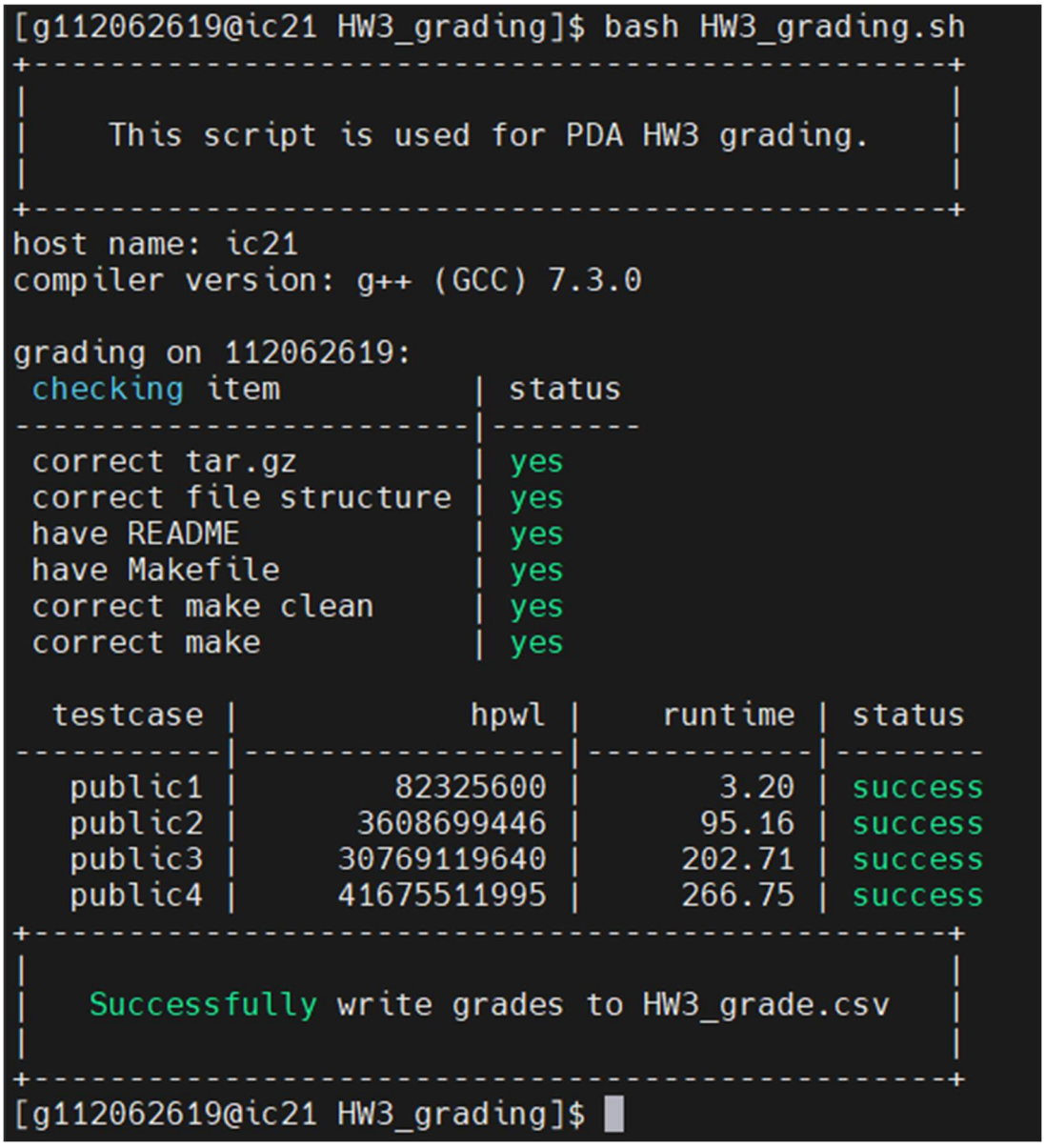
CS6135_HW3_112062619_report

1. 112062619，陳航希

2.



3.

