CS6135_HW5_110062619_report

(1)Your name and student ID

Name: 楊淨富

Student ID: 110062619

(2)How to compile and execute your program, and give an execution example.

- How to Compile

In /HW5/src directory, enter the following command:

> 💡 $ make

> 💡 It will generate the executable files "hw5" in "HW5/bin/".

- How to execute

In /HW5/src directory, enter the following command:

> 💡 Usage: ../bin/<exe> <number of current sources> <def file path>

```
e.g.:
 $ ../bin/hw5 4 ../output/CS_4.def
```

- How to verify

In /HW5/src directory, enter the following command:

💡 Usage: $ ../verifier/verify <k-value: 4, 16...> <.def file>

```
e.g.:
  $ ../verifier/verify 4 ../output/CS_4.def
```

(3)The details of your C/C++ program. How do you generalize the original C/C++ program to handle 16 or more current sources? You have to describe what you do step by step in detail.

- 程式的flow:

先定義int tmp = sqrt(number of current source)

1. 建Die的邊界
    a. Generalize x coordinate
        - CS_Width(current source的寬)* CS在x方向上的數量(即tmp*2)
        - M3_Width * M3在x方向上的數量(即tmp* tmp * 2): 任兩個CS之間有tmp個M3，x方向上總共會有tmp*2個CS，所以是tmp*tmp*2 = numCS * 2
        - M3_Spacing * M3_Spacing的數量:先把任兩個current sources之間的M3的數量算出來，就也同時算出M3的Spacing數量了(即M3的數量+1)，然後最右邊的部分要少加一次Spacing。
    b. Generalize y coordinate
        - CS_Height * CS在y方向上的數量(即tmp*2)
        - M4_Width * M4 port在y方向上的數量(即tmp/2 * tmp*2): 任兩個CS之間有tmp/2的port，y方向上總共算有tmp*2個CS，所以是tmp/2 * tmp*2
        - M4_Spacing * M4_Spacing的數量: 先算出任兩個CS之間有幾個M4 port，Spacing數量即M4 port數量+1，但最下方的部分要少加一次Spacing。

2. 建每個Current source(CS)的座標
    - 先確定總共有多少個CS，即有tmp*2個row，和tmp*2個col，故兩層for-loop，且外圈index從0~tmp*2-1，內圈index從0~tmp*2-1。
    - 如同講義算出Dx , Dy , off_y
    - 要特別注意off_y並非如同講義上只是M4_Spacing+M4_Width，還必須 * tmp，理由是一般化的時候，任兩個CS之間並不是只有一個M4 port。
    - 接下來套講義公式即可:
        - x = i * Dx
        - y = j * Dy + off_y * tmp
3. 建垂直的ME3
    - 先確定總共有幾個ME3，才能定義存ME3的資料結構和index的跑法。任兩個CS之間會有tmp個ME3(內圈index從0~tmp-1)，總共會有tmp*2個CS(外圈index從0~tmp*2-1)。
    - 接下來欲求x1, x2, y1, y2都可以直接套用講義公式
    - 值得注意的是Step 3的Dx和Step 2的Dx並不相同，我認為講義這樣取名會造成混淆，所以此處我有修改變數名稱。
4. 建ME4 Drain
    - 雖然Drain的數量是(tmp*2) * (tmp*2)，但由於我們只擺左下角，之後mirroring to x-axis and y-axis，所以外圈和內圈的for-loop的index都從0~tmp-1即可。
    - Mirroring to x-axis時，x1&y1外圈的index應該要改為整條的CS數量扣掉當前的外圈index。而整條row的CS數量是tmp*2，以CS=36的第一個CS舉例:
        - 左下角x1 = input->cs_array[0][j]->x + GP->CS_X1_TO_DRAIN
        - 右下角的x1 = input->cs_array[(tmp*2-1)-i = 11-0 = 11][j]->x + GP->CS_X1_TO_DRAIN
    - Mirroring to y-axis(即左上方時)，如同mirroring to x-axis，只是這次是內圈的index要作相應的調整，以上面例子為例，這次改內圈的index j，改成[i][(tmp*2-1)-j]即可。
5. 建ME4 Port

