Floorplanning of Hierarchical Layout in ASIC Environment

Hossein Modarres, Susan Raam, and Jiun-Hao Lai

LSI Logic Corporation, 1551 McCarthy Blvd, Milpitas, CA 95035

ABSTRACT

This paper describes a novel floorplanning approach which can be used in implementing complex designs which are defined in a hierarchical manner. Design-specific constraints are communicated to the floorplanner through a hierarchical structure. The presented approach can then be applied to automatically floorplan the chip while taking performance and feasibility issues into account simultaneously.

INTRODUCTION

Floorplanning is the process of partitioning the chip area into rectangular blocks and assigning each block to a functional module such that an optimum design is obtained. The degree to which a design is optimized is described partly by the performance of the final integrated circuit chip and partly by feasibility and cost-related issues such as the chip area, the total interconnection length, and the routability of the design.

From the mathematical point of view, both formulating and solving the floorplanning problem are quite difficult. A precise mathematical solution essentially cannot be obtained. Heuristic procedures are therefore employed in obtaining an approximate solution to the problem. Several methods have been employed in solving the floorplanning problem. Among such methods are constructive,[1, 2] Monte Carlo,[3, 4] nonlinear programming,[5, 6] dual graph formation,[7, 8] hypergraph projection,[9, 10] force-directed,[11, 12] various clustering, merging, and rearranging,[13, 14] and minimum cut and slicing methods.[15, 16]

Floorplanning is traditionally done at the chip level consisting of a few functional blocks.[17, 18] With the advent of application specific integrated circuit technologies, however, we may not only have hundreds of functional modules in a design, but each designer may have a different set of performance considerations. At the same time, the layout procedures, such as placement and routing software, are, however. mainly concerned about routability and feasibility of the design.

HIERARCHICAL LAYOUT METHODOLOGY

The structure of a complex design can be expressed in a hierarchical manner. The hierarchical structure, not only simplifies representation of a complex design but it can also be used to express the understanding of the designer as to how different functional modules should be put together to satisfy the performance requirements. The modules which are to be clustered together can be defined as a node in the hierarchy tree and in a recursive way (Figure 1). Such a structure gives the floorplanner the flexibility of making global decisions and at the same time controls the delays across the chip by enforcing the proper clustering of modules.

Leaf nodes in the hierarchy represent laid out megacells or collections of gates in standard-cell or sea-of-gates technologies. Megacells have fixed dimensions and are possibly rotatable, while a collection of gates can have various alternate shapes. Such flexibility can be used, as well, at the floorplanning stage to achieve a more optimum design.

FLOORPLAN DESIGN SYSTEM

The floorplanner described here operates on a design structured in a hierarchical manner as explained above. The designer can do feasibility checking and obtain predicted postlayout data by using the floorplanner as the design proceeds. The floorplanner can be used to do floorplanning at any node of the hierarchy tree and for any number of levels. A particular branch of the tree, representing part of a design, or the whole tree can be automatically floorplanned. The floorplanner can be used prior to i/o assignment or after that; in the latter case, the connections to placed i/o pads are taken into account in search of an optimum floorplan. The novel approach taken here considers all of the connections across all of the branches of the tree into account in obtaining a globally optimum solution. Similarly, the shapes of all of the branches are considered simultaneously in obtaining the final floorplan. In other words, different levels of the hierarchy tree are *not* independently floorplanned; rather, the cost function takes all of the global interactions across the branches of the tree, as well as connections to i/o pads, into account.

It usually takes a few minutes to automatically floorplan a rather complex design. The resulting floorplan would then be displayed graphically. The user is informed of the routability of the chip for the given chip size. Several graphics commands are available to observe the interconnectivity of modules across the hierarchy tree, possibly above a user defined threshold. At the same time, the resulting floorplan can be passed over to a delay prediction procedure which in turn would return the expected delays associated with different nets in the given floorplan. Such delays can be displayed and the ones which are above a desired threshold can be highlighted. Various interactive graphics commands are available for possible manual manipulation if desired. Manual manipulation of the floorplan is thus possible for the curious user who wishes to further massage the floorplan, possibly for investigating the effect of such modifications in achieving a possibly better performance and/or feasibility number. For any new floorplan, a routability number and a set of expected net delays are computed in real time. Figure 2 illustrates the architecture of the floorplanner in a simplified form.

