CS6135 VLSI Physical Design Automation

Homework 2: Two-way Min-cut Partitioning

Student id:110033638 **Name:林哲宇**

I. Compile and execute

File:

header.h

structure.h

structure.cpp

main.cpp

--How to Compile

In this directory, enter the following command:

$ make

It will generate the executable file "hw2" in "HW2/bin/".

If you wnat to remove it, please enter the following command:

$ make clean

--How to Run

In this directory, enter the following command:

$ ../bin/<exe> <net file> <cell file> <output file>

e.g.:

$ ../bin/hw2 ../testcases/p2-1.nets ../testcases/p2-1.cells ../output/p2-1.out

II. Result of each testcase

Intial partition by random

testcase	p2-1	p2-2	p2-3	p2-4	p2-5
Cutsize	38	551	8430	48524	129700
Runtime	0.02	0.2	51.23	295.9	295.91

```
grading on 110033638:
  testcase |     cutsize |     runtime | status
      p2-1 |          38 |        0.02 | success
      p2-2 |         551 |        0.20 | success
      p2-3 |        8430 |       51.23 | success
      p2-4 |       48524 |      295.90 | success
      p2-5 |      129700 |      295.91 | success
```

Put the first n cell in partition A until area balance areaA >= 0.5* total cell area

testcase	p2-1	p2-2	p2-3	p2-4	p2-5
Cutsize	21	217	2147	42075	125771
Runtime	0.01	0.1	2.26	90.74	295.88

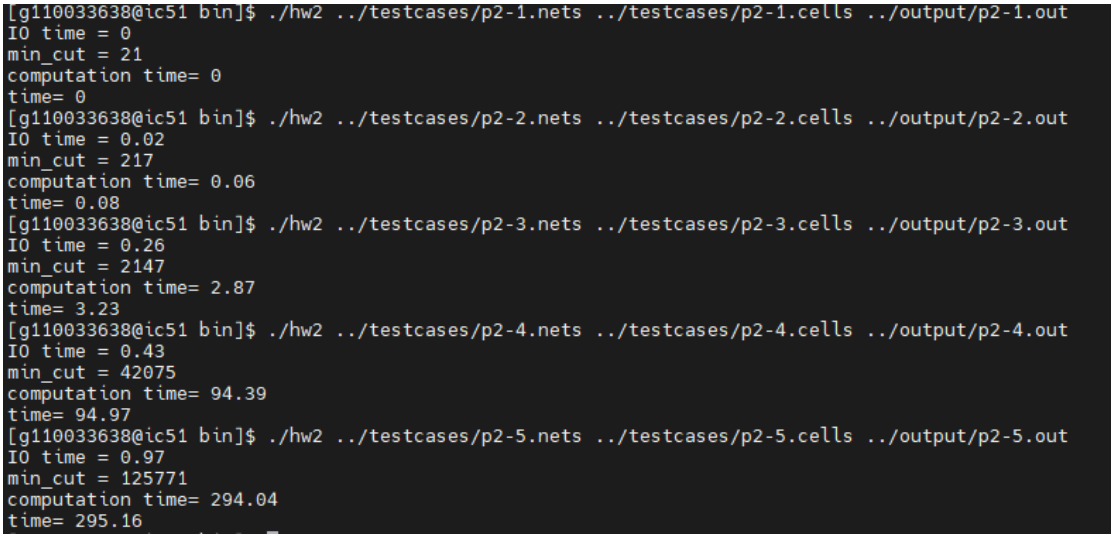
```
grading on 110033638:
find: cannot delete './HW2/.vscode': Directory not empty
  testcase |     cutsize |     runtime | status
        p2-1 |          21 |        0.01 | success
        p2-2 |         217 |        0.10 | success
        p2-3 |        2147 |        2.26 | success
        p2-4 |       42075 |       90.74 | success
        p2-5 |      125771 |      295.88 | success
```

III. Runtime

在 IO time 的部分，包含讀檔、建立 cell array 及 net array，一開始的設計是將讀進來的 cell 直接 push back 到 vector 裡，而未將 cell 進行排序，因此在將 net 讀進來後要建立 cell array 時就必須將所有有的 cell traverse 一次才能找到對應的 cell。之後也有嘗試使用 hash table 去存，但由於計算 hash function 也會消耗許多時間，因此效果不彰。最後設計的版本是先 traverse cell file 一次，找到最大的 cell index，建立一個大小為 max cell index 的 vector，再 traverse 一次，將 cell 直接放入對應的位置，因此在建立 cell array 時，就可直接找到該 cell 的位置，雖然會讀 cell file 兩次，但在速度上來看還是比較快的。