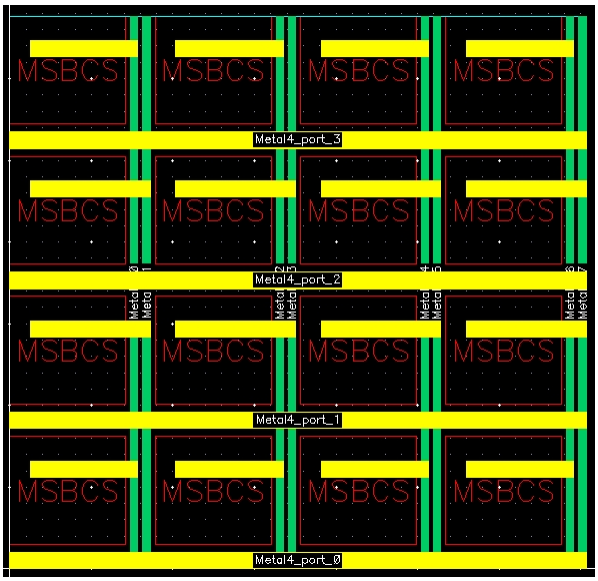
- 先確定Port的數量，這部分我一開始沒注意到CS = 4時，由於任兩個CS之間只會有一條Port，我一開始是用vector<Component*>去存，但後來發現CS =16時，任兩個CS就會有兩條以上的Port，所以就必須把原先的一維vector改為二維的vector<vector<Component*>>。
- 內圈index為任兩個CS之間會有的Port數量，即tmp。外圈index為總共有多少CS，即tmp*2。
- 與講義有所不同的地方是y1，因為不再只有一條Port，所以必須算出它在任兩個CS之間是第幾條Port(內圈index j = 0 ~ tmp/2-1條)。

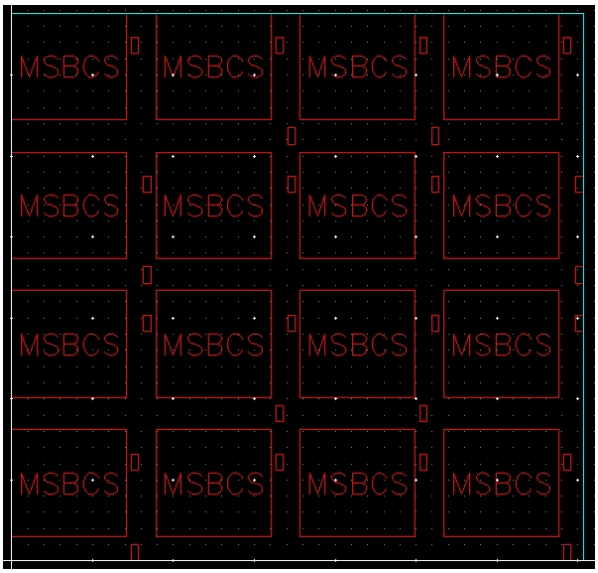
6. 建Via34 from ME4 Drain to ME3
    - 類似於Step 4，一樣只擺左下角，其他方位用mirroring的方式去計算。
    - 即如果是對稱於x軸，則右邊的x座標必須改為整個row上全部CS的數量減去當前的index i，才能找到對應的CS，而右邊的y座標也是同理，只要找到正確index所鏡射對應的CS，就可以算出座標。
    - y軸同理。
7. 建Via34 from ME3 to ME4 Port
    - 類似於Step 4 & Step 6，但差別在於，只利用x-axis來鏡射，即分為左右兩半，每次iterate ME4 Port來進行擺放。
    - 擺法為第一列和第二列的CS上的Port接到對應ME3的第一條，第三列和第四列的CS上的Port接到對應ME3的第二條，以此類推。只要知道如何接，就能透過幾個外圈和內圈的index去求出(i, j)和對應(x, y)所需要的index之間的關係。

