile plane where all space tiles are maximal vertical strips. In the tile plane, each space tile has four edges: two of them are called *spans* of the tile (which are completely covered by the solid tiles); the other two form *sides* of the tile. A space tile is a *bottleneck* tile if its sides are covered by the sides of adjacent space tiles. These are the areas where wire congestion is most likely to occur. A *junction region* is the maximal empty space which is completely surrounded by the solid tiles, bottleneck tiles or the plane boundaries.

Each side of a solid cell is divided into a set of *segments*. The *bottleneck* segments are those maximal intervals of sides of cells which are fully covered by the adjacent bottlenecks. The *junction* segments are the remaining maximal segments. We show the tile planes in Figure 2. (a_2, a_3), (a_4, a_5) and (a_7, a_1) are the vertical bottleneck segments of the cell A. (a_3, a_4) and (a_5, a_6) are vertical junction segments of cell A. Similarly, (b_2, b_3), (b_4, b_5) and (b_6, b_1) are the vertical bottleneck segments of B and (b_5, b_6) is a vertical junction segment of B.

For each segment of each cell we determine the number of feasible pin slots. First, we determine the number of pin slots on the bottleneck segments. Suppose that at most t parallel wires can pass through

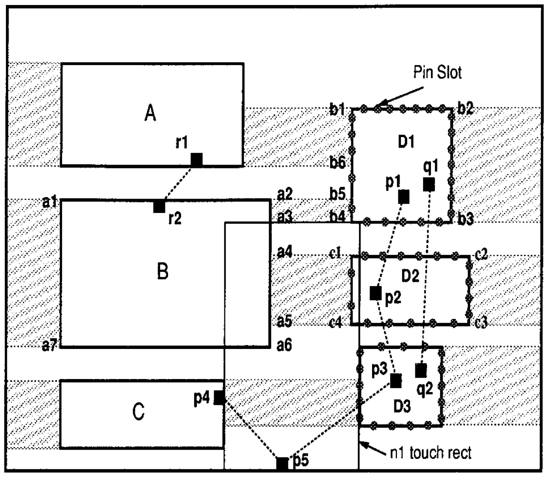

Figure 2: Partial floorplan solution prior to initial pin assignment.

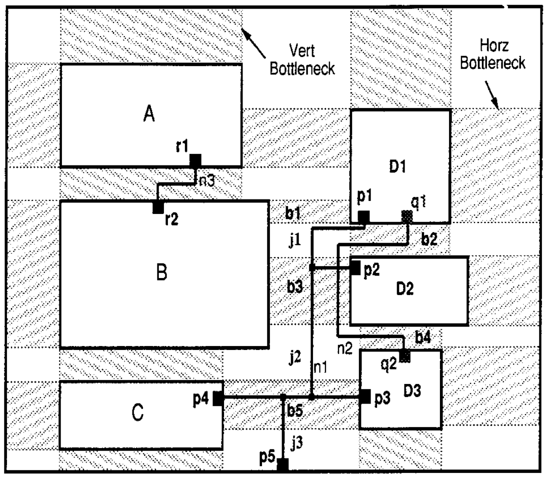

Figure 3: Partial floorplan solution prior to final pin assignment.

a bottleneck. We presume that $\alpha \times t$ pins can be placed on each segment covered by that bottleneck. $1/\alpha$ is the track utilization factor which is about 0.65 using a standard channel router. We uniformly distribute these pin slots along the bottleneck segment. (The length of a bottleneck segment and the number of pin slots in it determines the minimum pin-to-pin spacing which must be at least as large as that dictated by the design rules.) Having calculated the number of pin slots on all the bottleneck segments, we add them to get the number of pin slots on each cell. If a cell has less slots than it has floating pins, new pin slots are added on the junction segments. For each junction segment we pick the more pessimistic spacing of the adjacent bottleneck segments. For example, referring to Figure 2, spacing of pin slots in segment (b_4, b_5) is bigger than the spacing in segment (b_6, b_1), therefore, for junction segment (b_5, b_6), we pick the same spacing as that in segment (b_4, b_5). If after adding pin slots to the junction segments, there are not enough slots on some cells, we de-compact the node so that the number of feasible slots on each child cell may be increased to be equal to or bigger than the number of floating pins there.