CH2584-1/88/0000-0033 $1.00 © 1988 IEEE

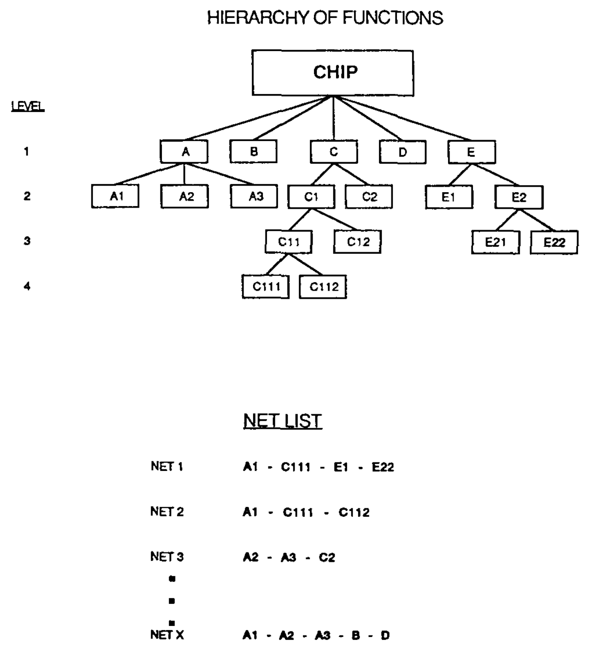

Figure 1. The simplified representation of a design shows the functions in a hierarchical structure with 4 levels of hierarchy and a total of 18 modules.

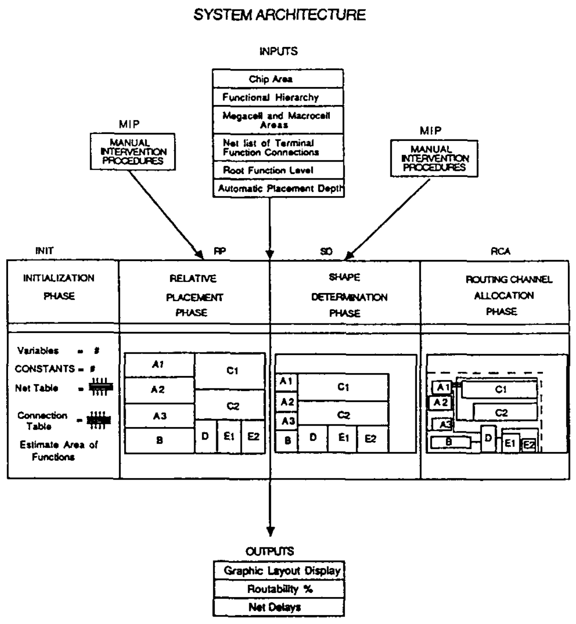

Figure 2. The simplified architecture of the floorplanner LPACE.

HIERARCHICAL FLOORPLANNING APPROACH

The hierarchical design methodology described above is an appropriate way of treating complex designs and communicating design-dependent requirements with the floorplanner. The adaptation of most floorplanning methods to the hierarchical design methodology is very difficult. Some of these methods are too slow to be able to be used in an interactive way, some others are too complex to handle a large number of functional modules properly, and yet others would result in a sparse floorplan unacceptable in high density and complex designs or lack the capability of having a global perspective in a hierarchical design environment. The objective of obtaining a tight design in the presence of a large number of functional modules is often uncompromising. Also, the fact that there are usually a large number of soft functional modules in an ASIC-based design would put the burden of the determination of the proper aspect ratio for each soft functional module on the floorplanner.

Considering the above facts and objectives, using a partitioning-based heuristic along with a slicing structure seem to be the proper choice. The combination of these two methods is not only quite fast but would often result in a very tight floorplan in the presence of a sufficient number of soft functional modules. The slicing method can also be used to simultaneously determine the proper aspect ratios for the soft functional modules in obtaining the overall global optimum. This would also prevent the floorplan from requiring any final block compaction procedure.

In our approach, the hierarchy tree is traversed several times, each time performing a subtask. Different steps can be summarized as follows.

