CAD for VLSI Design

Project Assignment 3

Analog Placement

Instructor: Andy, Yu-Guang Chen Ph.D.

TA: 曹寓恆

Department/Class: Electrical Engineering 4A

Name: 陳緯亭

Student ID Number: **109501201**

Contents

List Of Listings

1. How I compile and execute the program

```
(PA3) [s109501201@cad ~/PA3]$ make all
g++ -std=c++11 -c placement.cpp -o obj/placement.o
g++ -std=c++11 -c constraint.cpp -o obj/constraint.o
g++ -std=c++11 -c contour.cpp -o obj/contour.o
g++ -std=c++11 -c graph.cpp -o obj/graph.o
g++ -std=c++11 -c main.cpp -o obj/main.o
g++ -std=c++11 -c btree.cpp -o obj/btree.o
g++ -std=c++11 obj/placement.o obj/constraint.o obj/contour.o obj/graph.o obj/main.o obj/btree.o -o 109
501201
```

Fig 1: Compile my source code and generate the corresponding objects

```
(PA3) [s109501201@cad ~/PA3]$ make d target=case2 ratio=3
./109501201 3 public/case2/input/case2.netlist public/case2/input/case2.sym public/case2/intermediate/c
ase2.block ./public/case2/output/case2.output
V VDD
V VSS
Nodes connected to VSS found by BFS:
M63 M109 M98 M119 M26 M90 M111 M14 M116 M117 M118 M60 M113 M103 M6 M115 M114 M110 M3 M24 BB0 M112 M107
M106 M108 M104 M16 M10 M9 M15
--------------------------------------------------------------------------------

GdsGenerator ./public/case2/input/case2.netlist  ./public/case2/input/Process.tf  ./public/case2/output
/case2.output
scp s109501201@140.115.71.192:/home/CAD112/s109501201/PA3/./public/case2/output/case2.output.gds file.o
utput.gds
s109501201@140.115.71.192's password:
case2.output.gds                                              100%   88KB  87.5KB/s   00:00
```

Fig 2: Execute my analog placer and then download GDSII File for visualization purposes

```
(PA3) [s109501201@cad ~/PA3]$ make verifier target=case2 ratio=3

*****************************************************************************************************
* Check whether the calculation results of the device info. match your own calculation results or not *
*****************************************************************************************************
*                          From Device Info.                  From Your Output                      *
*    1. Total HPWL         2134.37                            2134.37                               *
*    2. Chip Area          20361.596                          20361.596                             *
*    3. Chip Width         136.6                              136.6                                 *
*    4. Chip Height        149.06                             149.06                                *
*****************************************************************************************************

Correctness:               PASS
Meet Symmetry Constraints: PASS
Cost                       9399.957535
```

Fig 3: Run verifier

2. Pseudo Code

Algorithm 1 Build a Tree

function buildATree(root, left, recurse, max_x, max_y, map, path)
  if $root$ = nullptr **then**
    return
  end if
  $w \leftarrow$ root width, $h \leftarrow$ root height
  $axis \leftarrow root \rightarrow axis$
  if $root \rightarrow parent$ = nullptr **then** ▷ Root node
    $root \leftarrow$ Initialize
  else
    $pw \leftarrow$ parent width, $ph \leftarrow$parent height
    if $root$ is self-symmetry module **then**
      $root\ x \leftarrow parent\ center\_x$
    else
      $root\ x \leftarrow parent\ max\_x$
    end if
    if $root$ is in the symmetry file **then**
      if $root$ is self-symmetry module **then**
        $root\ x \leftarrow root\ x - w/2$
      else
        $root\ x \leftarrow root\ x - w$
      end if
    else if $left$ **then**
      $root\ x \leftarrow root\ x + parent\ axis + root\ axis$
      if $root$ is self-symmetry module **then**
        $root\ x \leftarrow root\ x - w/2$
      else
        $root\ x \leftarrow root\ x - w$
      end if
    else
      $root\ x \leftarrow root\ x - parent\ axis + root\ axis$
      if $root$ is self-symmetry module **then**
        $root\ x \leftarrow root\ x - w/2$
      else
        $root\ x \leftarrow root\ x - w$
      end if
    end if
    $root\ center\_x \leftarrow root\ x + w/2$

```
        root max_x ← root x + w
        if root is self-symmetry module then
            y ← from checking contour
        else
            y ← from checking contour
        end if
        root y ← y
        root max_y ← y + h
        root center_y ← y + h/2
    end if
    if root max_x > max_x then
        max_x ← root max_x
    end if
    if root max_y > max_y then
        max_y ← root max_y
    end if
    refresh contour
    if root ≠ nullptr then
        if root is self-symmetry module then
            create another MOS in symmetry pair location
        end if
    end if
    if recurse then recurse
    end if
end function
```

