Hierarchical Global Routing Integrated with Floorplanning and Placement

Morteza Saheb Zamani and Saeed Kazemi
IT and Computer Engineering Department, Amirkabir University of Technology

Abstract

Physical design tasks are very complex and therefore, are usually decomposed into several simple subtasks. However, the interdependence between these subtasks and its effect on the quality of the design is increasingly becoming more obvious. On the other hand, the increasing complexity of systems prevents the integration of the physical design subtasks.

This paper presents an integrated approach to the floorplanning, placement and routing of very large macrocell-based designs. The framework is given a hierarchical specification and performs the physical design subtasks while preserving the specification hierarchy. There are some challenging problems in this approach mainly due to the high degree of interdependence between levels in the specification hierarchy which makes pure 'divide and conquer' strategies useless. A hierarchical global routing algorithm is also introduced in this paper.

1. Introduction

With the growing complexity of VLSI systems, the physical design of today's systems has become very complex. One approach to control overall complexity, which is usual in designing in higher levels of abstraction, is to divide the design into several levels of hierarchy. This produces poor results if the interaction between modules at different levels of hierarchy is not considered.

Additional complexity has recently been introduced by the deep submicron technology due to the large impact of physical parameters on the quality of the final design. This usually leads to many iterations between the processes over various levels of abstraction (e.g. high-level synthesis and physical design). The cost of iterating between these processes can be very high in practice if the layout result cannot provide helpful information to the higher level synthesis process. One effective way to provide bilateral information between these various design processes is to use a common specification hierarchy. This can facilitate incremental modification of the design while avoiding inconvergence in the design process.

However, the high degree of interdependence between levels in the specification hierarchy may result in poor layout if pure top-down or pure bottom-up algorithms are exploited. In the former approach, even though a good global interconnection view is available at the top of hierarchy, there is no information about module sizes/shapes since the placement

and routing internal to the modules has yet to be determined. On the other hand, algorithms starting from the leaf cells have the advantage of using accurate size, shape and pin position information about the modules but since they are unable to use the connectivity information of modules at the upper hierarchy levels, an optimal placement at one hierarchy level is unlikely to be optimal with respect to the higher levels.
The algorithm presented in this paper utilizes the advantages of the top-down and bottom-up strategies by quickly traversing the specification hierarchy in three passes to perform floorplanning, placement and routing.
In this approach, the algorithm first gathers appropriate information about the module sizes, pin positions and connection paths before attempting the actual placement and routing. The algorithm considers the effect of the hierarchy levels on each other as well as routing information during floorplanning.

2. Prior Works

There has been a large amount of research in the field of *single-level* floorplanning over the past decades [1] [2] [3] [4] [5] [6]. However, multi-level hierarchical floorplanning and routing of macrocell designs and the inter-dependence among the hierarchy levels have not been studied closely. One of the main reasons why this problem has not been addressed is that macrocell-based designs have not had many modules until recently and the difficulty in handling inter-level dependencies might have hindered CAD designers to adopt this strategy.
However, in the recent years, due to several reasons, including growing complexity of designs and the need for design reuse, the number of macrocells in a design is becoming very large. Therefore, physical design tools have to handle very complex designs with many hard macros [7].
However, there are few approaches which attempt to perform floorplanning hierarchically. [8], [9] and [10] enumerate all possible topologies of every cell at each hierarchy level and then select those that best satisfy the objective function for placement and routing within a module incorporating the cells. Dai and Kuh's algorithm[10] tries all possible templates in a top-down process to find the best topology according to a desired chip aspect ratio and I/O pad positions. However, since this approach fails to take module shapes into account at the higher hierarchy levels, it does not work well for chips containing cells with fixed geometries and it may need considerable backtracking to effectively search the solution space. Eschermann et al [8] propose an improvement on this approach which passes shape estimation information up the hierarchy levels from the bottom. This information is then utilized by a process operating top-down, to evaluate the various topological possibilities (TPs). This bottom-up process largely constrains the search space since it only computes the shape of one configuration at each level, by considering such measures as area and shape, and does not carry enough information about the shapes of the modules from the bottom levels. Pedram and Preas [9] propagate accurate information by enumerating all TPs in a bottom-up traversal and labeling each module with its shape function (the function which gives the possible dimensions a module can have). Some other techniques, such as [11] and [12] also use shape functions to pass size information to upper levels of the hierarchy.
These approaches are limited by the number of modules at each level, as well as the number of hierarchy levels, able to be considered, due to the fact that the number of TPs grows exponentially with the number of modules. For example, a five-module cell may have more than 300,000 different TPs [8].