Module Area Estimation

In the first pass, the area of each functional module is estimated. The area of a laid out megacell is known. The area of a functional module of cells in standard-cell or sea-of-gates technology can be estimated by predicting its postlayout routing channel track requirement based on which the area can be computed. As the number of gates in a design increases, the logic becomes more and more self-contained. As a consequence, the rate of routing channel track requirement in an array of cells decreases. In other words, the average channel track density follows a Rent-like rule of the form $a\,C^b$, where C is the gate complexity of the module, and a and b are functions of various layout parameters. This represents a family of curves for routing channel track requirement formed by the performance of placement and routing procedures and layout complexity measures such as pin density and average pin per net in a module.

Partitioning and Slicing

In this step, the hierarchy tree, or part of it which is to be floorplanned, is traversed in preorder. Each node of the tree is recursively partitioned and sliced using a graph partitioning approach. The cost function which is to be minimized in the partitioning process represents the cut-size as well as the balance of area on both sides of the cut-line:

$$C(a,b) = k_1.S(a,b) + k_2.\frac{|A(a)-A(b)|}{A(a)+A(b)}.$$

$S(a,b)$ is the cut-size of partitions a and b, and $A(a)$ and $A(b)$ are the total area of the partitions a and b respectively; k_1 and k_2 are constants. Furthermore, the cost function is modified to represent the connections to other branches of the tree as well as to i/o pads using a simplified and faster version of terminal propagation and in-place partitioning.[19, 16] At each step of the partitioning process, the set of modules are partitioned into two sets (bottom and top, or left and right) to achieve a minimum cost. If the number of modules is 15 or less, an exhaustive search for the optimum partition is made, otherwise a modified version of the Kernighan and Lin heuristic is used.[20] The direction of the cut line is chosen at each step dynamically to give a better result in terms of the area usage. At the same time a slicing tree is created for each node to represent the relative location of modules (Figure 3).[21] At the outset of this phase, thus, a tree of slicing trees are formed.

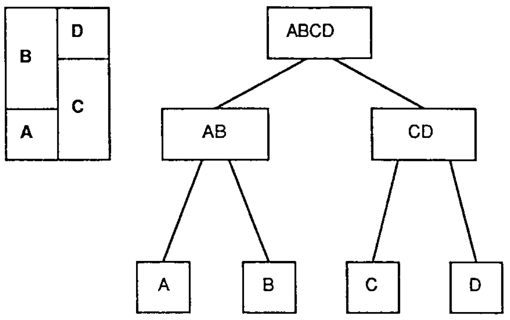

Figure 3. The slicing tree corresponding to a single node in the hierarchy with 4 functional modules.

Shape Function Formation

The hierarchy tree is then traversed in postorder to form a shape function for each slice and thus for each node in the hierarchy tree. The shape function has the advantage of representing all possible shapes of each individual module in a single algebraic form. The shape function of a fixed-area module is a hyperbola of the form $y = k/x$. For an array of cells, it can be shown that the routing congestion is expected to occur in the midsection of the functional module and is relatively independent of the aspect ratio of the module and the size of each individual cell in the array. [22] This means that a shifted hyperbola of the form $y = k/(x-a)+b$ best represents the shape function of a module of standard-cell or sea-of-gates cells (Figure 4). The shape function of a non-leaf node in the slicing tree can be obtained by the addition of the shape functions or their inverses of the constituent slices depending on how the parent slice is formed. The final result of this stage is a monotonic shape function for the root node of the hierarchy tree which can represent the whole chip or the part which is being floorplanned.

Shape Determination

The hierarchy tree is once again traversed, this time in preorder, to assign a particular shape and orientation to each individual module in the hierarchy tree. The shape function of the top module represents all possible shapes for that module based on all possible shapes and orientations of the leaf functional modules. A particular point on this shape function represents a particular shape for the top module. Such a point is selected to give a minimum area and an aspect ratio of close to 1. Given a point on a shape function, it is possible to find a unique set of points on the constituent slices (corresponding to the shapes of the constituent slices). This procedure can be repeated recursively to determine the shapes of all modules in the hierarchy tree. It is worth mentioning that due to the hierarchical structure adapted here, all tree traversals are in fact nested. That is, when the hierarchy tree is traversed, at each node, the corresponding slicing tree should be traversed as well. At the termination of this stage, each module in the hierarchy has a unique shape assigned to it.

