2003 IC/CAD Contest

Problem 1: Chip Floorplanning with Hard/Soft Macros

Source: Springsoft Inc.

Jan. 15, 2003

I. Introduction

After circuit partitioning, the area of each macro can be estimated and its shape can be determined via specifying the aspect ratio. The aspect ratio of a macro is the ratio of the width of the macro to its height. A macro is called *hard* if its dimension is known (aspect ratio cannot be changed) and a *soft* macro has an unfixed aspect ratio that can be changed within a specified range. Given several rectangular macros (soft or hard) with their core area and aspect ratio constraints, a chip floorplanner arranges all these macros within a given region.

Floorplanning plays an important role in an overall physical design cycle and mainly determines the quality of final layout such as area and performance. A floorplanner should consider many factors, such as aspect ratio, routability, timing, packaging and pre-placed macros. To simplify the problem, we assume the macros are rectangular. The aspect ratio varies between its upper bound and lower bound. The main objective is to minimize the total wire length of all nets. Since the final wiring paths are not known during floorplanning, we have to model the topology of interconnections. There are several methods to estimate the wire length, such as semi-perimeter method, complete graph, minimum chain, source to sink connection, steiner tree approximation and minimum spanning tree[1]. Here, we use the minimum spanning tree method to estimate wire length. For an *n*-terminal net, a minimum spanning tree can be constructed by determining the distances between all possible pairs of terminals, and connecting the smallest *(n-1)* edges that do not form cycles. Fig. 1[1] gives an illustration of the minimum spanning tree method.

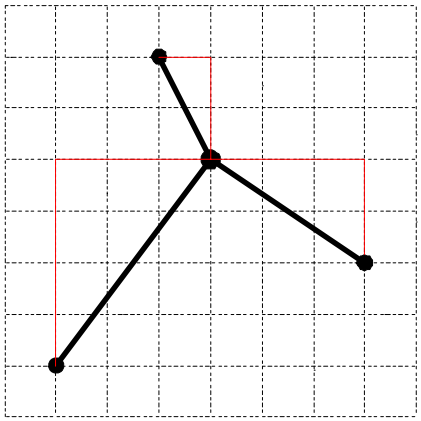


Fig. 1: Minimum Spanning Tree (length = 15) [1]

A *slicing* floorplan is obtained by recursively partitioning a rectangle into two parts in horizontal or vertical, and its *slicing tree* is a binary tree in which each leaf represents a partition and each internal node represents a cut. The figure of Fig. 2(a) is a slicing floorplan [2], and Fig. 2(b) is its corresponding slicing tree. The figure of Fig. 2(c) is a non-slicing floorplan. A floorplan is *hierarchical of order k* if it can be obtained by recursively partitioning a rectangle into at most *k* parts. We can represent a hierarchical floorplan by a

floorplan tree. Each leaf corresponds to a basic rectangle and each internal node corresponds to a composite rectangle in the floorplan. The figure of Fig. 3[2] shows a hierarchical floorplan of order 5 and its floorplan tree. The set of all slicing floorplans is an important class of hierarchical floorplans. There are four main graph models for the representation of floorplans, including polar graphs, adjacency graphs, channel intersection graphs and channel position graphs. Related articles can be found in [2][10].