This paper presents a fast algorithm which is capable of floorplanning followed by placement and routing of very large hierarchical designs. It considers the effects of the hierarchy levels on each other and takes routing into account during the floorplanning process. A global routing strategy is also presented in this paper.

3. Floorplanning and Placement

Our system is given a design hierarchy consisting of many levels, at each of which the modules' netlists are specified. The size and pin positions of all leaf cells in the design are assumed to be fixed. The position of I/O ports in the upper most level may be given as constraints or left unspecified. In the latter case, the algorithm will attempt to place them optimally. The objective of the algorithm is to find the positions and orientations of the leaf modules, as well as the shapes and pin positions of the parent modules, while preserving the specified hierarchy, in such a way that the combined area and signal path length of the circuit is minimized.

The algorithm consists of four recursive processes. In the first traversal, the size and shape of each module within each hierarchy level is regarded as equal because at the top level of the specification hierarchy there is no information on the geometry of the modules. It performs an initial placement of modules at each level based on the I/O port positions and the internal and external connectivities of the module at that level. This is followed by the assignment of I/O ports on each placed module's boundary, in an attempt to optimally route all interconnections at the parent module level. The process now recurs for each of the placed modules. This top-down process produces an initial floorplan containing the global connection paths for all modules in the design, which are locally optimum.Based on this information, a bottom-up process may be performed to produce an estimation of the module sizes, successively, at every level in the specification hierarchy. This is followed by another top-down process which gives a more accurate global view of connections, by using the more realistic module size information calculated in the previous pass. The next bottom-up process then uses the information gathered in the previous passes to perform the actual placement and routing. In the following, each process is explained in detail.

3.1 Initial floorplanning: This step is to find optimal port positions for each module to obtain interconnection paths throughout the design. To do this, it is necessary to determine the relative positions of modules within each parent module. Since module size information is unavailable initially, all modules at each hierarchy level are assumed to have equal size (or sizes based on a rough estimation).

In this stage, we consider modules to be circles, since there is no information available on size and shape. Based on fixed module sizes and port configurations (at a given hierarchy level), it is possible to find the optimal position of each submodule's ports through which connections are made to other submodules (at this level) and to the parent module. Representing a module as a circle also avoids the need to fix pins to one of the four sides of a rectangle at an early stage. The simplicity of the algorithm for finding an approximate placement of modules, guided by locally optimum wiring, is another reason for representing modules as circles, in this stage.

Using the circle representation, the task of finding the relative positions of modules at each level can be viewed as an optimization problem with an objective function which maximizes the sizes of submodule circles within a parent module circle, where each module has a known size and port configuration, in such a way that the total connection length within each hierarchy level is minimized. This problem may be solved by using classic optimization techniques with a composite objective function [13]. However, a heuristic method, explained in section 3.5, is used in our system to speed up processing.

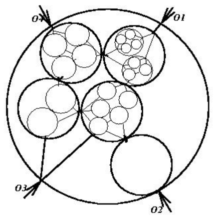

Figure 1: Connection path produced by initial floorplanning.

The first top-down process starts at the root module of the specified hierarchy, where port positions on the boundary correspond to desired I/O pad positions. After placing all submodules and calculating their maximum allowed size relative to the parent module, optimal port positions, for each module of this parent, are determined by connecting the centers of the corresponding circles. This procedure is applied recursively to each set of modules at each hierarchy level, until the leaf modules are determined. Routing paths can now be determined at each hierarchy level, as pin position information is passed from the top to the lower levels of the design. Figure 1 shows the result of the process for a 4-level hierarchy.

3.2 Size estimation and propagation: From the leaf modules, accurate information about sizes is available. The first bottom-up traversal attempts to propagate size information to the upper levels of the tree. Each module is modeled by its minimum enclosing circle, where the area of the circle is the minimum space required for a module without having to fix its orientation at this early stage.

Starting from the leaves, the size of each parent module is estimated based on the relative positions of its submodules. These are calculated from the placed circles (modules), having fixed the external port positions dictated by the interconnection paths obtained during the initial floorplanning process. These port positions are described in terms of the angles at which the wires connect to the module, and are, therefore, independent of the module's size. These are used, in turn, to find an optimal placement of submodules inside each parent module.