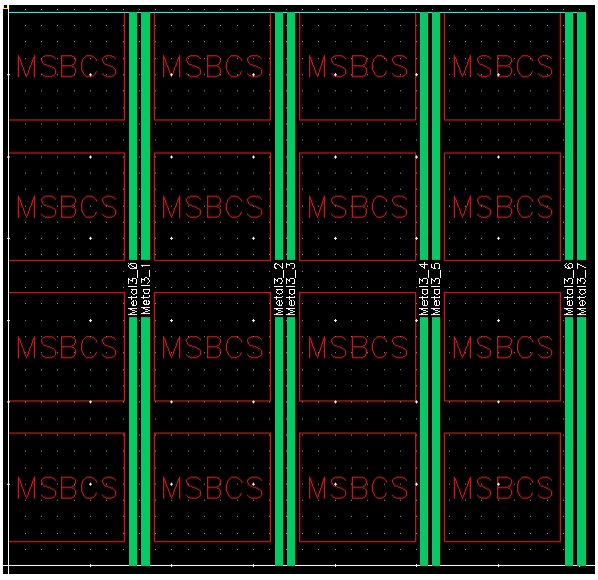
(4)The screenshots of your placement and routing results for the circuit produced by your Python program for the case of 4 current sources as well as by your C/C++ program for the cases of 4, 16, 36, 64, and 100 current sources

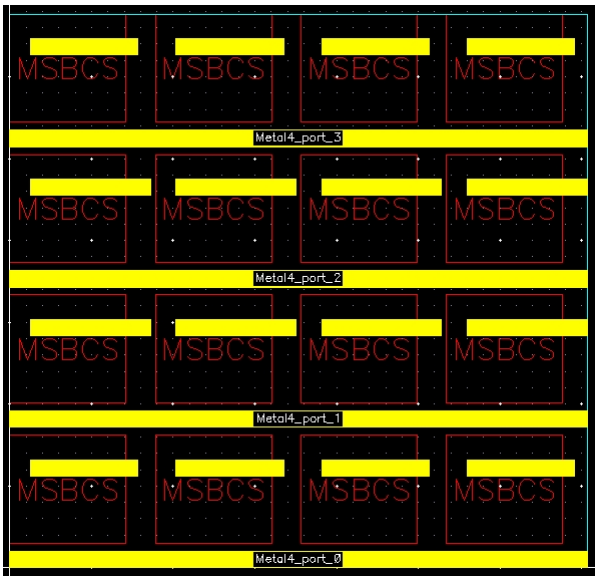
```
[g110062619@ic51 ~/HW5_grading]$ ./HW5_grading.sh
|-------------------------------------------------|
|                                                 |
|    This script is used for PDA HW5 grading.     |
|                                                 |
|-------------------------------------------------|
grading on 110062619:
  testcase |     result | status
    python |       pass | success
         4 |       pass | success
        16 |       pass | success
        36 |       pass | success
        64 |       pass | success
       100 |       pass | success
|-------------------------------------------------------|
|                                                       |
|    Successfully generate grades to HW5_grade.csv      |
|                                                       |
|-------------------------------------------------------|
```

- CS = 4 (Python)

