(1) Your name and student ID

Name : 陳俊元

Student ID : 110065536

(2) How to compile and execute your C/C++ program and give an execution example.

```
--How to Compile
 In this directory, enter the following command:
 $ make
 It will generate the executable files "hw5" in "HW5/bin/".
 If you want to remove it, simply enter the following command:
 $ make clean
--How to Run
 In this directory, enter the following command:
 Usage: ../bin/<exe> <k-value: 4, 16...> <.def file>
 e.g.:
 ../bin/hw5 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.

以下皆把 CS 的數量定義為 numCS, 開根號 numCS 為 sqrt_numCS，

Step 1 :

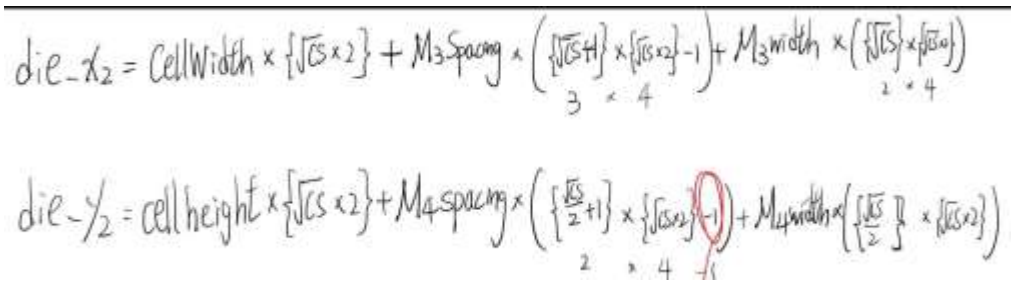
$$bx_2 = 7.1 \times 4 + 0.31 \times (3 \times 4 - 1) + 0.44 \times 2 \times 4 = 35.33$$

$$by_2 = 6.6 \times 4 + 0.49 \times (2 \times 4 - 1) + 1 \times 1 \times 4 = 33.83$$

上圖為講義的公式

先看 bx2，7.1 為 cellwidth，因此後面的 4 代表的是一個 die 在一個 row 上會有幾個 cell，因此我們將 4 改為 sqrt_CSnum*2，而。0.31 為 spacing，3 代表的意思為一個 col 有 2 個 ME3，而空隙要+1，因此 3 改為(sqrt_CSnum+1)[sqrt_CSnum 代表 ME3 的數量]，後面的 4 表示的是有幾個 col(計算方式為 sqrt_CSnum*2，最後因為邊界會少一個 spacing 因此-1。