The placement procedure in this traversal is similar to the one used in the floorplanning process, but with unequal bounding circle sizes and a scaling postprocess which uses real submodule sizes (see Section 3.5 for details). The size of each parent module is then determined as the minimum circle enclosing all its submodule circles. By applying this procedure recursively in traversing back up the specification hierarchy, the sizes of all parent modules in the design are estimated.

3.3 Floorplanning improvement: The information obtained during the floorplanning process is rather inaccurate due to the unavailability of module size information. With estimated sizes available after the first bottom-up traversal, a process similar to initial floorplanning is carried out with more realistic module sizes to obtain more accurate interconnection path estimates.

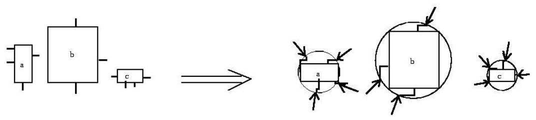

Figure 2: Setting block orientations

3.4 final Placement: At this stage, where information about the wiring paths, port positions, and module sizes are available, the final placement is performed using rectangular blocks. The algorithm for determining the relative placement of circles is based

on the last calculated pin positions. The rectangular blocks are then mapped into their corresponding circles. The mapping also involves determining the best orientation (reflection and 90º,180º, and 270º rotations) of the blocks, determined by enumerating the eight possible ways of positioning each block inside its corresponding circle and choosing the orientation which results in minimum length routing, when the actual ports of the blocks are connected to the port positions around their enclosing circles (Figure 2). In this way, the block orientations can be found in linear time with respect to the number of modules at each level. Then, in order to improve the placement we use a method based on simulated annealing [14], with a restricted move set and low starting temperature which preserves the relative placement and results in fast compaction.
The routing for this module can now be determined (see section 4). The minimum bounding circle of each compacted rectangular module has now been determined for the parent hierarchy level, and the procedure (placement and routing) is recursively applied (upward) until the final layout of all modules is determined.

3.5 Placement Algorithm for One Level of Hierarchy: The heuristic used here to determine the optimal placement of circles is based on the force-directed method [15]. The algorithm first finds the positions of module centers by considering each connection as an attractive force.

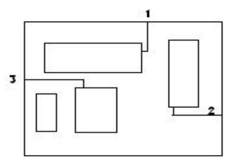

Figure 3: Pin assignment for external nets based on closest point on the boundary.

4. Routing

In the final bottom-up traversal, each module is routed before the placement of the upper-level modules is performed. For the global routing of each module, the routing space is divided into maximal horizontal rectangles (vacant tiles). Then, a routing graph is constructed whose vertices and edges correspond to the vacant tiles and their neighborhoods, respectively. The edge weights are assigned according to the available space (regarding the initial space and routing congestion). A shortest path algorithm [16] is used for the global routing of modules at each hierarchy level.
There are several problems in a hierarchical environment as follows:

a) External nets: In addition to the internal nets inside a module, there are many external ones whose pin positions have not been fixed yet. Three strategies were tried. The position of a pin (method A) for an external net can be set to the point on the boundary which is closest to the other terminal(s) of this net (Figure 3).

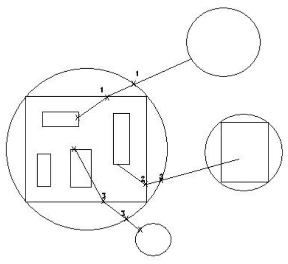

Figure 4: Use of gathered information from previous traversals in assigning external nets exit points.

Another method (method B) is to use the pin position information gathered during the previous traversals of the hierarchy. This method should be more effective since it uses more global information about the interconnection structure of the whole design (Figure 4).

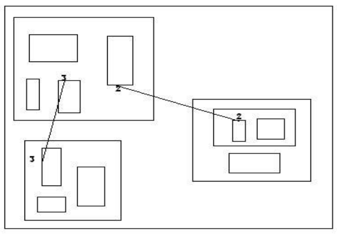
Figure 5: Routing of external nets can be postponed until their pin position are found.

The third method which is even more effective is to postpone the routing of the external nets until the position of all other terminals to which they are connected are known during the placement and routing of other modules in the hierarchy. In this way, the routing of such nets may be performed at the higher levels of hierarchy when their parent (or predecessors) are being routed. These pins may then be located somewhere inside the higher-level modules (Figure 5). This prevents wires to be unnecessarily routed around some modules. The experimental results in Section 5 shows the effectiveness of each strategy.