For the initial pin assignment we call the cost matrix $[C]$ and determine its entries as follows. For each net having floating pins on the child cells, we construct a minimum rectangle which touches the child cells and the external I/O pins connected by the net. (Figure 2 shows the touch rectangle for net n_1.) All the slots that fall within this *touch rectangle* and lie on the child cells connected by the net are assigned a zero cost. Other slots have positive costs proportional to their Manhattan distances from the touch rectangle. Pin slots on the cells which are not connected by the net are assigned infinite costs. Next, we run a linear assignment algorithm [11] on the matrix $[C]$. As a result, a subset S of entries c_{kn} of matrix $[C]$ is chosen such that the following holds:

$$\forall i \;\exists j^\star : \; c^{ij^\star} \in S,$$
$$if \; i_1 \neq i_2 \; then \; j_1^\star \neq j_2^\star,$$
$$\sum_i c^{ij^\star} \; is \; minimum.$$

Since rows in the cost matrix $[C]$ correspond to floating pins and columns correspond to the pin slots, the linear assignment determines pin assignment with the minimum cost.

During the initial pin assignment we only implicitly consider the bottleneck congestions (by controlling the number of available slots per segment of each child node). However, chip area and total wire length can be accurately estimated only after the global routing. It is, therefore, necessary to combine global routing with pin assignment as is described below. After initial pin assignment, we perform global routing on the partial floorplan. The global router produces the shortest connection paths for all nets. (We modified [12] to ignore the channel capacity constraints.) This routing scheme may result in over-congested channels. In that case, we do a final pin assignment which repositions the floating pins on the child cells in order to reduce congestions in the over-subscribed channels. First, we re-calculate the number of pin slots on each segment of each child cell based on the bottleneck congestions after global routing. In particular, we decrease the number of pin slots in the over-subscribed bottlenecks (i.e., $density > capacity$) and add to this number in the under-subscribed bottlenecks (i.e., $density < capacity$). Next, we calculate the new cost matrix $[D]$. Its structure is similar to that of the matrix $[C]$, that is, $[D]$ has the same number of rows as $[C]$ but may have different number of columns.

For each net, we examine the connection tree produced by the global router. For this tree, we identify the list of junction regions that the net goes through. All the slots that fall within these junction regions and are on cells connected by the net have cost zero. All other slots on the connected cells have a cost proportional to their minimum Manhattan distances from the nearest junction region. Slots on cells which are not connected by the net have infinite cost.

To motivate the above slot cost calculation, consider Figure 3. This figure shows the partial floorplan solution after the initial pin assignment and global routing on Figure 2. The global routing for n_1 goes through junction regions j_1, j_2 and j_3 and for n_2 goes through j_1 and j_2. Without loss of generality, assume that bottleneck region b_1 is under-subscribed and b_2 is over-subscribed. The goal of the second pin assignment is to alleviate routing congestion in b_2 by moving pins out of that region. To achieve this goal, we reduce the number of pin slots in b_2 and increase the number of pin slots in b_1. Therefore, there will be more competition (among pins of competing nets n_1 and n_2) for available pin slots in b_2 and less competition for those in b_1. Since there are not enough pin slots in b_2 to accommodate all the pins, pins of some nets have to be shifted out. There are pin slots in b_1 and b_2 which have the same Manhattan distances from the junction region j_1. Therefore, we can move either p_1 or q_1 into b_1 without increasing the sum cost. No pins in b_1 has to move out since the number of slots in b_1 have been increased. Thus, the linear assignment solver will move either p_1 or q_1 out of b_2 into b_1. In general, this cost calculation procedure tends to reduce the channel congestions with a minimal increase in the total interconnection length.

It is worthwhile noting that the above procedure based on linear sum assignment does not find the optimal pin locations within each routing channel, and therefore, must be followed by a channel pin arrangement procedure as in [5]. For example, consider net n_3 in Figure 3. This net does not pass through any junction region. We want to minimize the half perimeter length of the box enclosing its pins. It is well known that this task cannot be accomplished using linear assignment. (The cost of assigning a pin to a slot is dependent on the position of the other pin.)

Our method also handles nets whose pins were preassigned at the expense of more work during the floorplanning phase and slightly more complex processing during the slot generation and positioning phase. In particular, during the floorplanning step, we must optimize orientation of the cells based on the locations of their fixed pins, and when calculating the number and the distribution of pin slots within bottleneck boundaries, we must account for the presence of the fixed pins. (We maintain a list of free boundary regions for each child node.) One may wish to have a special pin assignment for power and ground nets to satisfy planar routing topology for these nets. In one such scheme, all *Vdd* pins are placed on pin slots located on the top and left cell boundaries and all *Gnd* pins are placed on bottom and right boundaries by giving infinite cost to undesirable pin slots for each *Vdd* or *Gnd* pin. If there are some critical nets, we assign very high costs to the pin slots which are located outside the zero-cost regions for the nets, hence, ensuring minimum interconnection lengths for these nets. Of course, this may lead to increased wire length for non-critical nets and increased total wire length.