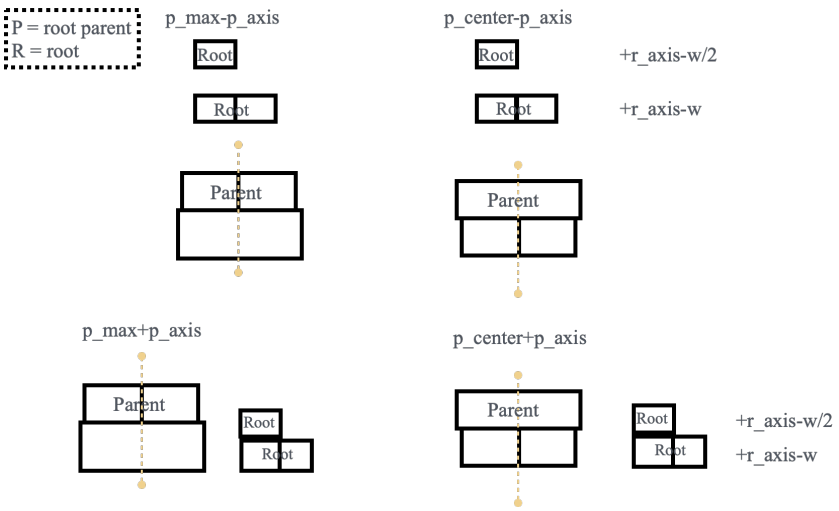

Fig 4: Find block ocation

2.1 The degree of completion of the assignment: Perturbation Not Yet Complete

3. Algorithms and data structures

There are four structures in my programme, InBlock, InSym, TreeNode and Contour. InBlock is for collecting input data from the .block file. InSym is for storing the information in a symmetry constraint file. TreeNode is for constructing the b* tree. And Contour is to record the y-contour to prevent block from overlapping.

Listing 1: The data structure

```
// structure.h
#ifndef STRUCT_H
#define STRUCT_H
#include <string>
#include <vector>

using namespace std;
//---------------------------------------------------------------------
struct InBlock // intermediate files (.block)
{
    string name;                // the first BB or MM that appears
    vector<string> contents;    // if it is BB that has MM
    vector<vector<double>> wh; // width, height, col_multiple, row_multiple
    int size_pick;              // pick the size of blocks in block.wh
    int num;                    // number types of block
};
//---------------------------------------------------------------------
struct InSym // symmetry constraint files (.sym)
{
    int index;          // symmetry key
    vector<string> sym; // symmetry block (at most 2)
};
//---------------------------------------------------------------------
// define b*-tree structure
struct TreeNode
{
    InBlock block;
    string sym_name;          // the block name which is symmetry to the first BB or
        MM
    double axis;              // symmetry axis
    vector<double> xy;        // xy location of the block
    vector<double> center_xy; // center xy value of the block
    vector<double> max_xy;    // max xy value of the block
    int isSym;                // that is the symmetry block
    TreeNode *left;
```

```
35      TreeNode *right;
36      TreeNode *parent;
37      TreeNode(InBlock n, string sym) : block(n), sym_name(sym), axis(0), left(nullptr),
            right(nullptr), parent(nullptr), xy(2), center_xy(2), max_xy(2), isSym(0) {}
38  };
39  //---------------------------------------------------------------------------
40  struct Contour
41  {
42      double x1, x2, y; // (x1, x2)
43      Contour *next;
44
45      Contour() : x1(0), x2(0), y(0), next(nullptr) {} // Proper default constructor
46  };
47
48  #endif // STRUCT_H
```

