

# Color-aware Routing for Double Patterning

## Group19

B03901141 梁雲翔 R04525061 葉峻孝 B02204048 莊奇凡

### 壹、題目介紹

當半導體的製作的進程縮小到 32nm 或以下的時候，過去所使用波長浸潤式顯影就無法應付這樣的需求，因為線寬的間距已經太小，單次曝光無法滿足這樣的需求，為了因應這樣的困難，而發展出的雙重曝光，其原理是將曝光分為兩次，例如將特定某些線先曝光一次，剩下的線再曝光，最後再將其疊在一起就完全線路的建立，如果在兩次曝光的曝光序在最小線寬中太過接近就有可能會有短路的問題發生。

在這次的題目中總共是四層金屬層，每個金屬層都各有兩種不同顏色來代表不同的曝光序，如果在相同金屬層中的最小線寬其相鄰的線顏色相同就會有 Color conflict 的問題，其結果就是再曝光後就會有接線**短路**的現象。

### 貳、問題分析

問題主要圍繞在處理繞線、Color conflict 與減少在不同金屬層中跳換的次數。

繞線上會有 Blockage 阻擋與在同一層金屬層是不能有交叉經過的繞線的限制，而最主要要達到的目標是如何在限制下找到最佳最短路徑將點連接在一起。

要如何讓 Color conflict 不要發生並且

讓其的 Color 分配能夠平均。

而跳層的問題就會是要如何讓跳層的最小，而在這次我們會將問題的重點放在繞線跟 Color conflict。

### 參、使用的演算法

再繞線的問題上，我們使用基於 Breath-first-Search(BFS)的 Lee Algorithm 來實作繞線的部分，從起點開始向四個方向去尋找終點，如果沒找到就在從這四個點分別向其可搜尋方向去尋找終點，並在每個走過的點標記上一個點加一的序號，最後到目標點後在從目標點開始根據編號回溯到起點，而此演算法能夠確保使用的距離是兩點之間最短的(Shortest Path)。

解決 Color conflict 上所用的方法是基於 Lee Algorithm 再繞線的過程中設置 Blockage 來做為解法。

### 肆、本組解法

在繞線的部分，我們使用 Lee Algorithm 來做為基礎的演算法，但 Lee Algorithm 的缺點會有相當高的時間及空間複雜度，為了要減少其記憶體的使用，我們根據 Akers's

Obvervation(Akers 1966)將原本是序號的編號改成 4、5、6、8 分別代表左、下、右、上來做為回溯這樣就只需要 4 個 bits 的就可以完成回溯，降低記憶體的使用量。

在本次的資料中 Pin 點座標是以 0.5 為一個單位，在程式操作上來做 Grid Routing 會有點麻煩，故我們將所有的 Pin 座標乘於 2，將所有的 Pin 點座標都改成整數方便在程式中陣列的操作，最後輸出在除於 2 回復到原先的 Pin 座標。

繞線時 Critical Net 先繞，主要的原因是 Critical Net 是由多個 Pin 點所組成，根據 Lee Algorithm 來繞線會形成 Steiner Tree，所以先將 Critical Net 較為複雜的線路連結繞好之後再繞較為單純的 Not Critical Net 這樣可以降低 Net Length。

在 Color conflict 上，我們是基於只要不要讓同一個金屬層的線繞在最小線寬上，這樣就不會有 Color conflict 的問題，作法是當繞線結束就將原本繞完線的最小線寬兩倍設成不能經過的 Blockage，當下一次有在同樣金屬層的線經過時就會因為此 Blockage 而繞開。

伍、實驗結果

這次比賽單位給了兩筆測資，我們會用這兩筆測資來作為程式效能分析與正確性，程式的執行平台為 Ubuntu 16.04。

在 Fig.1 上是兩個 Case 的比較，在這兩個 Case 中並沒有顯著的時間花費較長，有可能是因為這兩個 Case 的

資料量不大，但在 Memory 的部分已經使用到 29MB，我們認為這是 Lee

Case	Net	Pin	Time(Sec)	Memory(MB)
1	33	100	0.012	29.012
2	68	140	0.011	29.012

Algorithm 的缺點所造成的。

(Fig.1)

為了要方便確認是否結果有符合我們所預期的，我們在程式中加入將電路視覺化的函式，從 Fig.2 來看在上面 0 是電路空白處、2 是原先 Blockage、9 是目標點、7 是繞線結束後所設的 Blockage 而 1 是繞線時線所走的地方，Fig.2 是 Case2 的結果，可以看到在 1 的旁邊都有作為繞線結果的 Blockage 的 7 所包圍，這就代表如果有其他繞線過去就不會有 Color Conflict 的問題發生。



(Fig.2)

陸、複雜度分析

在 routing.cpp 中：  
設 M, N 分別代表 layout 的長和寬。  
bfs function (第 44-290 行)與 Lee's Algorithm 大致相同。BFS(第 94-132 行)和 backtrace (第 182-268 行)的時間複雜度皆為 O(MN)。  
routing function (第 304-441 行)分為初

始化(第 308-392 行)和實際繞線(第 397-420 行)兩大部分。前者包含兩層 for 迴圈，共執行 MN 次迭代，因此複雜度為  $\Theta(MN)$ 。設 input 中的總 pin 數為 p，則後者中兩層 for 迴圈的 iteration 總數為  $O(p)$  次。每次迭代中執行一次 bfs，並更新一次 layout( $\Theta(MN)$ )，故整體時間複雜度為  $O(pMN)$ 。

## 柒、結論

基於 Lee Algorithm 來完成 Multiple Layer 的 Double pattern 從 Case1 跟 Case2 的結果來看都有不錯的速度還有繞線完整度，但是因為 Lee Algorithm 的特性所以對於記憶體的使用沒這麼有效率，再加上測資所包含的 Pin 數並不大，所以可能有些在繞線上的問題並沒有顯現出來，但在目前的測資是可行且相當良好，如果測資一大可能就要考慮較為複雜的演算法，例如:Global Routing 跟 Detail Routing。

## 捌、個人貢獻

R04525061 葉峻孝

BFS(44-301)與 Routing(304-420) 程式部分

B03901141 梁雲翔

Visualization(422-429)、I/O(22-258 in main.cpp)、複雜度分析書面報告

B02204048 莊奇凡

Visualization(增加線的辨識度)、投影片、書面報告

## 玖、Reference

- [1]. Chen, H. Y., & Chang, Y. W. (2009). Global and detailed routing. In *Electronic Design Automation* (pp. 687-749).
- [2]. Hong, X., Xue, T., Kuh, E. S., Cheng, C. K., & Huang, J. (1993, July). Performance-driven Steiner tree algorithm for global routing. In *Proceedings of the 30th international Design Automation Conference* (pp. 177-181). ACM.
- [3]. Sarrafzadeh, M., & Lee, D. T. (1989). A new approach to topological via minimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(8), 890-900.
- [4]. Marek-Sadowska, M. (1984). An unconstrained topological via minimization problem for two-layer routing. *IEEE transactions on computer-aided design of integrated circuits and systems*, 3(3), 184-190.
- [5]. Akers, S. B. (1967). A modification of Lee's path connection algorithm. *IEEE Transactions on Electronic Computers*, (1), 97-98.