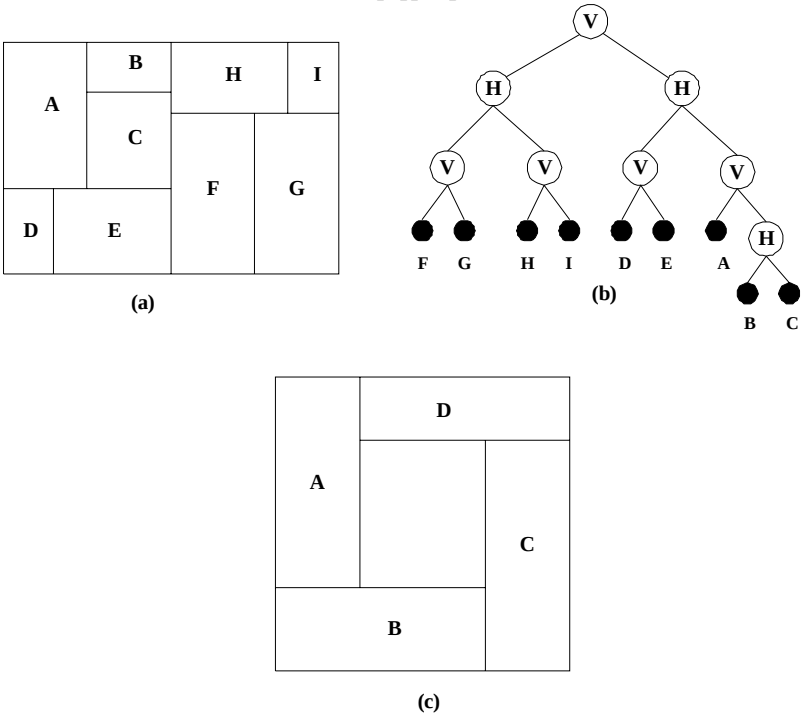


Fig. 2: A floorplan with slicing tree and a non-slicing floorplan [2]

Floorplanning algorithms[2][10] can be classified into seven classes: (1) constraint based[3], (2) integer programming based[2], (3) rectangular dualization based[5], (4) hierarchical tree based[6], and (5) iterative approaches[7]. The constraint based floorplanning algorithm constructs a floorplan of optimal area under a given set of horizontal and vertical topological (ordering) constraints. In integer programming based floorplanning, this problem is modeled as a set of linear equations using 0/1 integer variables. Two types of constraints are considered, i.e., the overlap constraints and routability constraints. The rectangular dualization based algorithms model the floorplan by converting the partition graph into its rectangular dual. The hierarchical tree based floorplanning represents a floorplan as a tree. There are two ways to generate physical hierarchy: top-down partitioning and bottom-up clustering. The simulated evloution (genetic) algorithms and simulated annealing algorithm are iterative floorplanning approaches. These iterative approaches start from an initial floorplan, and then this floorplan performs a seris of perturbations until no more improvement can be achieved. Recently there are several timing-driven floorplanning approaches [8][9] proposed to solve the performance problems.

Cluster growth [11] is a greedy approach. In the beginning, a seed macro is selected and placed into the lower left corner of the floorplan region. Then the remaining macros are selected one at a time and placed to the partial floorplan, while trying to grow evenly on upper, diagonal and right sides simultaneously. (see Fig.4(a)) The aspect ratio constraints are also maintained simultaneously. To determine the order of macro selection, the macros are initially organized into a linear order. Linear ordering algorithms order the given macro netlist into a linear list so as to minimize the number of nets that will be cut by any vertical line drawn between any consecutive macros in the linear order. A general linear ordering algorithm is shown in Fig. 4(c). Initially a seed macro is selected, then the algorithm enters a