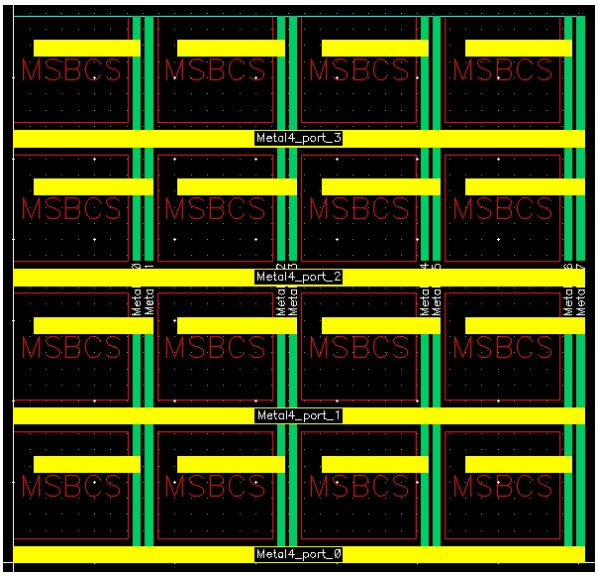


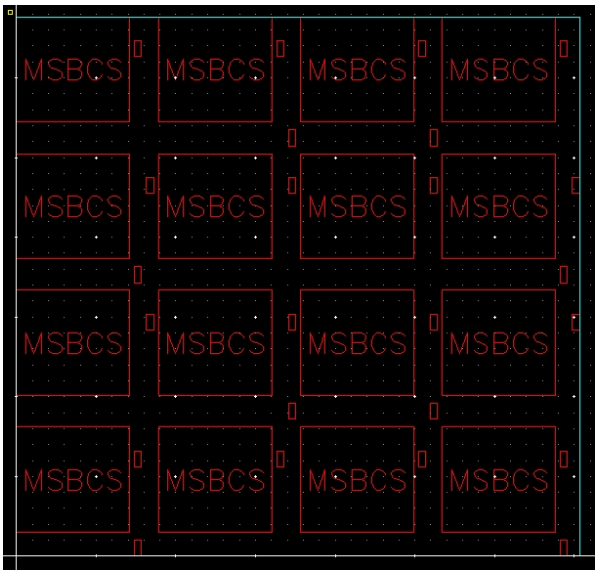


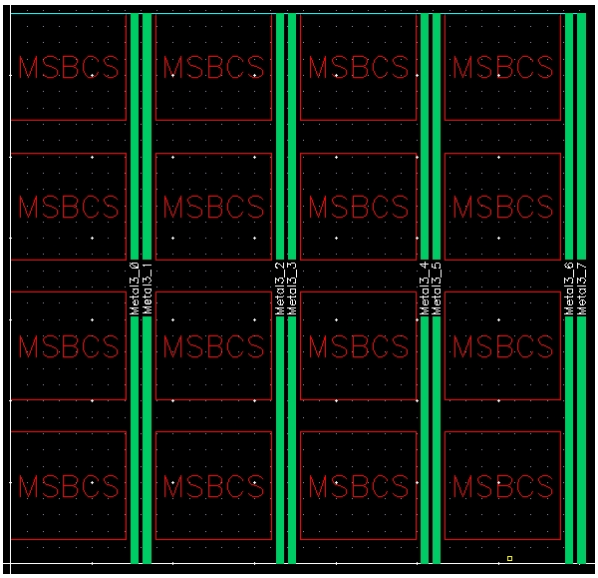

MSBCS
Metal3_0
Metal3_1
Metal3_2
Metal3_3
Metal3_4
Metal3_5
Metal3_6
Metal3_7

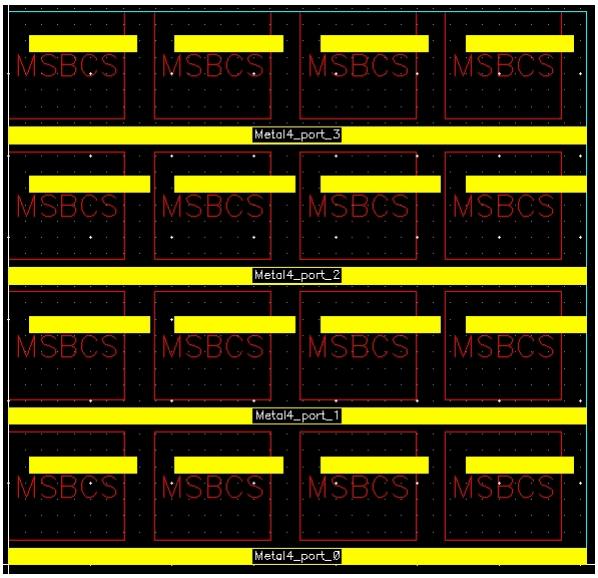

MSBCS
Metal4_port_3
Metal4_port_2
Metal4_port_1
Metal4_port_0

- CS = 4 (C++)