0.44 表示的是 ME3 的 width，因此我們要計算的是 ME3 的數量，每個 col 有(sqrt_CSnum 個)共(sqrt_CSnum*2 個 col)因此為 M3_width * sqrt_CSnum * (sqrt_CSnum*2)

by2 基本的邏輯相同，不同的只有計算 ME4 的數量變成是 sqrt_CSnum/2，因此把上述提到關於 ME3 數量的部分改成 sqrt_CSnum/2 即可，手寫公式如下



Step2 :

$$off_y = M4_{spacing} + M4_{width}$$

$$D_y = CS_{height} + M4_{spacing} \times 2 + M4_{width}$$

$$D_x = CS_{width} + M3_{spacing} \times 3 + M3_{width} \times 2$$

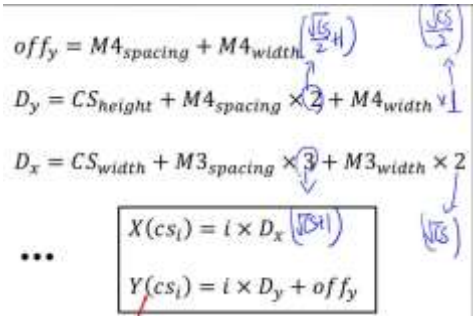
\

step2 要改變的部分為 *2 *3 的地方，其中比較容易遺忘的是 offy 其實 M4width 後面有一個*1

我們要把它改成 sqrt_num/2 +1 因為當 CS 變成 16 以上 ME4 不會是 1，因此要把 ME4 的數量考慮進去

而 Dy 把 2 跟 step1 一樣提到的(sqrt_CSnum/2+1)因為這邊要計算的是 spacing 數量也就是 M3 數量+1，後面的 M4width *1 的 1 要改成(sqrt_CSnum/2)

Dx 的部分把 3 改為 sqrt_CS+1，2 改成 sqrt_CS，計算 spacing 數量時就是將 ME3 的數量計算出來然後+1。



index 的部分就按照講義的 i, j 填入即可。

Step 3 : step3 的計算沒有甚麼需要改變的，都按照講義的數字填入即可，讓人比較搞混的是這邊的 Dx 跟前面的 Dx 其實是不一樣的，而另外一個比較需要注意的是原本講義給我們的矩陣是 4*2 的矩陣，這邊要改成 sqrt_numCS*2 X sqrt_numCS 大小的矩陣，CS4 之所以是 4*2 就是因為 sqrt(4) *2 X sqrt(4) 因此是 4*2 的矩陣。