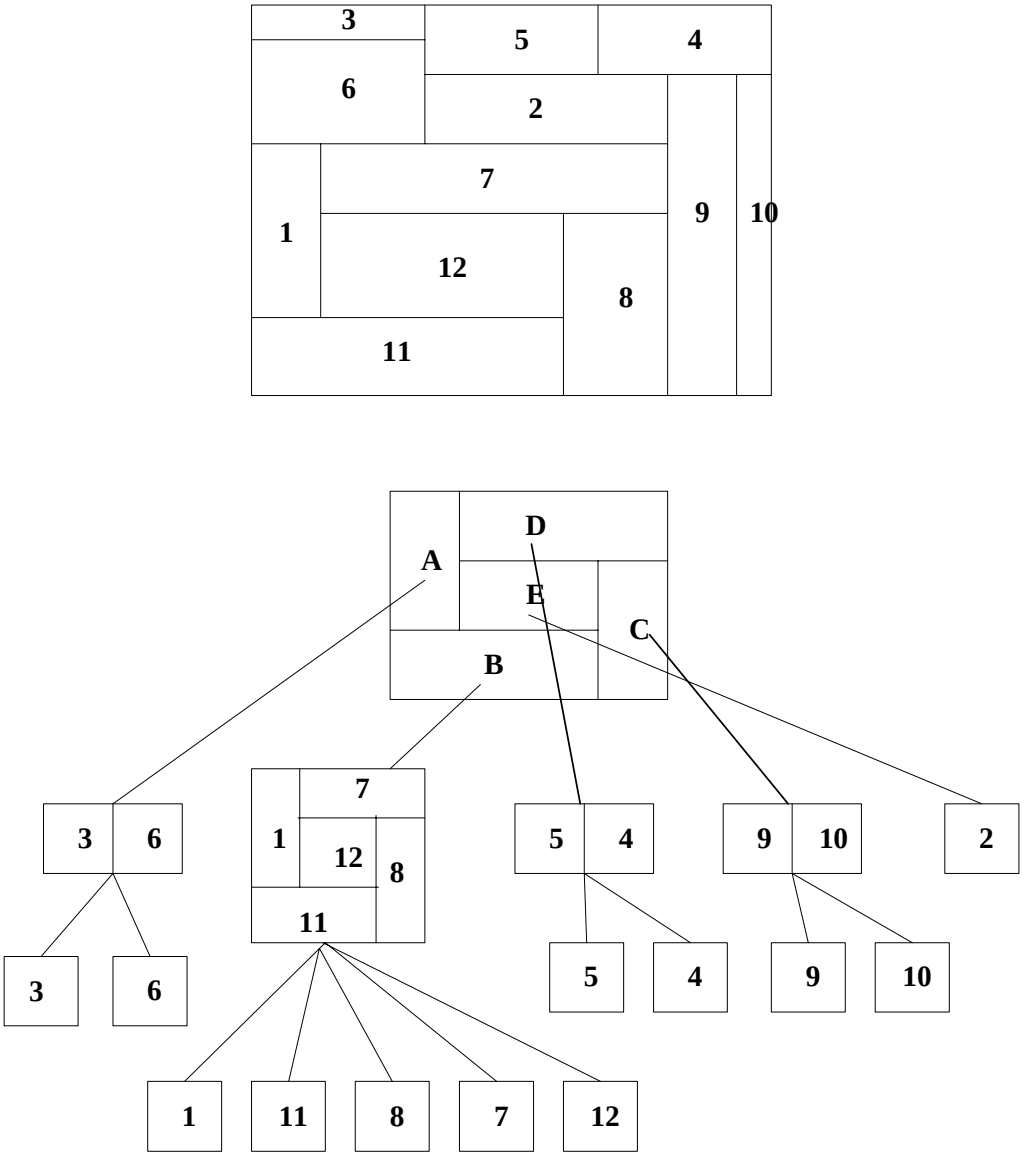
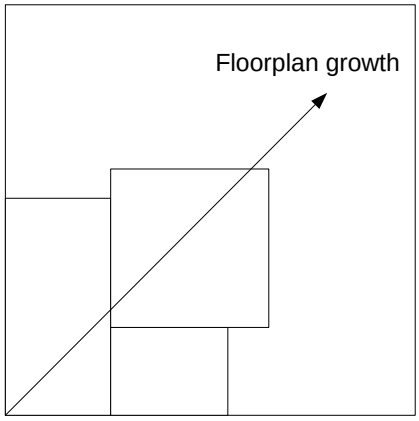
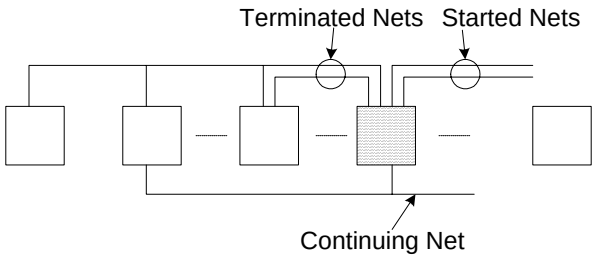


Fig. 3: Hierarchical Floorplan [2]

repeated loop. During each loop, the gain function of each unordered macro will be computed. The macro with the maximum gain will be selected, removed from the unordered list, and added to the sequence of ordered macros. If there exists a tie among several macros, the macro terminating the largest number of started nets is selected. If another tie occurs, the macro connecting to the largest number of continuing nets is selected. If there is one more tie, the most lightly connected macro is selected. Fig. 4(b) gives an example for net classification. In Fig. 4(c) the notation !*S* is used to represent the elements of sequence *S*. Square brackets are used to represent sequences and curly braces are employed with sets. The cluster growth algorithm is shown in Fig. 4(d).