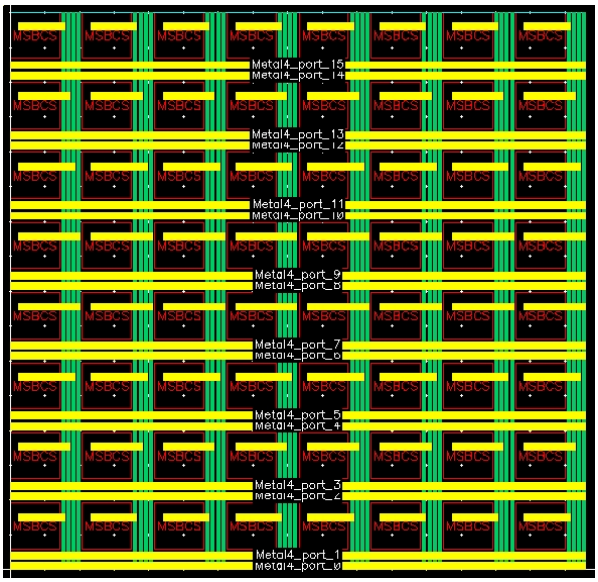

MSBCS
Metal4_port_3
Metal4_port_2
Metal4_port_1
Metal4_port_0

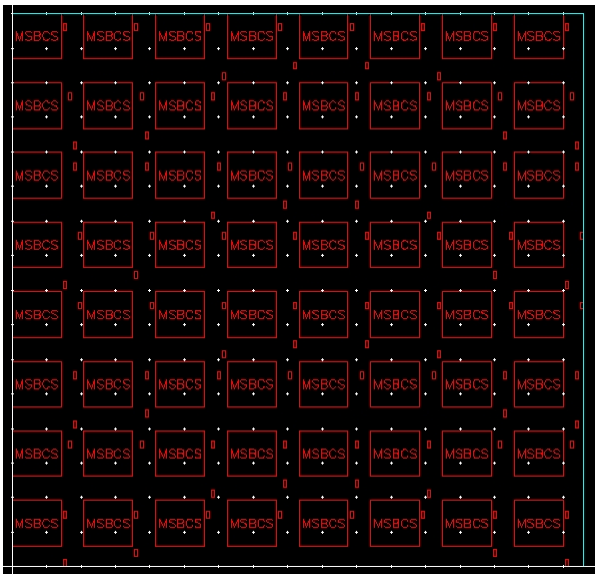

MSBCS

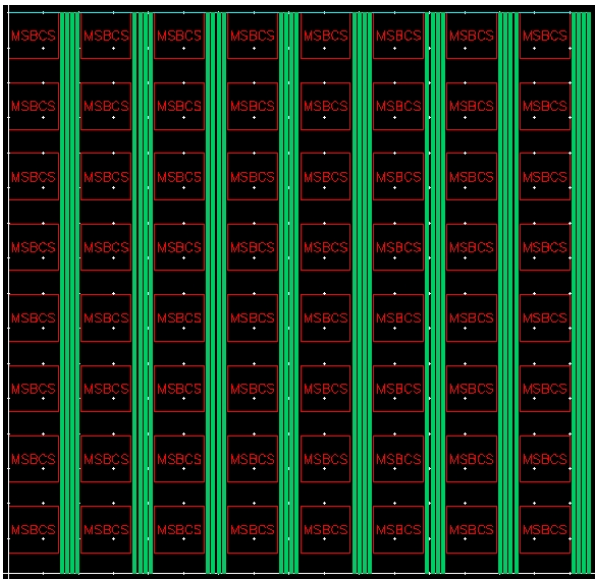

MSBCS
Metal3_0
Metal3_1
Metal3_2
Metal3_3
Metal3_4
Metal3_5
Metal3_6
Metal3_7

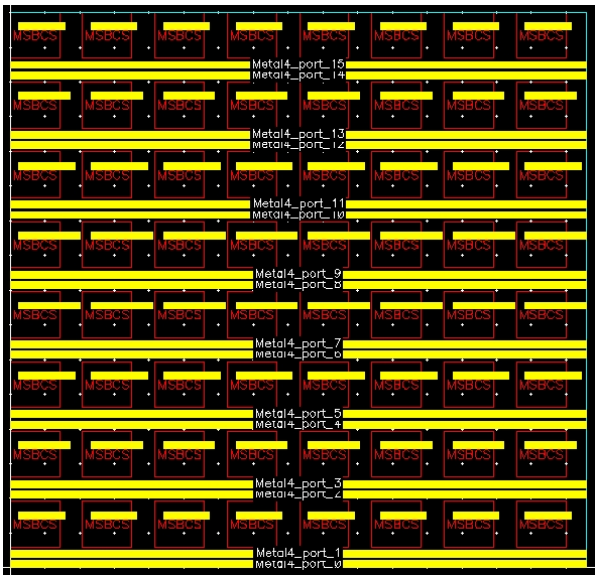

MSBCS
Metal4_port_3