Step 4 : 講義覆蓋掉的部分滿有趣的，原本有點看不懂，後來發現就是每個 index 都用紅綠藍黃的順序去對應的 array 把值拿出來因此先此 CS4 來舉例，我們要找的順序就是(3, 3)(3, 2)(2, 3)(2, 2)，

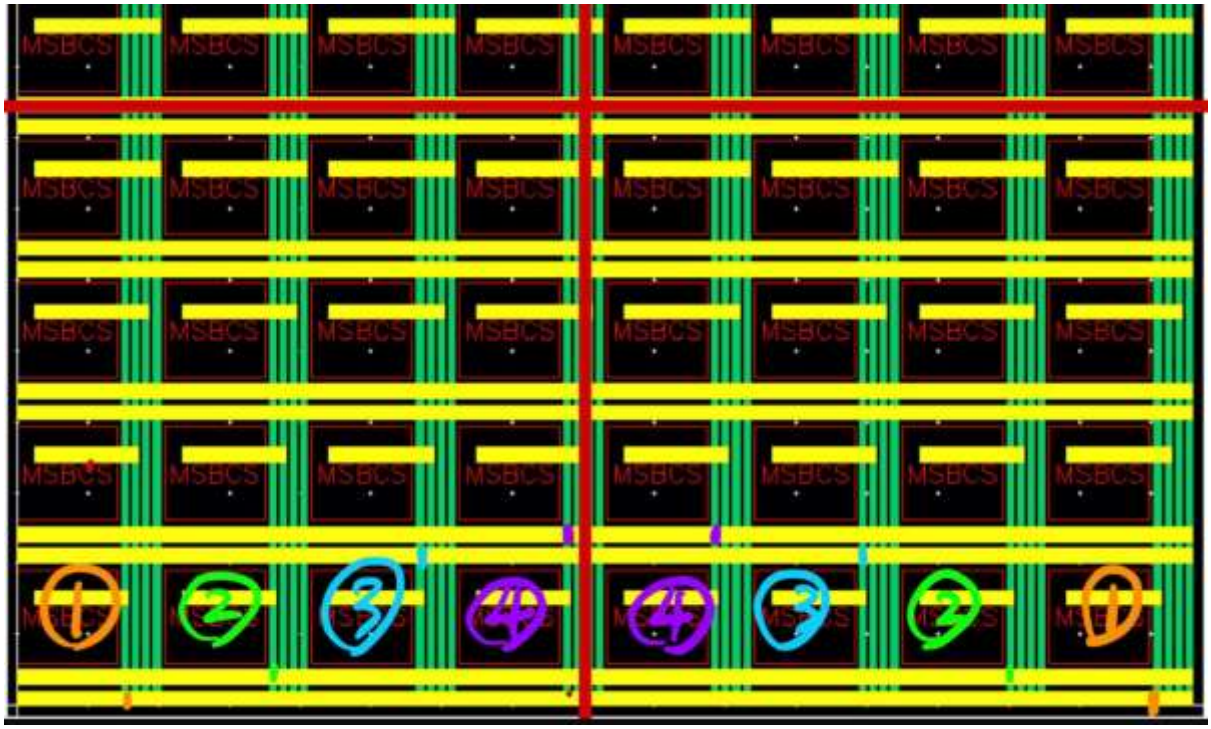
```
# right top corner units
inst_name = 'Metal4_drain_' + str(i * 2 + j + 3 * 4)
x1 = cs_array[3-i][3-j]._x + CS_X1_TO_DRAIN
x2 = ME3_specialnet[3-i][j]._x2
y1 = cs_array[3-i][3-j]._y + CS_Y1_TO_DRAIN
y2 = y1 + M4_WIDTH
ME4_specialnet_drain[3-i][3-j] = SpecialNet(inst_name, layer, x1, y1, x2, y2)
```

因此填入的順序即為圖所示，而要改為一般化的部分，由於在 CS4 的情況我們會有 4 個顏色因此就是都是[3-i][3-j]等等，而這個 3 的意義就是 sqrt_numCS*2 -1 ，因此只需要把宣告 inst_name 的 3 改為 sqrt_numCS*2 -1 以及下方要尋找對應 array 的 index 的 3 改成 sqrt_numCS*2 -1 即可。

Step 5 : 一開始要先知道 ME4 port 的數量計算方式為 sqrt_numCS*2 乘上 sqrt_numCS/2，原本講義只有出現 4*1，我一開始一直以為一直都是 sqrt_numCS*2 *1 這邊卡住了一段時間，sqrt_numCS/2 代表的是每一個 row 會有幾個 ME4 port，而 sqrt_numCS*2 算的是有幾個 row，這邊的一般化也沒什麼特別的。

Step 6 : 和 step4 相似，只要抓到紅綠藍黃這個邏輯，剩下就是把原本的 3 (sqrt_num*2-1)替換掉即可。array 的長度原本是 4*4，一般化後為 sqrt_numCS*2 X sqrt_numCS*2。

Step 7 : step7 是讓我思考最久的部分，我甚至一個一個 loop 展開因為要找到對應的 indexing 方式，