LEF 三種 types:

1. LAYER

LEF 中的 LAYER 用來描述每一層金屬或孔洞層的製程規則，包括：層的名稱、是用來布線的金屬層還是用來上下接層的孔洞層（CUT）、推薦的走線方向（水平或垂直）、最小線寬與間距、pitch 等等。這些資訊提供 router 依循製程規範來合法布線。

3. PIN

在 MACRO 區塊內的 **PIN** 用來描述元件的引腳，包括引腳的方向（如 INPUT / OUTPUT / INOUT）、用途（例如 signal、power）、以及該 pin 在不同金層上對應的 PORT 幾何形狀。

2. VIA

VIA 描述一個跨層接點的幾何結構，例如它連接哪兩層金屬、via 本身的尺寸、以及與上下金屬 pad 的對應位置。LEF 會以座標方式（通常以原點 (0,0) 為中心）描述 via 與其上下金屬接觸區域的形狀，提供 routing 工具合法地放置跨層接點。

DEF 三種 types:

1.TRACKS

TRACKS 主要用來告訴 router 金屬線應該沿著哪些固定的軌道走。
它會指定軌道的方向、第一條軌道在哪裡、總共有幾條，以及軌道之間的間距。會替金屬線畫出「預設的行車道」，讓後面的 routing 工具照著這些軌道去佈線，避免金屬線亂擺位置。

2. GCELLGRID

GCELLGRID 用來把整個晶片切成一個個比較大的格子（g-cell），給 global router 當作「地圖」。它會定義分隔線方向、第一條分隔線的座標、有幾條線以及間隔多少。

3. + USE POWER

當 DEF 裡面的某個 pin 或 net 有 + USE POWER，意思是這個腳位/網路

是電源線（例如 VDD），不是普通訊號。

4. Describe the details of your implementation

這份作業最主要的挑戰在於在 300 秒內，對所有 movable cells 做多次 local 改善同時避免違反 legalization。

我的流程如下

```
Read LEF / DEF → 建立資料結構
↓
buildAdjacency()   // 建立 cell → nets 對應
initAllNetBBox()   // 計算所有 nets 的 bounding boxes
↓
for W = 2..Wmax:
    buildSegments()
    local reorder (window-based)
    single-cell sliding
    legalize segments
↓
Run globalSwap()
↓
Final legalization & output
```

主要資料結構

```
struct Segment {
    int rowId;
    int xL, xR;
    vector<int> cells;
};
```

Segment 建立方法為對 row 上的 fixed cells 做線段合併，沒有重疊區域就會是 free segment，接著再把 movable cells 依照 x 座標塞進每個 segments。

在前半段迴圈的過程裡面只有做局部的變化 也就是 window-based reordering, single-cell sliding，在 reordering 的時候針對 w 個連續的 cells 會做列舉所有的 permutations，嘗試重排+放進去，計算影響到的 nets HPWL 變化，如果有改善就採用。

在 Sliding 方面，我針對每個可移動 cell 計算他的合法滑動範圍 [left, right],微調並嘗試所有位置，如果 HPWL 降低就採用新的位置。

在做完上面的局部變化以後，我加入了 global swap 進去，發現對 public2,

public4 特別有效。

我實作的方訪是首先對 cell cid 使用 FindOptRegion() 找出最可能改善的 x y 區域，接著投影成 row index 範圍並在 rowSegIdx 中快速搜尋可能碰到的 candidate cell，去掉不行的以後再用 evalSwapDelta() 測試 HPWL 差異，如果有改善就交換位置。

做完上面的 global swap 以後因為 global swap 沒有去做 legalization 的確認，所以最後再做一次 legalization, 確保 cell 對齊 site, 不會 overlap, y 對齊 row

5. What have you learned from this homework? What problem(s) have youencountered in this homework?

在這次詳細排置的實作過程中，我遇到不少意料之外的瓶頸。最明顯的是 global swap，一開始我直接讓它搜尋所有可能的交換對象，結果程式直接跑不完。後來我想到很多 cell 彼此沒有共同的 net，就算交換也不可能改善 HPWL，所以加入 shared-net 的過濾之後，速度才變得能接受。另一個意外是 segments 的建立成本比想像中高。原本以為只是簡單地把 fixed cells 分段，實際寫起來發現 row 多、cell 多時，反覆重建 segments 不但慢還很容易出錯。後來我用 interval merge 搭配 rowSegIdx，才讓 segment 查詢變得比較輕鬆。

雖然整體的實作根原本的論文差了不少,例如 local reorder 通常會搭配更複雜的 DP 做, 但我是直接用 permutation 找, 時間雖然長了很多但因為是直接列舉全部所以實作上輕鬆了不少, global swap 的部分, 通常會用更聰明的圖模型來做但我的方法也簡化了很多也就是找附近 row 的 candidate、用 pruning 過濾掉不可能改善的 pair、然後做 HPWL 測試，不過這個方法也成功地過了 baseline,算是相當不錯。總體而言學到了很多實用的方法。

6. AI Tool Usage Disclosure 在這份作業的撰寫過程中，我使用 GPT 作為輔助工具，用來協助我除錯與程式重構