Metal4_port_2
Metal4_port_1
Metal4_port_0

- CS = 16

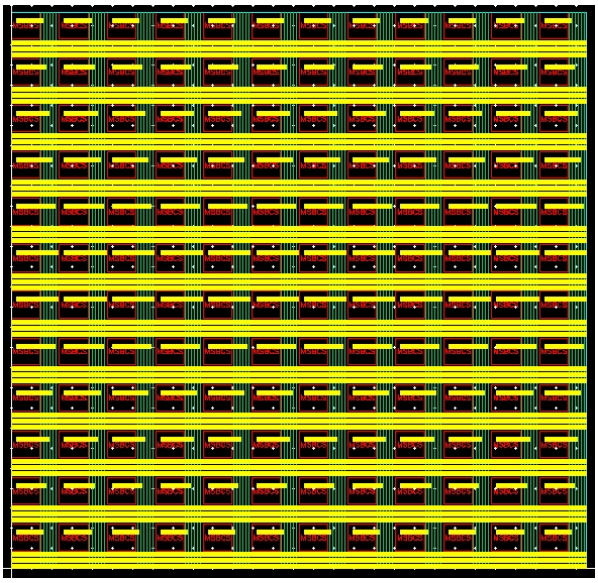

MSBCS
Metal4_port_15
Metal4_port_14
Metal4_port_13
Metal4_port_12
Metal4_port_11
Metal4_port_10
Metal4_port_9
Metal4_port_8
Metal4_port_7
Metal4_port_6
Metal4_port_5
Metal4_port_4
Metal4_port_3
Metal4_port_2
Metal4_port_1
Metal4_port_0