(a) Cluster growth floorplanning



(b) Net classification during linear ordering

Algorithm Linear_Ordering
S: Set of all macros;
Order: Sequence of ordered macros; // initally empty
begin
Seed:=Select Seed macro;
Order:=[Seed];
S:=*S*-{Seed};
repeat
foreach macro *m* in S **do**
compute the gain for *m*;
gain_m:=number of nets terminated by *m* - number of new nets started by *m*;
end foreach;
Select the macro *m** with maximum gain;
if there is a tie **then**
Select the macro that terminates the largest number of nets;
else if there is a tie **then**
Select the macro that has the largest number of continuing nets;
else if there is a tie **then**
Select the macro with the least number of connections;
else break remaining ties as desired;
end if
Order:=[!Order,*m**]; // append m* to the ordered sequence
S:=*S*-{*m**};
until $S = \phi$;
end

(c) Linear ordering algorithm

Algorithm Cluster_Growth
S: Set of all macros;
begin
Order:=Linear_Ordering(*S*);
repeat
nextmacro:=*b* where Order=[*b*, !rest];
Order:=rest;
Select a location for *b* that will result in minimum increase in cost function;
until Order = ϕ ;
end

(d) Cluster growth algorithm

Fig. 4: Cluster growth approach [11]

II. Input/Output Specification

Input Format

Each testcase has two input files, *problem_no.mac* and *problem_no.net*. The first file defines chip and macro information. The former includes chip width and chip height, and the later includes name, area and aspect ratio constraints of a macro. The second file describes all net connectivities. For problem 1, there are two input files, *problem1.mac* and *problem1.net*. The first file format is as follows:

.chip_bbox (width, height)
// the lower-left corner of this bounding box is (0, 0)
.macro macro_name macro_area low_aspect high_aspect
.macro macro_name macro_area low_aspect high_aspect
... More macros
// low_aspect: lower bound of aspect ratio
// high_aspect: upper bound of aspect ratio
// for hard macro, low_aspect = high_aspect

The format of the second file (netlist) is :

.net net_name macro_name1 macro_name2 …
.net net_name macro_name1 macro_name2 …
… More nets
// one line defines a net
// for example, if net N1 connect macro A, B, and C, the definition is
// .net N1 A B C

Output Format

The output file *problem1.rpt* records problem output. This report consists of three parts: (1) bounding box for each macro (specified by lower-left corner and upper-right corner), (2) total wire length estimated by minimum spanning tree (Manhattan distance), and (3) area (it may be smaller than chip bounding box) . The area can be obtained by X * Y where X(Y) is the difference between rightmost(topmost) edge and leftmost(bottommost) edge among all macros. The report file format(*problem1.rpt*) is :

.macro macro_name (x1, y1) (x2, y2)
.macro macro_name (x1, y1) (x2, y2)
// (x1, y1): lower-left corner, (x2, y2): upper-right corner
… More macros
.mst total_wire_length
.area chip_area
// area = (max_x2 – min_x1) * (max_y2 – min_y1)

III. Problem Statement

Given 1) 15 different rectangle macros (5 hard macros and 10 soft macros), their core area and aspect ratio constraints, and (2) 20 nets among these macros, the tool arranges all these macros within a specified rectangular bounding box. We assume the lower-left corner of this bounding box is the origin (0,0) and no space (channel) is needed between two different macros. The main objective is to minimize the total wire length estimated by minimum spanning tree. The net terminals are assumed to be at the center of their corresponding macros. The second objective is to minimize the chip area.

We give an example for all IO files in Fig. 5.