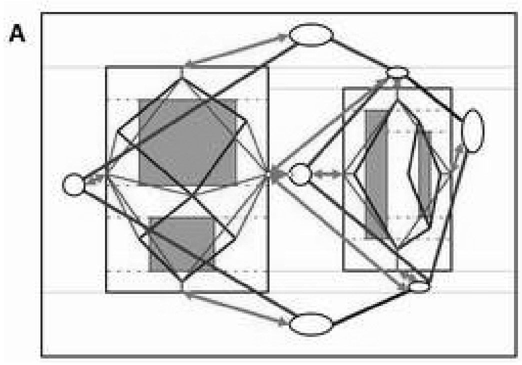

Figure 6: Extended routing graph

b) Routing through modules: When the algorithm is routing a module at a hierarchy level, a net may unnecessarily be routed around some blocking module(s) if all the modules are considered as rigid. However, there are sometimes some space available inside the modules and therefore, can be used for routing. We use this space by including the routing graph of the blocking module into the routing graph of the current hierarchy level (Figure 6).

c) Routing area estimation: Accurate estimation of routing area during placement is crucial. Two kinds of approaches were attempted in this work, namely static and dynamic estimation.

Static estimation: routing space can be estimated on a preprocessing step before placement. In this method, routing space is estimated and added to block dimensions and then SA is applied to these new blocks.

Dynamic estimation: Routing space can be calculated during the placement. In this method, routing space for each block is calculated at each iteration of SA algorithm, based on current placement of modules.

For static estimation, two methods were tried in our framework. BEAR-FP[8] adds routing area on each side of a module proportional to the number of ports on that side. CHAMP[17], on the other hand, finds all module sides coinciding the minimum bounding

box of each net and distributes routing space for that net among such module sides. This can be done as a preprocessing step (static) or calculated at each iteration (dynamic).
Murata et al. [18] calculates routing space for a parent module based on the position of nets within it. Therefore, the routing space depends on the current placement of submodules and, therefore, is estimated dynamically.

5 Experimental results

Since there are no hierarchically specified benchmarks for floorplanning, placement and routing, three MCNC benchmarks, i.e. ami33, ami49 and primary1[19] were partitioned using the partitioner introduced in [10]. In addition, we produced a large benchmark containing 10,000 modules (sample10000) and 24,435 nets generated by gnl-1.1 [20].
In order to compare our results with other works, we applied the algorithm in [9] to the hierarchical designs. Table 1 shows that our approach is both faster and more effective in terms of area and total connection length.

Table1: Comparison with BEAR-FP

Bench-mark	BEAR-FP		Our framework		
	Area (mm2)	Net length	Area (mm2)	Net length	Time improve-ment
Ami33	1.67	66902	1.68	66982	90%
Ami49	43.1	809376	44.19	1,075,551	200%
Primary1	Did not finish		37.66	2,538300	

Table 2 shows the reduction in total connection length of external nets. As the results show, using routing information gathered in the three hierarchy traversals can reduce routing length by up to 10%. Table 2 also shows that deferring pin assignment for external nets until the positions of their terminals are determined, can improve results further, especially for large designs.
Routing through the modules at the lower levels of hierarchy (Figure 6) were also embedded into the framework. Table 3 shows that this can improve the routing length of external nets with acceptable process speed reduction (the experiments were done on a system with a 850 MHz AMD Duron and 256MB RAM).
To compare the effectiveness of the different approaches for routing area estimation in our hierarchical framework, we embedded two static (BEAR-FP and CHAMP) as well as two dynamic (dynamic CHAMP and Murata) methods into the algorithm. Table 4 shows the improvement in terms of total area and total connection length of each example after successful global routing compared with BEAR-FP. The table shows that dynamic methods give better results. This is achieved at the expense of small running time (<15%).

Table 2 – External nets length improvement of methods B and C compared with method A regarding external net routing

Benchmark	Routing strategy	
	B	C
Ami33	6.72%	6.82%
Ami49	6.68%	6.97%
Primary1	7.53%	9.42%
Sample	8.28%	10.03%

Table 3 – Improvement by routing through modules

Benchmark	External net length improvment	Time(sec) without feedthroughs	Time(sec) With feedthroughs
Ami33	9.35%	2	3
Ami49	9.79%	5	7
Primary1	12.64%	58	82
sample10000	18.16%	520	890

Table 4- Area and total routing length improvement with respect to BEAR-FP method.