這邊我想附上我的思考圖以方便解釋，比較特別的是我把 array 拆成(numCSX2)因為我注意到 Via34toME4 port 的數量剛好等於 numCS*2，而擺放的時候我又剛好可以對稱的擺放，step1~6 如果有需要用到以 x/y/原點對稱時我們用的都是先把一個 col 的 CS 走完再往下一個的方式而在這邊若是用一樣的思考邏輯在取得 y 上面會有點複雜，因此我改為先考慮完一個 row 再往上，這樣剛好在取得 via 的 y 上面剛好可以一路從 00, 01, 02….0n-1 10 11 ….的 ME4 port。

這邊我想先提一下我們要取得 via 的 xy，X 的來源剛好可以從每一個 CS 的 via34ME4 drain to ME3 取得，而 y 可以從對應到的 ME4 port 取得。因此問題就變成我要怎麼在 for loop 中描述對應的 array。

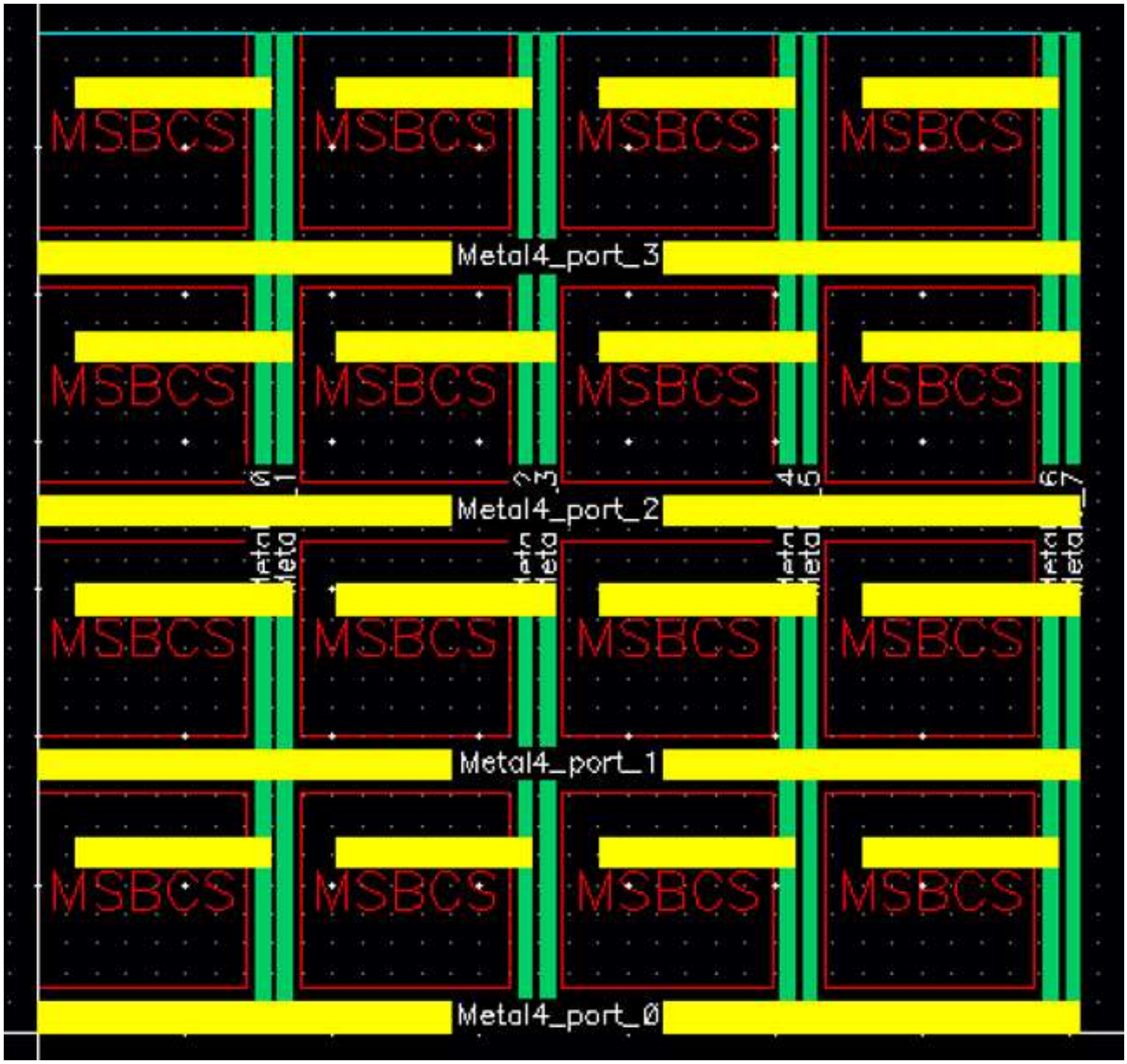
因此，我先將內部迴圈跟 ME4 的數量對齊，而因為迴圈裡面用兩個 block，代表相同顏色，最外面的迴圈就理所當然的是 sqrt_numCS*2，這樣做的好處是我 y 就很單純的跟著迴圈走即可，每一個顏色跟好讓 y 從 00, 01, 10, 11 一路走下去(以 CS16 舉例)，接下來只要煩惱 x 即可。

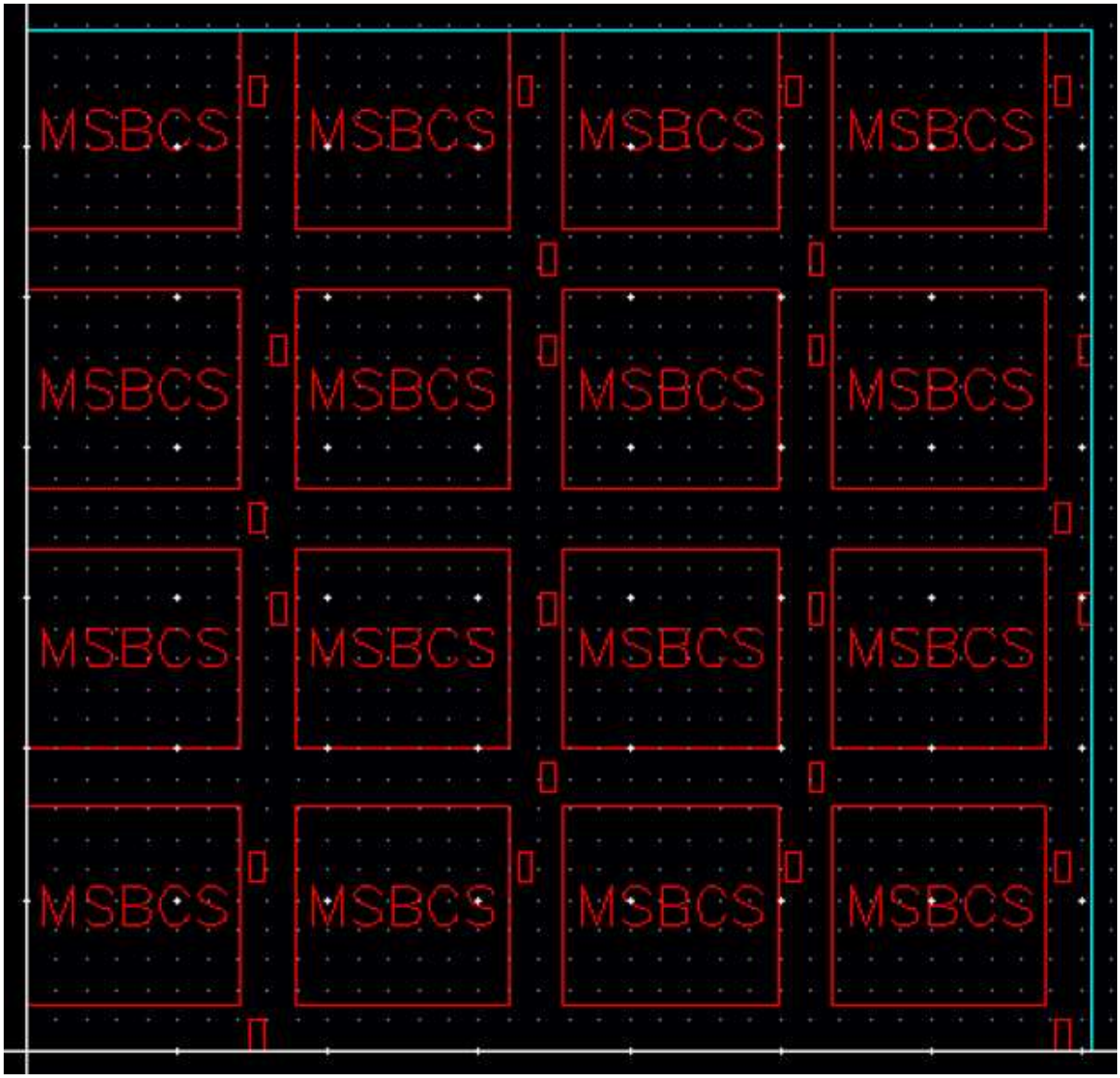
Via34_drain2ME3 以下簡稱 Sarray，Sarray 的第一維要出現的順序，以兩個 block 來說分別是 07, 16, 25, 34….這樣一直循環，又剛好知道他是每 sqrt_numCS 循環一次，因此我們可以只要用一個 count 去記錄他的關係即可，而兩個 block 加起來要等於 2*sqrt_numCS-1，因此另外一個 block 就用一樣 block 的 indexing 去減掉即可，而第二維就比較簡單了，我希望他每 sqrt_numCS 次循環一次，從 0, 1, 2, 3.....sqrt_numCS-1, 0, 1…因此就用 i % sqrt_numCS 即可

(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

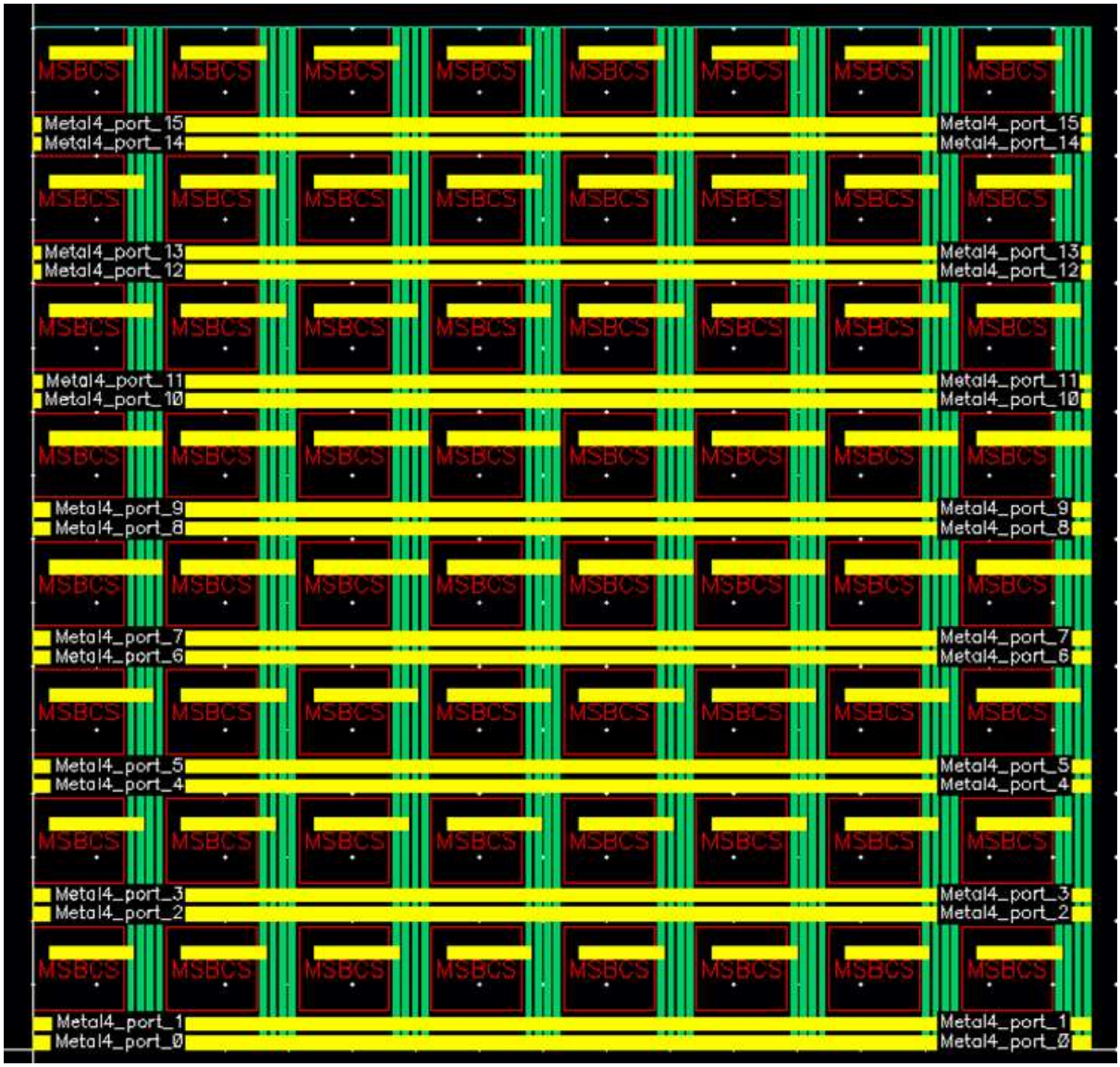
.