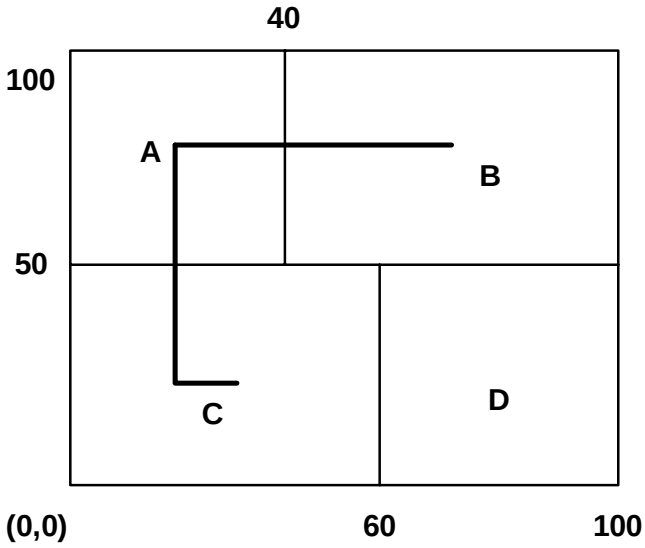


Fig. 5: A floorplan problem and its solution, bold line represent net N1.

Input files :

[*problem1.mac*]:

.chip_bbox (100,100)
.macro A 2000 0.6 1.5
.macro B 3000 0.8 1.2
.macro C 3000 0.8 1.5
.macro D 2000 0.8 0.8 // hard macro

[*problem1.spc*]:

.net N1 A B C

Output files :

[*problem1.rpt*]

.macro A (0, 50) (40, 100)
.macro B (40, 50) (100, 100)
.macro C (0, 0) (60, 50)
.macro D (60, 0) (100, 50)
.mst 110
.area 10000

IV. Advanced Features

Provide GUI (Graphic User Interface) to show the floorplanning result with interconnections (minimum spanning tree for each net).

V. Language/Platform

1.Language: C or C++.
2.Platform: SUN OS/Solaris or PC DOS/Windows.

VI. Evaluation

The score will be given based on the total wire length and minimal rectangular area needed for floorplanning, the time efficiency, and the memory requirement.
Bonus will be rewarded if the GUI is provided.

VII. Questions

Please report any question regarding this problem to cad@cs.nthu.edu.tw with the subject "CAD Contest: Problem 1." Your question(s) will be answered in two weeks, and the Q&A's will be posted at the contest web site

References

[1]Naveed Sherwani. *Algorithms For VLSI Physical Design Automation*. 3rd ed., Kluwer Academic Publishers, pages 222-223, 1999.

[2]Naveed Sherwani. *Algorithms For VLSI Physical Design Automation*. 3rd ed., Kluwer Academic Publishers, pages 193-196, 1999.

[3]G. Vijayan and R. Tsay. A new method for floorplanning using toplogical constraint reduction. *IEEE Transactions on Computer-Aided Design*, pages 1494-1501, December 1991.

[4]S. Sutanthavibul, E. Shragowitz, and J. Rosen. An analytical approach to floorplan design and optimization. *IEEE Transactions on Computer-Aided Design*, pages 761-769, June 1991.

[5]B. Lokanathan and E. Kinnen. Performance optimized floorplanning by graph planarization. *Proceedings of 26th ACM/IEEE Design Automation Conference*, pages 116-121, 1989.

[6]W. W. Dai, B. Eschermann, E. Kuh, and M. Pedram. Hierarchical placement and floorplanning in bear. *IEEE Transactions on Computer-Aided Design*. VOL. 8, pages 1335-1349, December 1989.

[7]M. Rebaudengo and M. S. Reorda. Gallo: a genetic algorithm for floorplan area optimization. *IEEE Transactions on Computer-Aided Design*. pages 943-951, 1996.

[8]S. M. Sait, H. Youssef, S. Tanvir, and M.S. T. Benten. Timing influenced general-cell genetic floorplanner. *Proceedings of the ASP-DAC'95/CHDL'95/VLSI'95., IFIP International Conference on Hardware Description Languages. IFIP International Conference on Very Large Scale Integration., Asian and South Pacific*, pages 135-140, 1995.

[9]H. Youssef, S. M. Sait, and K. J. Al-Farra. Timing influenced force directed floorplanning. *Proceedings EURO-DAC'95*., pages 156-161, 1995.

[10]S. M. Sait, and H. Youssef, VLSI Physical Design Automation – Theory and Practice. McGraw-Hill Book Company Europe and IEEE PRESS, pages 80-130, 1995

[11]S. M. Sait, and H. Youssef, VLSI Physical Design Automation – Theory and Practice. McGraw-Hill Book Company Europe and IEEE PRESS, pages 91-95, 1995