We also allow feedthrough insertion on the non-leaf nodes. After constructing the minimum touch rectangle for a given net, if the rectangle is completely 'blocked' by a child node, we insert one or two feedthrough pins on the child node. Next, we decompose the net into two spanning subtrees as follows. We find a minimum spanning tree connecting all pins of the net (which must include the feedthrough pin(s) and the edge that goes through the child node). Then, we remove the feedthrough edge and obtain two connected subtrees. The pins in each subtree define a subnet which is then passed into the pin assignment and the global routing phases.

5 Experimental Results

We have implemented our floorplanning technique in the *C* language on a *DEC3100* running Ultrix Worksystem V2.1 and have incorporated it into the *BEAR* system [10]. Due to lack of published results on floorplanning and pin assignment, we can not present comparative results for our system. However, we are able to compare layouts produced by our system with those produced by *BEAR* release 1.0 placement [7]. *BEAR* release 1.0 uses a top-down hierarchical placement approach similar to ours but does not perform floorplan sizing and pin assignment. Nor does it honor the timing constraints. We ran *BEAR* release 1.0 placement on *Xerox* general cell benchmark and recorded layout area and interconnection length after detailed routing. (Chip aspect ratio was varied from 0.5 to 3.0 and all leaf cells were assigned rigid shapes.) Next, we ran our floorplanner / pin assigner on the benchmark using the shape lists specified in [13] and obtained an average 7.5% reduction in the layout area and 25% reduction in the total interconnection length. Finally, we introduced the net-length constraints for the 16 critical nets in the *Xerox* benchmark. For a number of chip aspect ratios, our program generated floorplanning solutions with minimum area and planar power and ground routing which satisfied *all* the constraints. The run time for floorplanning the benchmark was about 5 minutes.

6 Conclusions

We have presented a hierarchical technique for floorplanning and pin assignment of general cell layouts. Our technique avoids cell or net ordering problems. Floorplanning, pin assignment and global routing influence one another during the hierarchical steps of the algorithm. Floorplanning determines positions and shapes of cells, sets the channel topology and assigns capacities to routing regions. Pin assignment sets positions of the I/O pins for the next lower level of hierarchy. The algorithm is quite flexible and allows for various user specified constraints, e.g, pre-specified pin locations, feedthrough pins, length-critical nets and planar net topologies for power and ground routing. The same pin assignment technique can also be applied to floorplanned circuits in a flat fashion.

Acknowledgement - The authors wish to thank George Carvalho who has helped with the implementation of the system.

References

[1] N.L. Koren, "Pin Assignment in Automated Printed Circuit Board," *Proc. 9-th Design Automation Workshop*, 1972, pp.72-79.

[2] H.N. Brady, "An Approach to Topological Pin Assignment," *IEEE Trans. on Computer Aided Design*, vol CAD-3, 1984, pp.250-255.

[3] X. Yao, M. Yamada and C.L. Liu, "A New Approach to the Pin Assignment Problem," *Proc. 25-th Design Automation Conference*, 1988, pp. 566-572.

[4] X. Yao and C.L. Liu, "Pin Position Assignment for Movable Pins in Macro-Cells," to appear in *Int'l Journal Computer Aided VLSI Design*, 1990.

[5] J. Cong, "Pin Assignment with Global Routing," *Proc. Int'l Conf. on Computer Aided Design*, 1989, pp. 302-305.

[6] M. Pedram and B.T. Preas, "A Hierarchical Floorplanning Approach," to appear in *Proc. Int'l Conf. on Computer Design* 1990.

[7] W.M. Dai, B. Eschermann, E.S. Kuh and M. Pedram, "Hierarchical Placement and Floorplanning for BEAR," *IEEE Trans. on Computer Aided Design*, vol CAD-8, no 12, 1989, pp.1335-1349.

[8] Y. Ogawa, M. Pedram and E.S. Kuh, "Timing-Driven Placement for General Cell Layouts ," *Proc. Int'l Symposium on Circuits And Systems*, vol. 2, May 1990, pp. 872-875.

[9] J.K. Ousterhout, "Corner Stitching: A Data Structuring Technique for VLSI Layout Tools," *IEEE Trans. on Computer Aided Design*, vol CAD-3, 1984.

[10] W.-M. Dai, H.H. Chen, R. Dutta, M. Jackson, E.S. Kuh, M. Marek-Sadowska, M. Sato, D. Wang, and X.M. Xiong, "BEAR: A New Building-Block Layout System," *Proc. Int'l Conf. on Computer Aided Design*, 1987, pp. 34-37.

[11] R.E. Burkhard and U. Derigs, "Assignment and Matching Problems: Solution Methods with Fortran Programs," Springer Verlag, 1980.

[12] M. Marek-Sadowska, "Route Planner for Custom Chip Design," *Proc. Int'l. Conf. on Computer Aided Design*, 1986, pp. 246-249.

[13] "General Cell Floorplanning Benchmarks," *MCNC Int'l Workshop on Layout Synthesis*, Research Triangle Park, NC, 1990.