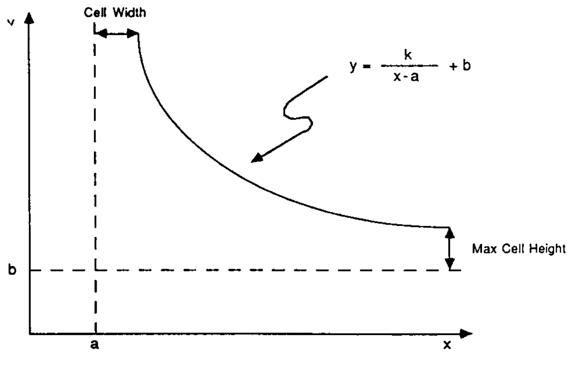

Figure 4. Shape function of an array of cells in columns represents all possible shapes for the area accommodating the module in the floorplan.

Intermodule Routing Space Estimation

Once again the hierarchy tree is traversed in preorder to estimate the routing space needed for intermodule traffic. A statistical routing approach is taken here for simplicity, accuracy, and to assure a uniform distribution of routing traffic. [23] The nets are considered one by one. It is assumed that a net is routed inside its bounding box. All possible routes for the net are considered by scanning the bounding box from bottom to top and from left to right; at each step a weighted channel requirement is associated to each possible route.

Intermodule Routing Space Allocation

The hierarchy tree is traversed two more times, once in preorder and once again in postorder, to allocate the routing channels by updating the shape functions formed in the previous steps and then updating the shapes and coordinates chosen for each individual module. The amount of space that can be allocated without the size of the top module going beyond that of the given chip size is an indication of the routability of the chip.

Interactive Layout Manipulation

There are cases for which the user has to manipulate the floorplan through interactive graphics commands to improve the feasibility or the performance of the chip. An extensive set of interactive graphics commands has been developed to edit the floorplan, to make queries, or to modify the display window. The editing commands include those which can change the size of a functional module (e.g., ***magnify***, ***shrink***, ***fit***, and ***bound*** commands), its shape (the ***aspect ratio*** command), or its position (***place***, ***move***, ***rotate***, and ***flip*** commands). The query commands allow the user to analyze the floorplan for routability and/or gate utilization. The display commands allow the user to highlight delay violations or the degree of interconnectivity of functional modules and pads. Real time feedback on routability of the chip or net delays can be provided.

IMPLEMENTATION AND PERFORMANCE

The collection of algorithms and heuristics described here has been implemented in the LSI Logic's Planning And Chip Evaluation (LPACE) software package. The program is implemented in the C language and runs on a number of different platforms including Sun/UNIX, VAX/VMS, and IBM/CMS environments. The software can be run in batch mode or in an interactive fashion. A wide selection of commands are available in interactive mode to run the autoplacement, evaluate the floorplan, or do manual manipulation. The performance of the software is dominated by the partitioning procedure and can thus be expressed as $O(m.n^2\log(n))$, where m is the number of levels being floorplanned and n is the maximum number of modules in each level ($m.n$ roughly represents the total number of modules in the hierarchy). Table 1 gives some empirical data about the performance of the software.

SUMMARY AND CONCLUSIONS

With the continued advances in IC fabrication technology, the integrated circuit chips with up to 100K used gates are being designed and simulated in the *Modular Design Environment* (MDE) of LSI Logic Corporation. The designers have to be concerned about performance and feasibility issues at various stages of the design process. The hierarchical design methodology can be used to control delays

Chip	Total No. of Modules	No. of Levels	Automatic Floorplanning Time (Secs)
1	60	2	21
2	200	4	73
3	335	4	222
4	509	7	180
5	771	4	341

Table 1. Performance of the software in automatic floorplanning of a hierarchical design. A balanced hierarchy would significantly speed-up the floorplanning time. The automatic floorplanning times for the whole hierarchy are measured in CPU seconds on an IBM 3090.

and simplify design representation. Such a hierarchical structure can be used by the floorplanner LPACE to feedback feasibility and performance information about the design. Each functional module in the hierarchy is a small part of the chip, further details of its placement and routing are better handled by layout design system.

