

# CS6135 VLSI Physical Design Automation

## Homework 2: Two-way Min-cut Partitioning

1. 112062525 蔡品棠

### 2. How to compile and execute my program:

```
HW2 > src > README
1  How to change the target testcase:
2    in /HW2/src/Makefile, edit the INPUTFILENAME
3    e.g. INPUTFILENAME = public* (just replace the * with a number within 1-6)
4    e.g. INPUTFILENAME = public1
5
6  How to compile & run & verify:
7    in /HW2/src file, give "make" command in terminal
8
9  How to compile:
10   in /HW2/src file, give "make build" command in terminal
11
12 How to run:
13   in /HW2/src file, give "make run" command in terminal
14
15 How to verify:
16   in /HW2/src file, give "make verify" command in terminal
17
18 How to run & verify:
19   in /HW2/src file, give "make test" command in terminal
20
21 How to clean:
22   in /HW2/src file, give "make clean" command in terminal
```

如 README 檔所示，進入/src 資料夾後，只要在終端機下「make」指令，就能一鍵跑完編譯與執行流程。

e.g. \$ cd ./HW2/src

\$ make

若要改變 testcase，於/src 的 Makefile 檔裡編輯「INPUTFILENAME」即可。

e.g. \$ vim ./HW2/src/Makefile

in Makefile:

INPUTFILENAME = public1

若要分別進入/src 編譯和/bin 執行也沒問題，在/src 下 make build 指令後進到/bin 輸入以下指令即可：

\$ ./hw2 ../testcase/public1.txt ../output/public1.out

### 3. The final cut size and the runtime of each testcase

\*因為我在 initial partition 使用 random shuffle，因此每次產出的 cutsize 都會有些微差異

public1: cut size = 686, runtime = 0.16 secs

public2: cut size = 12708, runtime = 35.72 secs

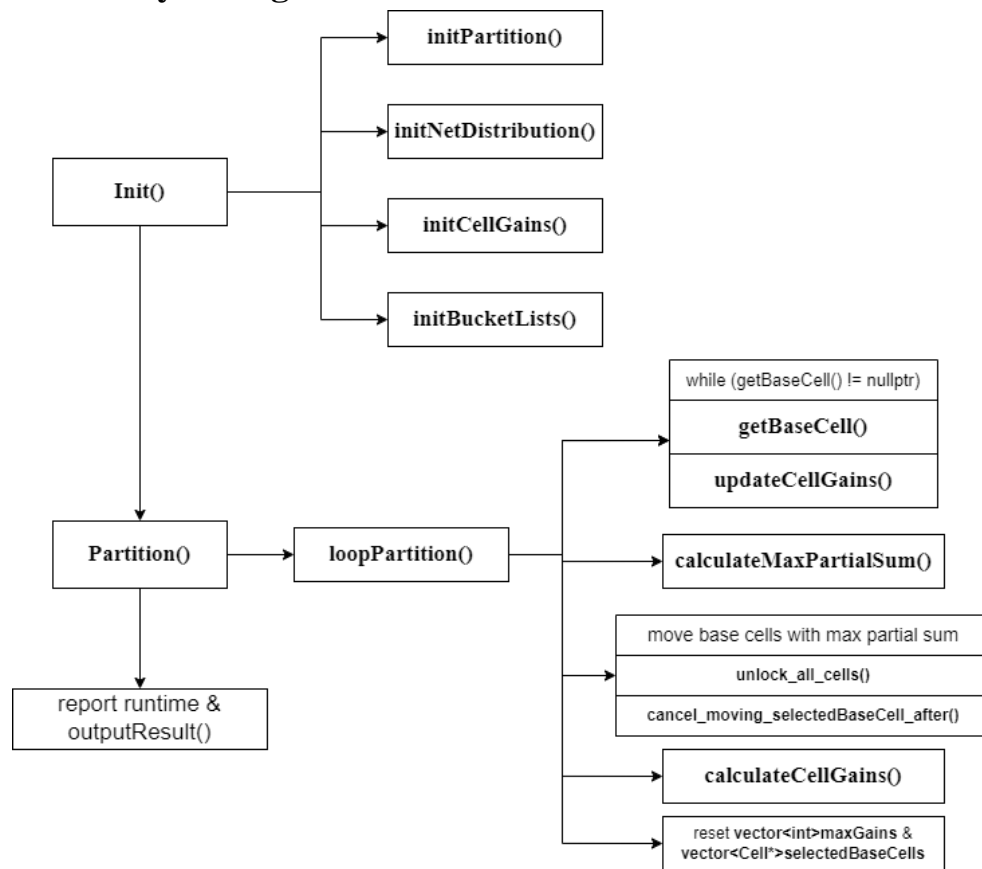
public3: cut size = 63435, runtime = 285.57 secs