在 Computation time 的部分，testcase p2-5，跑到設定的最大執行時間 295 秒後跳出迴圈，預留 5 秒鐘輸出 output file，而其他 case 均可以在 300 秒內執行完畢。

testcase	p2-1	p2-2	p2-3	p2-4	p2-5
IO time	0.004	0.02	0.26	0.43	0.97
Computation time	0.001	0.06	2.87	94.39	294.04
Runtime	0.006	0.08	3.23	94.97	295.16



IV. Difference

基本上的功能與 FM algorithm 近乎相同，差異僅在沒有使用 partial sum 的部分，因為時做過後發現選擇 gain<0 的 cell 幾乎對最後成績沒有影響，並且會浪費許多時間，因此改成選完 gain>=0 的 cell 後就將 cell lock 都 free 掉，回到當前 minimum cut size stste，並重新做一次。

V. Bucket list data structure

有使用 bucket list，並且有實作 doubly linked list，原先認為使用 doubly linked list 與速度無關，因此 bucket list 中 cell 是使用 vector 去存的，而解果來看在 teastcases p2-1 and p2-2 速度上無明顯差異，然而在 p2-3 p2-4 p2-5 上，會發現在搬動 bucket list 的 cell 會花費過長的時間，導致整個程式效率下降，在時間 5 分鐘內，搬動次數很少始得結果相當差。

改成 doubly linked list 後，由於插入及移除的動作只對 cell 的 prev 及 next 的 pointer 進行操作，並不會使用到記憶體的搬移，因此速度會快相當多，實測下來至少有 10 倍以上的速度成長。

VI. Solution quality and speed up strategy

在 improve solution quality 的部分，由於選 gain<0 的 cell，發現對 cutsize 只會一直增加，對 quality 並無太大幫助，且會耗費許多時間，因此改成選完 gain>=0 的 cell 後就回復到目前 minimum cut size 的狀態後將全部 cell 的 lockstate free 掉，從新再做一次，總共做 4 次。

而在 speed up 的部分，利用 early stop 來進行加速，因為在實測後發現在 bucket list 裡 gain=0 的 cell 數量相當多，並且搬動 gain=0 對 cutsize 的效益不大，因此如果只搬動完 gain>=1 就停，速度會快許多，實測後發現 p2-1 cutsize 70 左右，與 cutsize 20 結果差許多，而 p2-5 平均落在 130000 左右，與 cutsize 125771 不會差太多，但 runtime 卻可以從 300 秒減少 10 秒。

另外，也有發現在我程式裡，花費許多時間在更新 best result 的部分，因為是 cutsize 有可能在搬動後，反而變大，因此要隨時儲存最好的 cutsize 及 partitionA,B 的 cell，因為每次儲存都要 trasverse 整個 cell list，這會造成花費過多的時間，因此設定在選擇的 base cell 的 gain<=0 時才進行更新，以減少 trasverse 整個 cell 的次數。

VII. Parallelization

本次作業並無使用平行化

VIII. Compare

在 initial partition 時用 random 的方式分與前 5 名比較後，quality 都沒有比前五名好，尤其在 testcase p2-3 的差距相當大，測試許多次後仍然無法改善，但在 p2-4 及 p2-5 的差距卻小許多，並且在 p2-1 多次 random 後有找到最小的 cutsize 14。

如果 initial partition 先將 cell 放入 partitionA 直到 areaA>=total cell size，然後剩下的 cell 都放入 partitionB，結果會好非常多， runtime 也比 random 的快，並且在 p2-4 的 cutsize 是比前 5 名的成績還要好的，但在 Runtime 上與前五名相比速度仍有些差距。

因此可以發現，FM algorithm 在 initial partition 是非常重要的，大機率會影響最後的結果，如果 initial partition 沒有分好，是有機會永遠都無法將 cutsize 壓下來的。因此之後可以在 initial partition 進行改進。

Result:

testcase	p2-1	p2-2	p2-3	p2-4	p2-5
Cutsize	14(random)	217	2147	42075	125771
Runtime	0.01	0.1	2.26	90.74	295.88

IX. Learn from homework

這次作業花費許多心力在完成，包含熟悉 C++語法以及如何設計資料結構，並且大部分的時間都在處理速度上的問題，測資數量龐大，因此要避免 trasverse 整個 cell，在嘗試各種資料結構後，最後使用 vector 及 doubly linked list，雖然最後速度上有改進許多，但是與前五名相比速度仍有些差距，並且發現利用 random 進行 initail partition 是非常不穩定的，最後的 cutsize 的變動會非常多，必須要經過多次實驗才能取的最好的解。

而這次作業中，收穫良多，深入了解了 FM algorithm 的基本操作及原理，並且也學到了如何利用 C++實現 doubly linked list 以及習慣使用 pointer 進行資料的操作。另外也學到如何利用 makefile 來編譯程式。