REFERENCES

[1] Preas, B.T., and W.M. van Cleemput. 1979. "Placement Algorithms for Arbitrary Shaped Blocks," *16th Design Automation Conference*, San Diego, CA.

[2] Horng, C.S., and M. Lie. 1981. "An Automated Interactive Layout System for Arbitrary Sized Rectangular Building Blocks," *18th Design Automation Conference*, Nashville, TN.

[3] Kirkpatrick, S., C.D. Gelatt, Jr., and M.P. Vecchi. May 1983. "Optimization by Simulated Annealing," *Science*.

[4] Jepson, D.W., and C.D. Gelett, Jr. 1983. "Macro Placement by Monte Carlo Annealing," *IEEE International Conference on Computer Design*, Port Chester, NY.

[5] Markov, L.A., J.R. Fox, and J.H. Blanks. 1984. "Optimization Techniques for Two-Dimensional Placement," *21st Design Automation Conference*, Albuquerque, NM.

[6] Sha, L., and R.W. Dutton. 1985. "An Analytical Algorithm for Placement of Arbitrary Sized Rectangular Blocks," *22nd Design Automation Conference*, Las Vegas, NV.

[7] Heller, W.R., G. Sorking, and K. Maling. 1982 "The Planar Package Planner for System Designers," *19th Design Automation Conference*, Las Vegas, NV.

[8] Leinman, S.M., and Y.T. Lai. 1984. "An Algorithm for Building Rectangular Floor Plans," *21st Design Automation Conference*, Albuquerque, NM.

[9] Hall, K.M. November 1970. "An r-Dimensional Quadratic Placement Algorithm," *Management Science*.

[10] Fukunaga, K., S. Yamada, H.S. Stone, and T. Kasai. April 1984. "A Representation of Hypergraphs in the Euclidean Space," *IEEE Transactions on Computers*.

[11] Quinn, N.R., Jr., and M.A. Breuer. June 1979. "A Force Directed Component Placement for Printed Circuit Boards," *IEEE Transactions on Circuits and Systems*.

[12] Antrich, K.J., F.M. Johannes, and F.H. Kirsch. 1982. "A New Approach for Solving the Placement Problem Using Force Models," *IEEE International Symposium on Circuits and Systems*.

[13] Chen, C.C., and E.S. Kuh. 1984. "Automatic Placement for Building Block Layout," *International Conference on Computer-Aided Design*, Santa Clara, CA.

[14] Dai, W., and E.S. Kuh. 1986. "Hierarchical Floor Planning for Building Block Layout," *International Conference on Computer-Aided Design*, Santa Clara, CA.

[15] Lauther, U. 1980. "A Min-Cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation," *Journal of Digital Systems*, Vol. IV, Issue 1.

[16] LaPotin, D.P., and S.W. Director. 1985. "Mason: A Global Floor-Planning Tool," *International Conference on Computer-Aided Design*, Santa Clara, CA.

[17] Woo, L.S., C.K. Wong, and D.T. Tang. August 1986. "PIONEER: A Macro-Based Floor-Planning Design System," *VLSI Systems Design*.

[18] Macaluso, E. April 1987. "Graphical Floorplan Design of Cell-Based VLSI Circuits," *VLSI Systems Design*.

[19] Dunlop, A.E., and B.W. Kernighan. January 1985. "A Procedure for Placement of Standard-Cell VLSI Circuits," *IEEE Transactions on Computer-Aided Design*.

[20] Kernighan, B.W., and S. Lin. February 1970. "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell System Technical Journal*.

[21] Otten, R.H.J.M. October 1982. "Layout Structures," *IBM Research Report*, RC 9657, Yorktown Heights, NY.

[22] Sutherland, I.E., and D. Oestreicher. May 1973. "How Big Should a Printed Circuit Board Be?" *IEEE Transactions on Computers*.

[23] Ueda, K., H. Kitazawa, and I. Harada. January 1985. "CHAMP: Chip Floor Plan for Hierarchical VLSI Layout Design," *IEEE Transactions on Computer-Aided Design*.