CS6135 VLSI Physical Design Automation

Homework 5: Placement Legalization

112062682 張宇越

1. How to compile and execute my program :

- **Compile:**
  Enter HW5/src/ and make, it'll generate the executable file to HW5/bin/
  $ cd HW5/src/
  $ make
- **Execute:**
  $ cd HW5/bin/
  $ ./hw5 <testcase file> <output file>

  E.g.
  $ ./hw5 ../testcase/public1.txt ../output/public1.out

2. The screenshot of the result of running the HW5_grading.sh:

```
[g112062682@ic21 ~/HW5_grading]$ ./HW5_grading.sh
+-----------------------------------------------+
|                                               |
|    This script is used for PDA HW5 grading.   |
|                                               |
+-----------------------------------------------+
host name: ic21
compiler version: g++ (GCC) 7.3.0

grading on 112062682:
 checking item           | status
-------------------------|--------
 correct tar.gz          | yes
 correct file structure  | yes
 have README             | yes
 have Makefile           | yes
 correct make clean      | yes
 correct make            | yes

  testcase | total disp. |  max disp. |    runtime | status
-----------|-------------|------------|------------|--------
   public1 |     8350243 |       2944 |       0.10 | success
   public2 |    35023268 |       5281 |       0.40 | success
   public3 |    47551034 |       4809 |       0.19 | success
   public4 |     4672689 |        346 |       0.82 | success
   public5 |     8039563 |        151 |       1.89 | success
+-----------------------------------------------+
|                                               |
|   Successfully write grades to HW5_grade.csv  |
|                                               |
+-----------------------------------------------+
```

3. The details of your implementation and different with the ISPD-08 paper:

(1) readInput():

依 testcase 分段讀取 maxDisplacement、cells、blockages、rows；同時把 Row 初始化成單一 subRow。

(2) sliceRows():

- 先對 blockages 依 X 排序，降低重疊判斷成本。
- 對每顆 blockage，若其 Y 區間與某 Row 有交集，呼叫 Row::slice() 把該 Row 的 subRows **橫向切割**；未重疊的部分保留原 subRow。
- slice() 內部會依四種相對位置（完全左/完全右/完全覆蓋/中間切割）調整或插入新的 subRow。

(3) abacus():

- 先將所有 cell 按原始 X 座標排序。
- 對單一 cell：
  1. **選 Row**：
     getRowIdx() 找 Y 位移最近的 Row 作為起始；再往上／下展開。
  2. **選 subRow**：
     getSubRowIdx() 掃描可容納的 subRows，以“額外水平位移最小”為關鍵指標。
  3. placeRowTrial():
     - 若新 cell 放在 lastCluster 右邊且不接觸，就把它獨立成一個新 cluster。
     - cell 和 lastCluster 發生重疊：先把 cell 合入 lastCluster，取得更新後的 clusterX；若新位置仍壓到左側的 cluster，就沿鏈結向左持續 collapse。
     - Penalty Check:
       若 addPenalty == true，自右向左走訪剛才合併的 cluster 鏈，將每顆 cell 先對齊 site 再估算位移。只要有任何 cell 超過 maxDisplacement，此 subRow 直接宣告失敗。
     - **雙階段搜尋**：
       第一輪(addPenalty = true)，先嘗試在**位移受限** 的情況下找到合法 subRow。若第一輪全部失敗，才放寬最大位移限制再搜一次(addPenalty = false)，確保演算法最終一定能為 cell 安排位置。
  4. placeRowFinal():
     真正修改 subRow / Cluster 狀態並更新 freeWidth。

collapse 時以 while loop 反覆將左側 cluster 併入直到不再重疊 。

- determinePosition():
  沿 subRow 由左往右走訪 Cluster，將 Cluster 起點傳遞給群內 cell；並以 alignToSite() 對齊到 site 網格。

(4) writeOutput():
計算 cell 的總位移和最大位移及座標，輸出結果。

(5) Different:
由於原論文未將固定元件（即 blockage）納入考量，因此必須先把每條 row 切分成多個 subRow。對於每一條 row，先檢查是否被 blockage 占據；若有，就把該 row 切成多個 subRow。重複此流程，直到所有 row 皆完成檢查。

4. How did you handle the row if it is divided by the blockage:

(1) 每一塊 blockage 的左右邊界(先各自做一次 site 對齊:左邊 floor，右邊 ceil)，得到一段 [blkL, blkR)。

(2) 走訪目前 row 裡的每一段 subRow（[minX, maxX)），依照它跟 [blkL, blkR) 的相對位置，決定要**原封不動保留**或**直接整段砍掉**或**把原 subRow 切成兩段（左、右）並更新它們的 minX / maxX**，切完後，subRows 陣列就只剩「真正可放置的連續區段」。

5. What methods did you use to handle the max displacement constraint:

第一輪 (addPenalty = true)

- 嚴格檢查位移；只要任何候選 subRow 使 cell 位移超過 maxDisplacement 就視為失敗。

第二輪 (addPenalty = false)

- **只有** 第一輪完全找不到合法位置時才執行。
- 關閉位移限制以保證演算法總能放置所有 cell；同時仍採用「距離優先」的花費函式，盡可能保持位移最小化。

6. What tricks did you do to speed up your program or to enhance your solution quality ?

我用來降低執行時間的訣竅，是直接挑選距離最近的 subRow，而不是把所有

subRow 都搜尋一遍。如此一來，總運行時間會更短。

7. What have you learned from this homework ? What problem(s) have you encountered in this homework ?

為了滿足 maxDisplacement，又要保證能放進所有 Cell，我實作了「嚴→鬆」的雙階段搜尋；這讓我第一次感受到**演算法正確性**與**工程穩健性**之間的折衷。一開始 collapse 時只更新最右 cluster，忘記同步調整左側鏈結，導致下一顆 cell 進來後陷入無限合併迴圈。