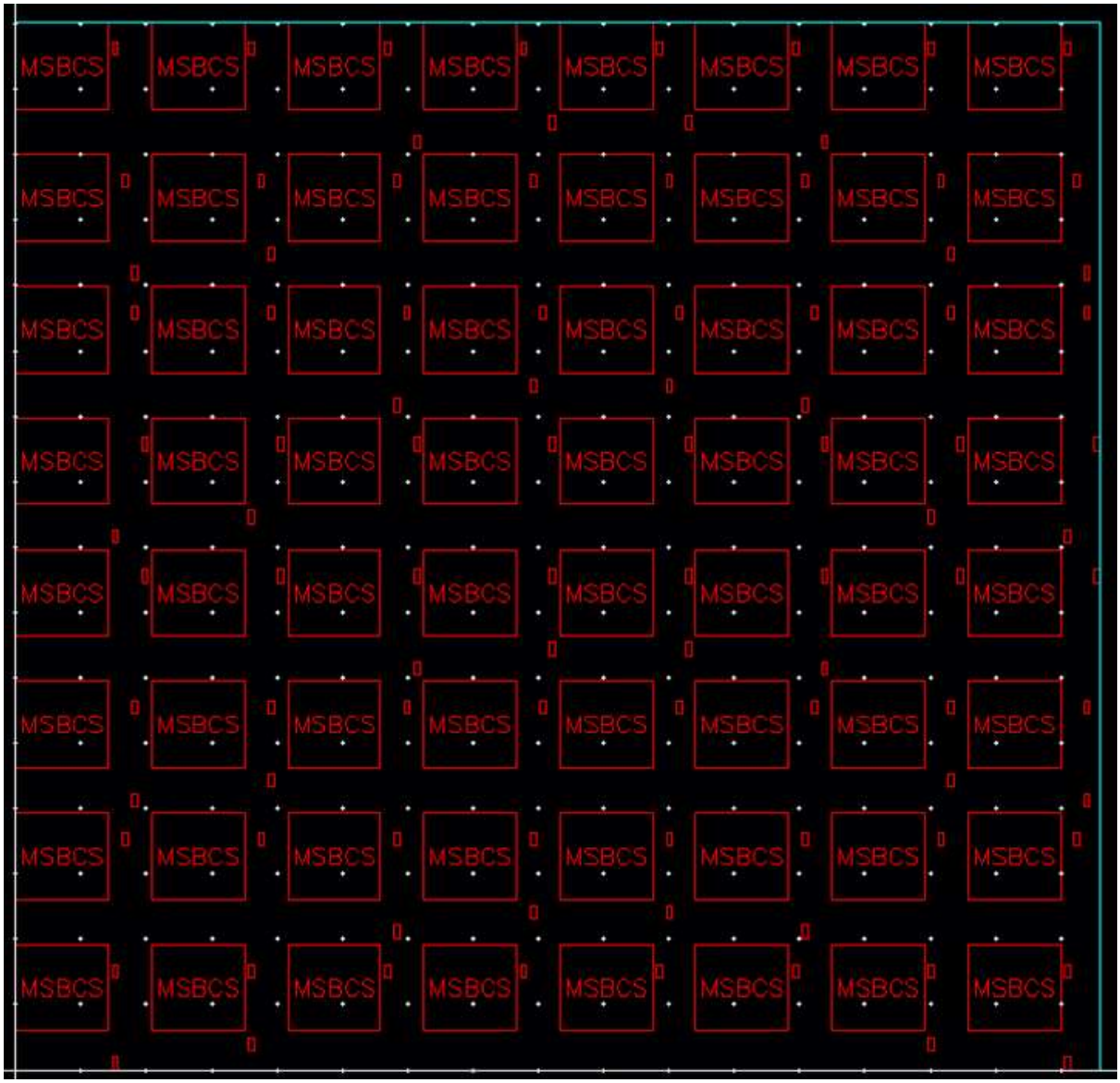
4:



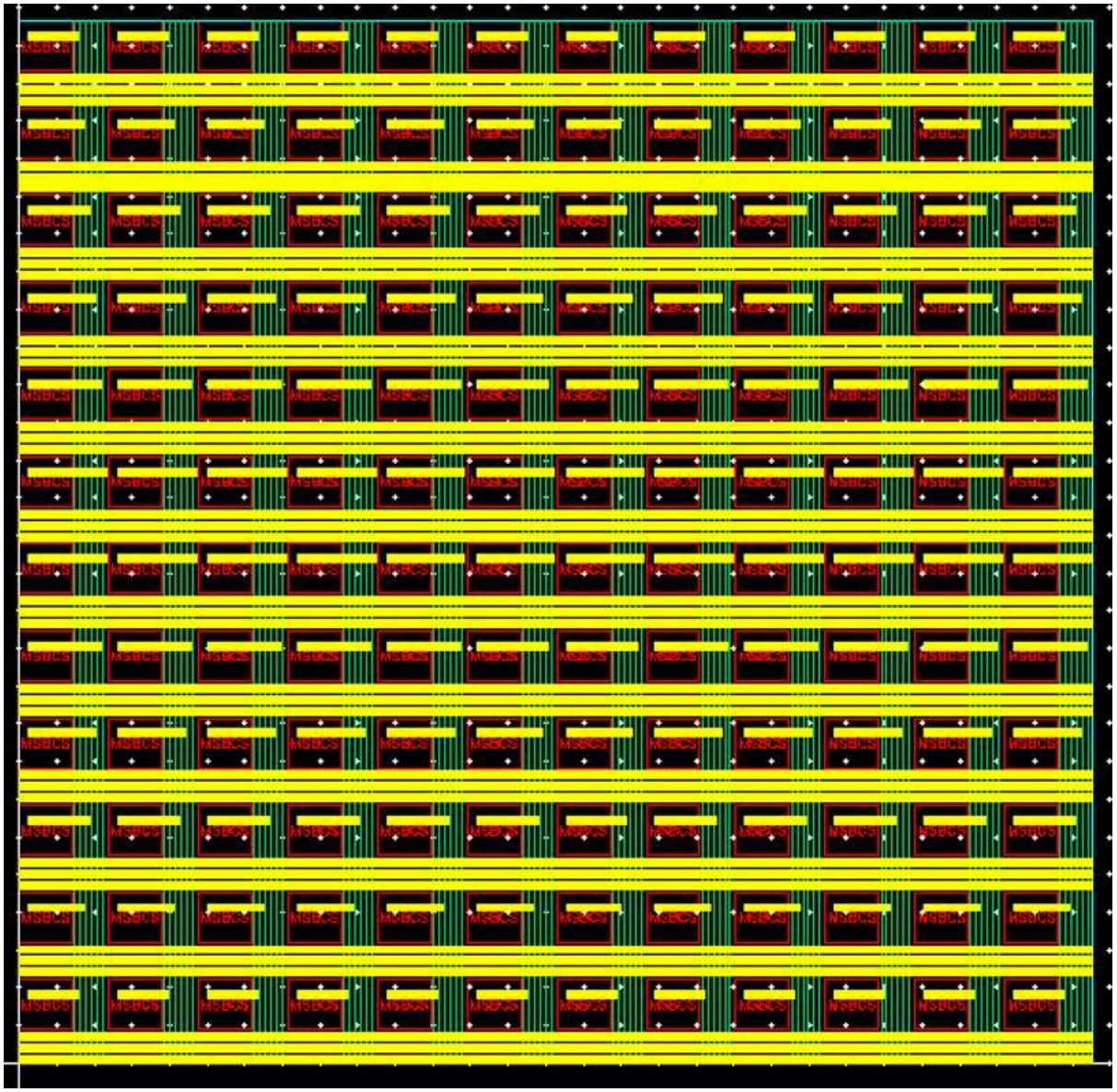

MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS

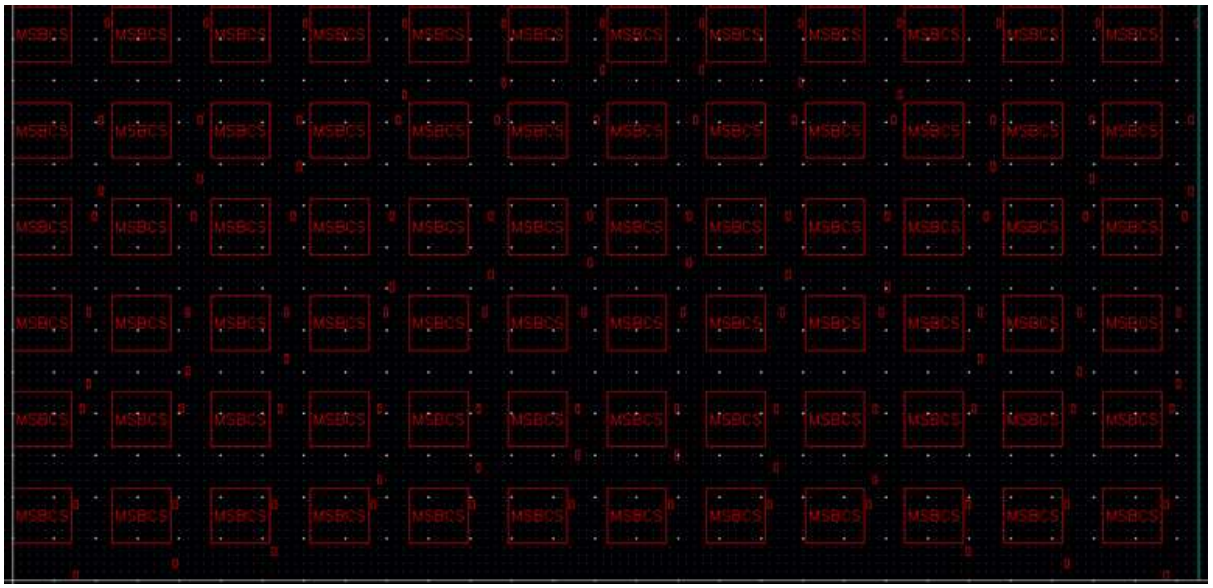
16:


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