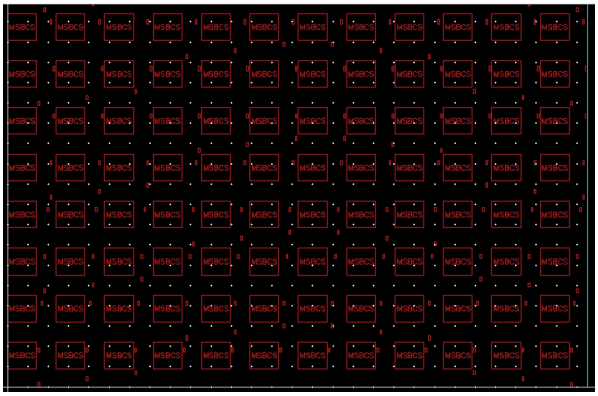

MSBCS



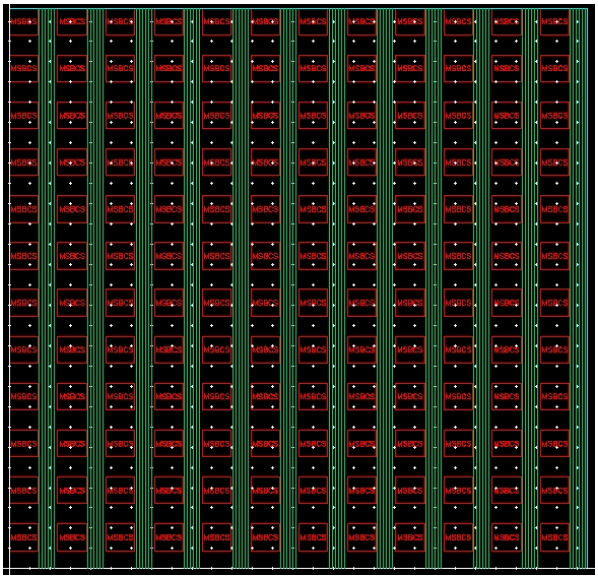


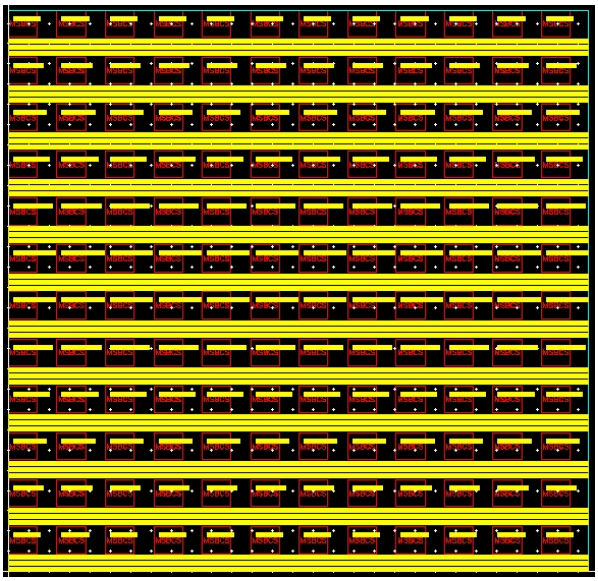
- CS = 36



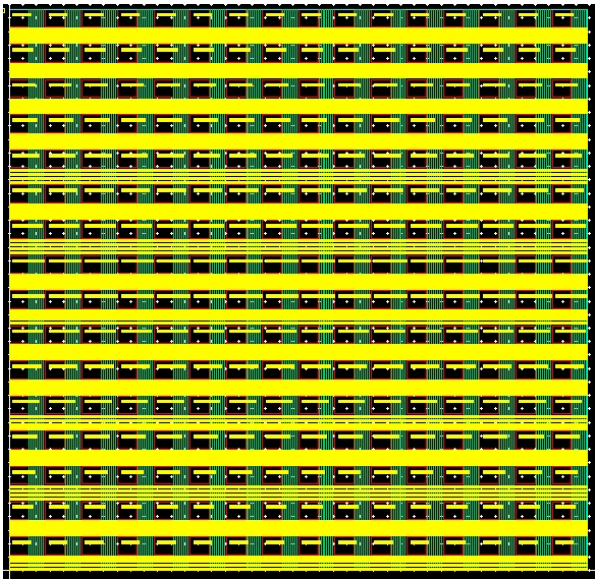


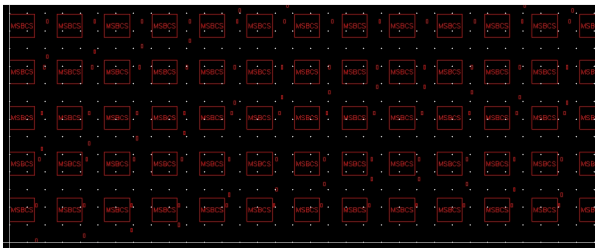