public4: cut size = 4427, runtime = 3.60 secs

public5: cut size = 32063, runtime = 285.31 secs

public6: cut size = 328506, runtime = 286.69 secs

#### 4. The details of your algorithm



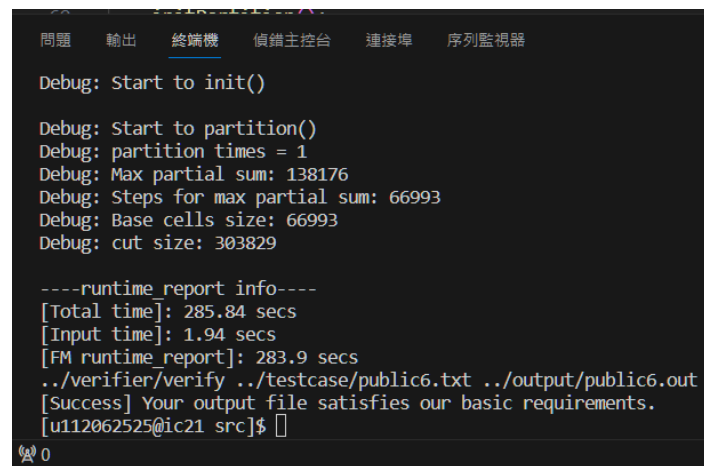
基本上就是以 FM 演算法來完成此次作業，但仍有一些小細節不同。除了在 **initPartition()** 用 random shuffle 分散 cells，還有 **getBaseCell()** 這兩種情況會用到的 **isBalanced(areaA, areaB)** 這個 balance function 改成了這次作業所要求的限制外，最大的改動是以 max partial sum 搬動完對應的 base cell 後，需要對所有 cells 重新計算一次 gain。原本的 FM 只會 update free cells 的 gain，可是沒有連著被搬動的 base cells 一起 update 的話，會導致

下一輪 loopPartition() 尋找 base cell 時因為錯誤資訊而找不到理應找到的 base cell。

Partition() 和 loopPartition() 的差異是在 loopPartition() 裡會用 while loop 遍歷出所有 base cells，並以 maxPartialSum() 得到的 max partial sum 搬動對應數量的 base cells；而 Partition() 則是做完一次 loopPartition() 後，若該次得到的 max partial sum 不小於 0，就重複執行 loopPartition 直到 max partial sum 是負數為止。

## 5. What tricks did you use to enhance your solution quality?

在 FM 執行上，我僅對 runtime 設了上限，確保我的 program 可以在規定時間內產出 .out 檔。另外是在 input 階段時，我讓每個 class 的 object 存取其他 object 的資訊時盡量以 pointer 的形式去做，例如一個 Cell 需要記得和自己連接的 nets 有哪些，我便讓 cell 以 vector<net\*> nets 儲存必要的資訊。由於內部所存的資料為 pointer，不僅在空間上比起儲存整個 object 有優勢，操作時也不需要花費複製整個 object 的時間，可以留更多時間給執行 FM。即使 cell 的數量龐大如 public6，我仍能在 2 秒內建構出所有必要資訊(如下圖)。



```
問題 輸出 終端機 偵錯主控台 連接埠 序列監視器

Debug: Start to init()

Debug: Start to partition()
Debug: partition times = 1
Debug: Max partial sum: 138176
Debug: Steps for max partial sum: 66993
Debug: Base cells size: 66993
Debug: cut size: 303829

----runtime report info----
[Total time]: 285.84 secs
[Input time]: 1.94 secs
[FM runtime report]: 283.9 secs
../verifier/verify ../testcase/public6.txt ../output/public6.out
[Success] Your output file satisfies our basic requirements.
[u112062525@ic21 src]$
```

## 6. Parallelization?

我沒有做任何平行化。

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

(1) 在寫 `initPartition()` 的時候，起初我想將所有 cells 先丟去 DieA，再依照這些 cells 在 DieB 的 size 由大到小丟到 B，直到 `area(B)` 快要逼近 DieB 的 max util 為止。但在運行結果不如預期後，發現在幾個 public 的 testcases 裡會有數十個 cell size 非常大的 cells，若按照這個方法會發生 `area(A) & area(B)` 都超出上限 (`isBalanced(area(A), area(B)) == false`) 的狀況，`initPartition()` 便會失敗。

後來我決定改採 random shuffle 所有 cells 後，一樣先丟進 DieA，直到 DieA 即將到面積上限後再丟去 DieB。而即使採用隨機分配也仍有可能發生上述狀況，因此我設定此 random partition 必須重複執行直到 `isBalanced()` 為 true。

而經過一些研究與觀察後，cut size 的好壞似乎與 initial partition 很有關係，因此我有考慮是否可以先以 random 試出好答案的時候，紀錄該次 random shuffle 的 seed，但後來觀察到每次的差距大概都會落在 10% 內，加上完成這次作業時已相當接近繳交期限，因此作罷。

(2) 實作 `isBalanced()` 時也遇到不少錯誤。主要是數字運算的型態轉換上會遇到一些 precision 或是 overflow 的問題，比如運算時若不小心有 operand 的型態是 int，就有可能產生 overflow；或是比較兩邊面積是否皆低於所設上限時使用 double 的話，遇到 cell size 極端龐大的 cell 就有可能會出現 precision 問題。

(3) 進到 `partition()` 後，因為各 function 要各自 debug 確定沒問題再串接有相當難度，但串接後發生 bug 要怎麼找出問題點並解決也相當麻煩，這邊算是花費我最多力氣的地方。