CS6135 VLSI Physical Design Automation
Homework 5: Automated P&R for Analog Circuits

111062625 蔡哲平

1. **How to compile and execute my program.**
   - Compile: Enter ***src/*** and make, it'll generate the executable file to ***bin/***.

     ```
     $ cd src
     $ make
     ```
   - Execute

     ```
     $ ./hw5 [NUM_CURRENT_SOURCES] [DEF_FILE_PATH]
     ```
     e.g.

     ```
     $ ./hw5 16 ../DEF/CS_16.def
     ```

2. **The details of my implementation:**
   The following is a detailed explanation of each step.

   Step 1: Create a Die Boundary

   Let NCS denotes the number of current sources. We can conclude that the number of cells on each side of the layout is sqrt(NCS * 4), denoted as Num_Cell; the number of vertical ME3 metals in each column is Num_Cell / 2, denoted as Num_M3; the number of ME4 ports in each row is Num_CELL / 4, denoted as Num_M4. The boundary of the die is represented by its left bottom coordinate, denoted as (die_x1, die_y1), and its right top coordinate, denoted as (die_x2, die_y2). The generalization formula of each coordinate is given below:

   - die_x1 = 0
   - die_y1 = 0
   - die_x2 = CS_Width * Num_Cell + M3_Sacing * ((Num_M3 + 1) * Num_Cell-1) + M3_Width * Num_M3 * Num_Cell
   - die_y2 = CS_Height * Num_Cell + M4_Spacing * ((Num_M4 + 1) * Num_Cell - 1) + M4_Width * Num_M4 * Num_Cell

   Step 2: Create CS Placement

   In this step, we create a 2D array to store cells and several parameters will be introduced. Off_y denotes the starting y coordinate of the first cell, formulated by NUM_M4 * (M4_Spacing + M4_Width); D_y denotes the y-direction padding of each cell, formulated by Off_y + CS_Height + M4_Spacing; Similarly, D_x denotes the x-padding of each cell, formulated by CS_Width + Num_M3 * (M3_Width + M3_Spacing) + M3_Spacing. The placement of the cell is represented by its left bottom coordinate (x, y). The generalization formula of CS[i][j] is given below:

- 2D array: Component CS[Num_Cell][Num_Cell]
- x = i * D_x
- y = j * D_y + Off_y

Step 3: Create Vertical ME3

In this step, we create a 2D array to store ME3s and several parameters are introduced. D_x denotes the x-distance between the column cell and the first ME3 metal, formulated by CS_Width + M3_Spacing. P_x denotes the padding between each ME3 metal in the same column, formulated by M3_Width + M3_Spacing. The placement of the ME3 metal is represented by its left bottom coordinate (x1, y1) and the right top coordinate (x2, y2). The generalization formula of ME3[i][j] is given in below:

- 2D array: SpecialNet ME3[Num_Cell][Num_M3]
- x1 = CS[i][0].x + D_x + j * P_x
- x2 = x1 + M3_Width
- y1 = 0
- y2 = die_y2

Step 4: Create ME4 Drain Connection

In this step, we create a 2D array to store each ME4 drain and several parameters are given. CSX1toD denotes the x-distance from the cell's left bottom x coordinate to the drain's left bottom x coordinate. CSY1toD is similar to CSX1toD. In this step, we generate the connections for the four units of a device in a single for-loop iteration. The formula of the coordinate of the left bottom portion M4_drain[i][j] is given in below, and other portions can be derived from the left bottom portion using the mirror technique.

- 2D array: SpecialNet ME4_drain[Num_Cell][Num_Cell]
- x1 = CS[i][j].x1 + CSX1toD
- y1 = CS[i][j].y1 + CSY1toD
- x2 = ME3[i][j].x2
- y2 = y1 + M4_width

Step 5: Create ME4 Port

In this step, we create a 2D array to store each ME4 port and several parameters are introduced. D_y denotes the starting y-coordinate of the first port in each row, formulated by CS_Height + Num_M4 * P_y + M4_Spacing. P_y denotes the padding between each ME4 port, denoted as M4_Width + M4_Spacing. The formula of ME4_Port[i][j]'s coordinate is given below:

- 2D array: SpecialNet M4_Port[Num_Cell][Num_M4]
- x1 = 0
- x2 = die_x2
- y1 = i * D_y + j * P_y
- y2 = y1 + M4_Width

Step 6: Create Via34 from ME4 Drain

In this step, we create a 2D array to store each via. The method is very similar to Step 4. Each via's coordinate in the left bottom portion can be formulated as below, and other portions can be derived from the left bottom portion using the mirror technique.