Benchmark	Static CHAMP		Dynamic CHAMP		Murata	
	area	Net length	area	Net length	area	Net length
Ami33	-2.92%	+0.08%	+2.24%	+1.68%	-2.19%	-2.59%
Ami49	-2.05%	-1.43%	-1.48%	-0.05%	-0.09	-1.31%
Primary1	-1.24%	-1.82	+5.56%	+3.80%	+5.17%	+2.97%
sample10000	-3.78%	-4.45%	+7.59%	+7.18%	+9.02%	+9.83%

6 Conclusion

In this paper, a hierarchical approach to the physical design of large macrocell-based designs was proposed. Floorplanning, placement and routing processes, which are normally performed in separate consecutive phases (due to the complexity of these processes), are integrated in our framework. Specification hierarchy can be preserved in this framework in order to facilitate and speed up the design cycle.

References

[1] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, 2nd ed., Kluwer Academic Publishers, 1999.
[2] A. Kahng, “Classical floorplanning harmful,” *Proc. Of IEEE Intl Symp. Physical Design*, 2000, pp. 207-213.
[3] P. Chen, and E. S. Kuh, “Floorplan sizing by linear programming approximation,” *Proc. Of IEEE Intl Design Automation Conf.* , 2000, pp. 467-471.
[4] X. Tang, and D. F. Wong, “Floorplanning with alignment and performance constraints,” *Proc. Of IEEE Intl Design Automation Conf.* , 2002, pp. 848-853.
[5] Y. Feng, D. Mehta, and H. Yang, “Constrained “modern” floorplanning,” *Proc. Of IEEE Intl Symp. Physical Design*, 2003, pp. 128-135.
[6] Y. Ma, X. Hong, S. Dong, S. Chen, Y. Cai, C.-K. Cheng, and J. Gu, “An integrated floorplanning with an efficient buffer planning algorithm,” *Proc. Of IEEE Intl Symp. Physical Design*, 2003, pp. 136-142.
[7] R. Camposano, “Physical design: The whole enchilada,” *Proc. Of IEEE Intl Symp. Physical Design*, 2003, pp. 3.
[8] B. Eschermann, W.-M. Dai, E. S. Kuh and M. Pedram, “Hierarchical placement for macrocells: A “meet-in-the-middle” approach,*” Proc. Of IEEE Intl Design Automation Conf.* , 1988, pp. 460-463.
[9] M. Pedram, and B. Preas, “A hierarchical floorplanning approach,” *Proc. Of Intl Conf. Computer Design*, 1990, pp. 332-338.
[10] W.-M. Dai, and E. S. Kuh, “Simultaneous floorplanning and global routing for hierarchical building-block layout,” *IEEE Trans. Computer-Aided Design*, vol. 6, no. 5, 1987, pp. 828-837.
[11] T. Lengauer, R. Mueller, “ Robust and accurate hierarchical floorplanning with integrated global wiring,” *IEEE Trans. Computer-Aided Design*, vol. 12, no. 6,1993, pp. 802-809.
[12] B. Schuermann, J. Altmeyer, and G. Zimmermann, “Three-phase chip planning – an improved top-down chip planning strategy,” *Proc. Of IEEE Intl Conf. Computer-Aided Design*, 1992, pp. 598-605.
[13] L.C.W. Dixon. *Nonlinear Optimization.* English Universities, 1972.
[14] S. Kirkpatrick, C.D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. Science,1983, 220 (4598):671-680.
[15] N.R Quinn. “The placement problem as viewed from the physics of classic mechanics”. *Design Automation Conference*,1975, pp. 173-178.
[16] E. W. Dijkstra, “ A note on two problems in connexion with graphs “ Number. Math., vol. 1, Oct. 1959,pp. 269-271.
[17] K. Ueda, H. Kitazawa, and I. Harada. "CHAMP: Chip floorplan for hierarchical VLSI layout design". In IEEE Transactions on Computer Aided Design, 1986, pp. 12-22.
[18] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair", IEEE Trans. on CAD, Vo1.15, No.12, 1996. pp.1518--1524
[19] http://www.cbl.ncsu.edu
[20] Peter Verplaetse, Dirk Stroobandt, Jan Van Campenhout, "Synthetic benchmark circuits for timing-driven physical design applications".In Proc. of the Intl. Conf. on VLSI, June 2002, pp. 31-37.