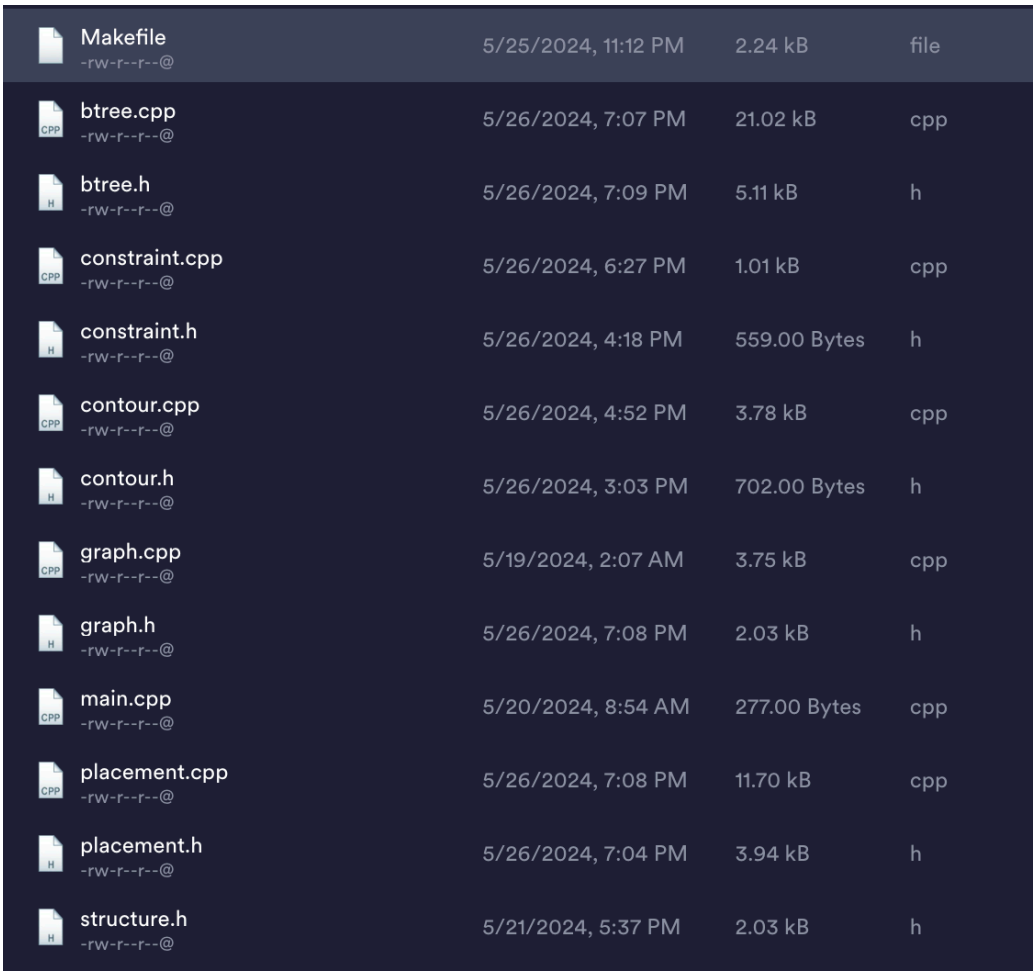

Fig 5: Source code

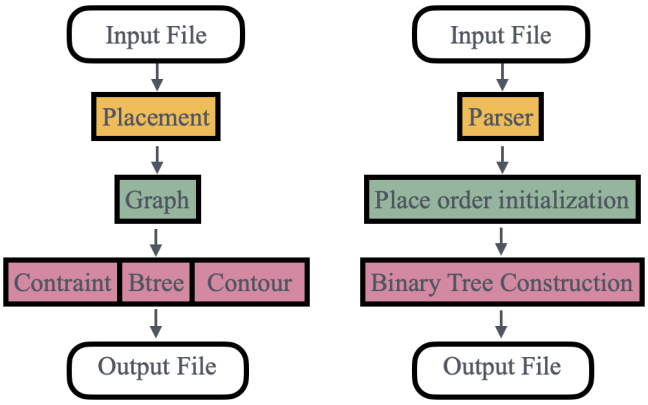

Fig 6: Source File and its functions

The program is constructed using six .cpp files and six .h files. The placement.cpp file serves as a parser for both the input and output files. The graph.cpp file is responsible for arranging the blocks that are placed in the first instance. The btree.cpp file is used to construct a b*-tree and employs a pre-order traversal to traverse all the data. The constraint.cpp file ensures that the aspect ratio limitations are met, thus preventing the blocks from becoming excessively long. Finally, the contour.cpp file is responsible for processing the y location in order to place the blocks.