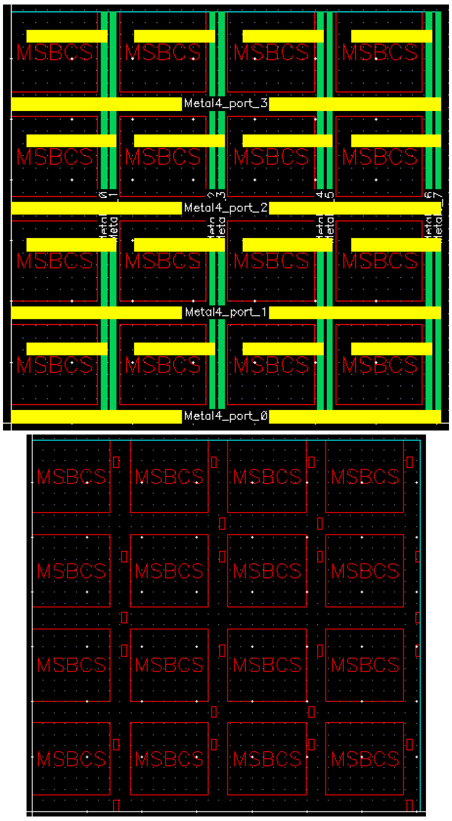
- 2D array: Component Via34_Drain[Num_Cell][Num_Cell]
- x1 = ME3[i][j].x1
- y1 = CS[i][j].y1 + CSY1toD

Step 7: Create Via34 to ME4 Port

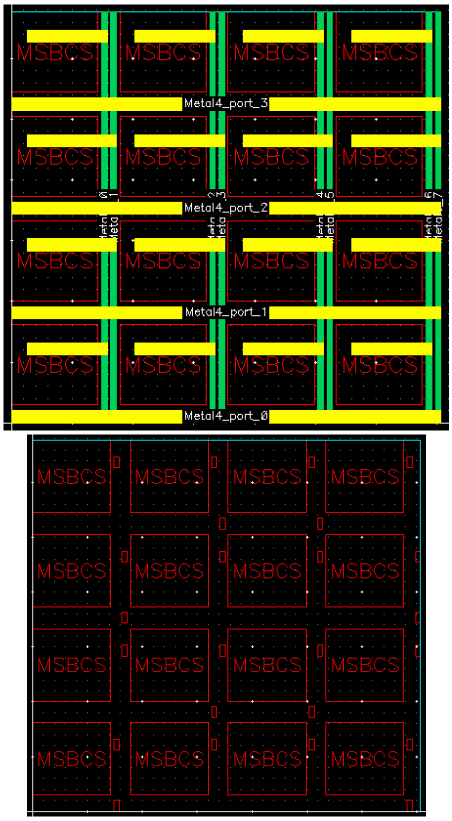
Since the paper didn't consider fixed cells, which are blockages in this assignment. In this step, we create a 3D array to store each via. For each ME4 port, there will be 2 vias. One on the left and the other on the right. Both vias share the same y-coordinate, which is the y1 of ME4_Port[2 * j + i / Num_M4][i % Num_M4]. As for the x coordinate of the left via, it's the x1 of ME3[i][j]. Similarly, the x coordinate of the right via is the x1 of ME3[Num_Cell – 1 - i][j]. For a more detailed explanation, you can refer to the code snippet below:

```
/* Step 7: create Via34 to ME4 port */
Component Via34_port2ME3[NUM_CELL][NUM_M4][2];
for (int i = 0; i < HALF_NUM_CELL; i++) {
    for (int j = 0; j < HALF_NUM_CELL; j++) {
        std::string inst_name;
        int x, y;
        // left via
        inst_name = "Via34_port2ME3_" + std::to_string(i * HALF_NUM_CELL + j + 0 * n);
        x = ME3_specialnet[i][j].x1;
        y = ME4_specialnet_port[2 * j + i / NUM_M4][i % NUM_M4].y1;
        Via34_port2ME3[2 * j + i / NUM_M4][i % NUM_M4][0] = Component(VIA34_LIB_NAME, inst_name, x, y);
        // right via
        inst_name = "Via34_port2ME3_" + std::to_string(i * HALF_NUM_CELL + j + 1 * n);
        x = ME3_specialnet[NUM_CELL - 1 - i][j].x1;
        Via34_port2ME3[2 * j + i / NUM_M4][i % NUM_M4][1] = Component(VIA34_LIB_NAME, inst_name, x, y);
    }
}
```