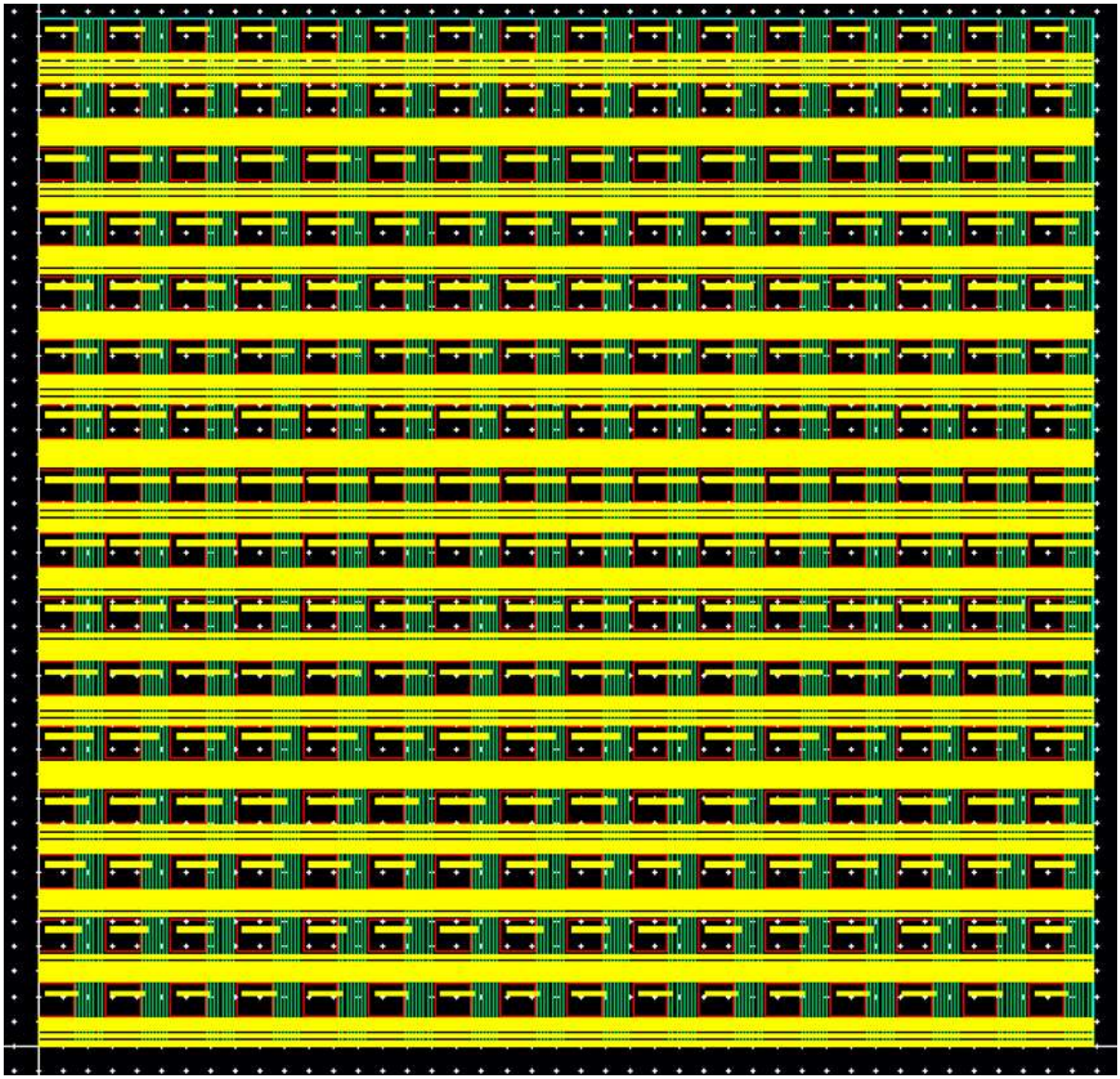

MSBCS

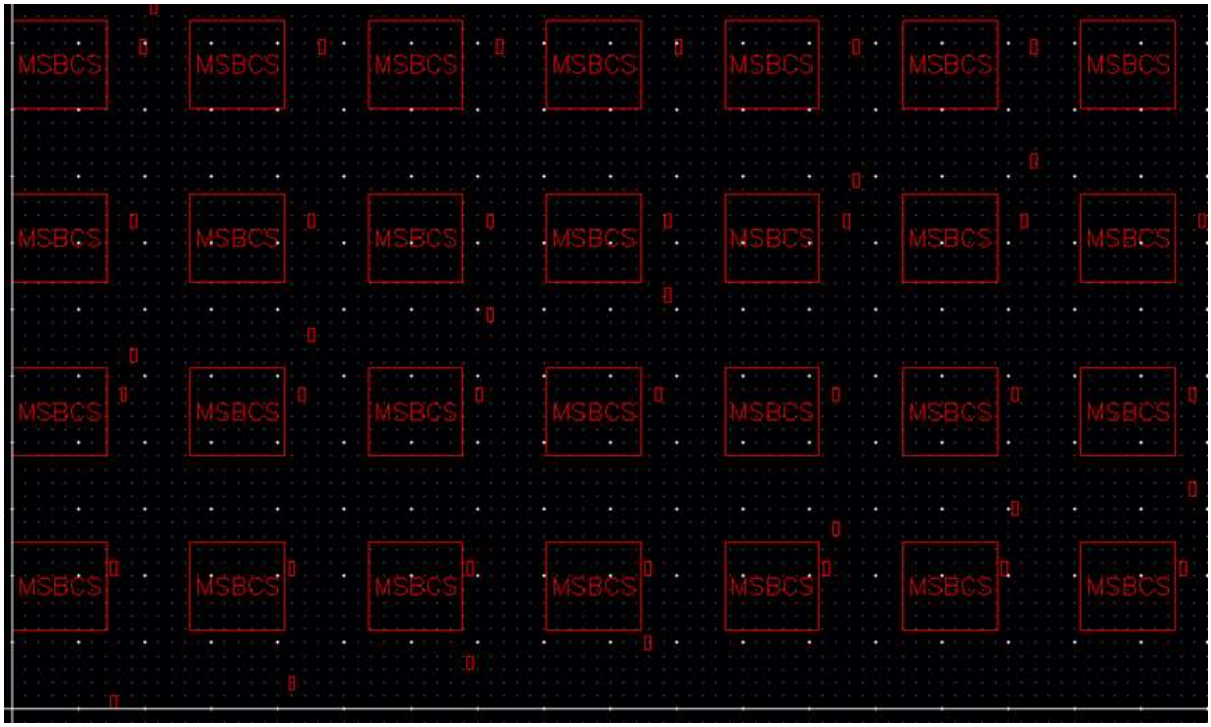
36:



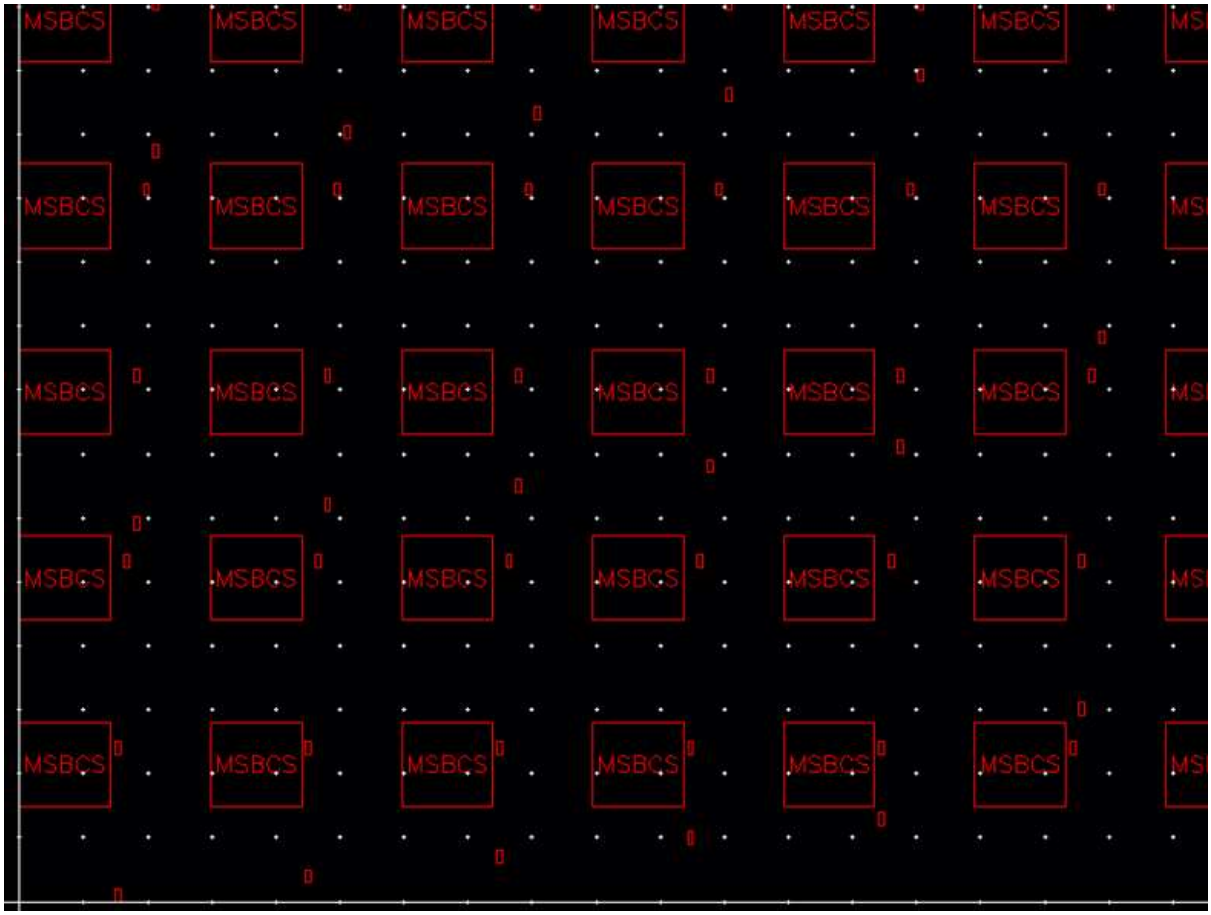


64:




MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS
MSBCS

100:



PS. 36, 64, 100 的 via 無法全部截圖出來，因此只截了部分