4. Perturbation strategy and operations

Using Simulated Annealing to pick the new TreeNode according to cost function in constraint.cpp. This is not completed. Therefore, the refresh TreeNode answers are wrong.

Listing 2: The Simulated Annealing

```
49 void BTree::SA(vector<string> &order1)
50 {
51     int times = 0;
52     std::vector<std::string> now_order = order1; // order for BB and MOS
53     double now_HPWL;
54     TreeNode *now_trees = nullptr;
55     double t = 6000; // temperature
56     double cost;
57     map<std::string, TreeNode *> now_map;
58     Contour *path = nullptr;
59
60     // Initialization start
61     time(&start);
62
```

```
63      while (!this->constraint.TimeLimited(times, start, end))
64      {
65          double now_x = 0, now_y = 0;
66          swapOrder(now_order); // swap for two string in vector
67          TreeSort(this->trees, now_order);
68          path = nullptr;
69
70          std::cout << "Swapped Order: ";
71          for (const auto &o : now_order)
72          {
73              std::cout << o << " ";
74          }
75          std::cout << std::endl;
76
77          std::cout << "Before buildATree" << std::endl;
78          now_trees = copyTree(trees[0]); // copyTree
79          buildATree(now_trees, 0, 1, now_x, now_y, now_map, path);
80          std::cout << "After buildATree" << std::endl;
81
82          if (trees.size() > 1)
83          {
84              for (size_t i = 1; i < trees.size(); ++i)
85              {
86                  std::cout << "Before connectTrees, i: " << i << std::endl;
87                  connectTrees(now_trees, trees[i], 0, now_x, now_y, now_map, path);
88                  std::cout << "After connectTrees, i: " << i << std::endl;
89              }
90          }
91
92          calculateHPWL(now_HPWL, now_map);
93
94          cost = this->constraint.Cost(now_x, now_y, now_HPWL, ratio) - this->constraint
                .Cost(ans_x, ans_y, ans_HPWL, ratio);
95          if (this->constraint.pick(cost, t))
96          {
97              std::cout << "change" << std::endl;
98              ans_map = copyTreeMap(now_map);
99              ans_x = now_x;
100             ans_y = now_y;
101             ans_HPWL = now_HPWL;
102             a_trees = copyTree(now_trees);
103         }
104         t = t * 0.99;
105         times++;
106     }
107 }
```

5. Cost function and how to determine to next state

The cost function should come in the area, minimize half-perimeter wire length, my aspect ratio and the expected ratio from input. It should be used in the Simulated Annealing.

Listing 3: Cost function

```
108 double Constraint::Cost(double w, double h, double HPWL, const double &ratio) // cost
        function
109 {
110     double cost;
111     cost = (0.25 * w * h + 0.75 * HPWL) * (1 + (AspectRatio(w, h) - ratio / 3) * (
            AspectRatio(w, h) - ratio / 3));
112     return cost;
113 }
```

The aspect ratio is $max\left(\frac{w}{h}, \frac{h}{w}\right)$.

Listing 4: Aspect ratio

```
114 double Constraint::AspectRatio(double w, double h) // aspect ratio
115 {
116     double r1 = w / h;
117     double r2 = h / w;
118     double ans;
119     if (r1 > r2)
120         ans = r1;
121     else
122         ans = r2;
123     return ans;
124 }
```

6. The hardness of this assignment / I overcome it

1. Contour can't not be refresh in the correct y.
   Ans: The contour value (x_1, x_2, y_1) and the check scale (x_3, x_4, y_2) are set, and if $x_3 = x_2$, then y_2 would be y_1. I didn't expect that. So I give it a little bias ($x_1 = x_1 + 0.0001$) to solve this problem.

2. How to find the axis in self-symmetry module and symmetry pair?
   Ans: I find that it can be solved by using recursion.

3. How to place the block and mos?
   It is necessary to consider whether the parent is a symmetry pair or self-symmetric, as well as whether the object itself is a symmetry pair or self-symmetric.

7. Suggestions

No.