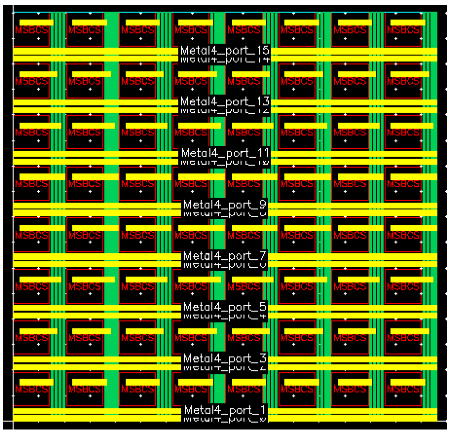
3. The screenshots of your placement and routing results.



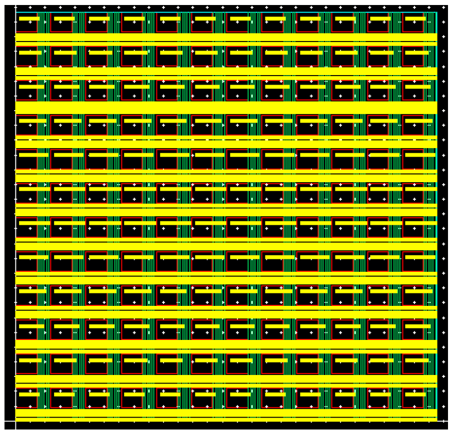
Python version: CS=4



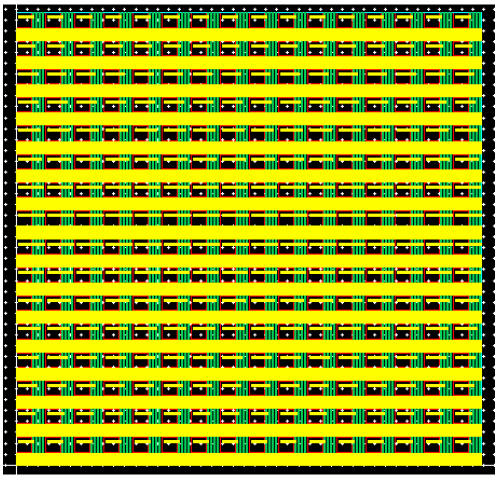
C++ version: CS=4



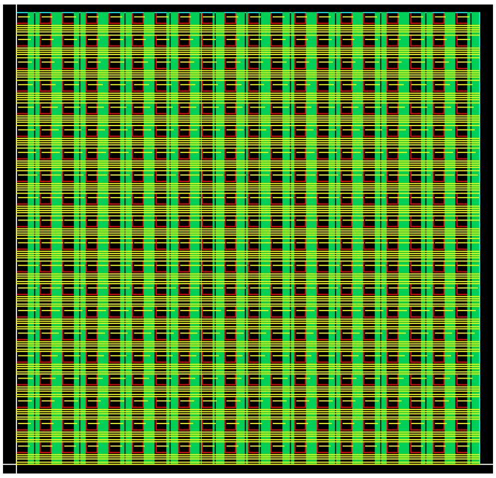
C++ version: CS=16



C++ version: CS=36



C++ version: CS=64



C++ version: CS=100