P.S.: 上圖的Via34(CS=36)只展示一部分而已



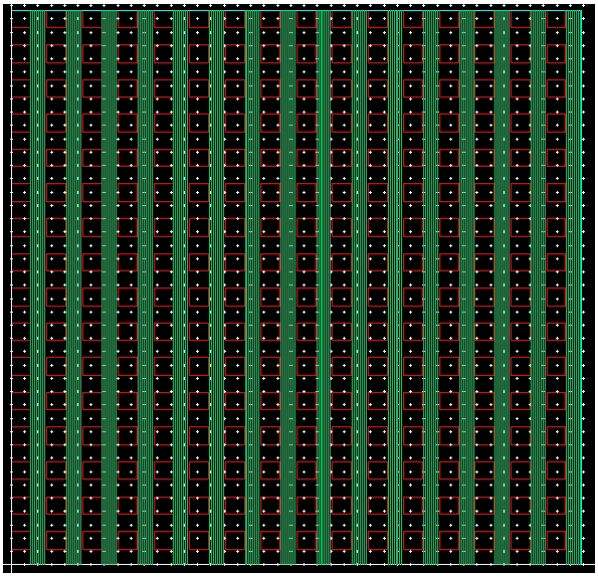


- CS = 64

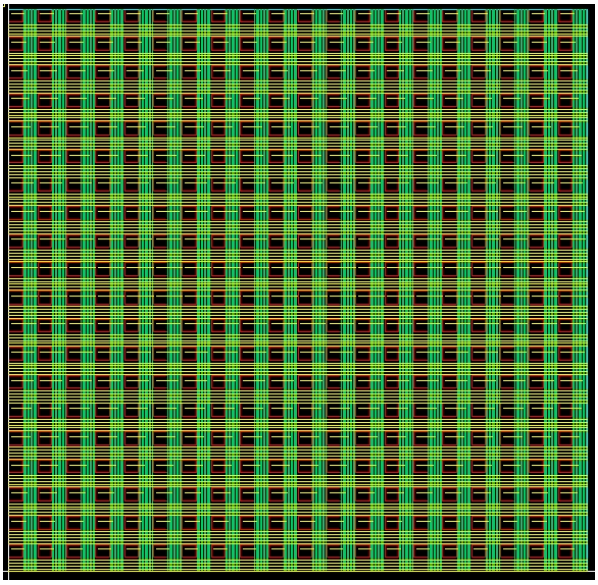


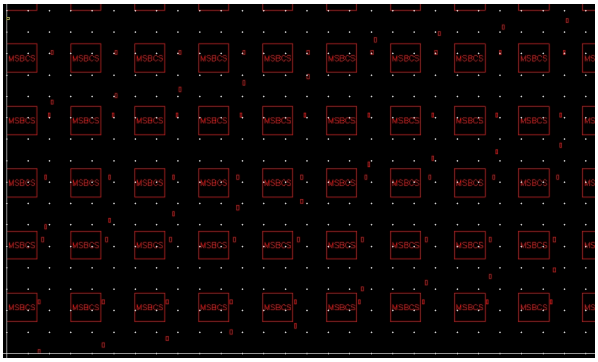


P.S.: 上圖的Via34(CS=64)只展示一部分而已



- CS = 100





P.S.: 上圖的Via34(CS=100)只展示一部分而已