CS6135 VLSI Physical Design Automation

Homework 2: Two-way Min-cut Partitioning

107062536 李岳容

1. How to compile and execute your program, and give an execution example. If you implement parallelization, please let me know how to execute it with single thread.

   Step 1 : 進入個人資料夾，存放主要程式的地方

   command line: $cd/HW2/src

   Step 2 : 使用 makefile compile 主程式檔

   command line: $make

   Step 3 : 執行執行檔

   command line: $ ./main ../testcases/p2-1.cells ../testcases/p2-1.nets ../testcases/p2-1.out

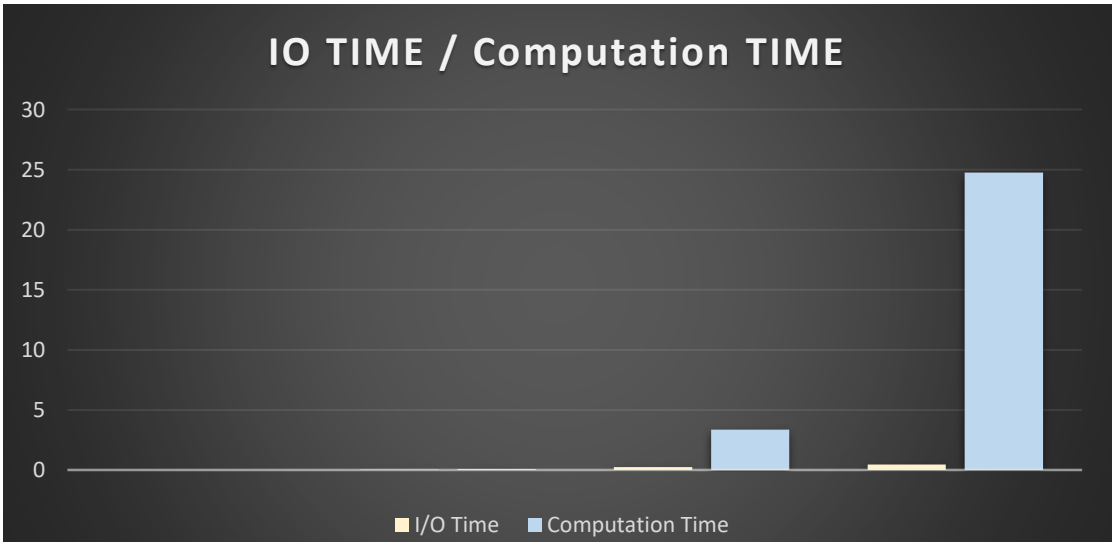
   (以上 p2-1.cells/p2-1.nets 檔案可以替換成其他測資)

2. The final cut size and the runtime of each testcase

runtime				cutsize			
2-1	2-2	2-3	2-4	2-1	2-2	2-3	2-4
0.01	0.10	3.65	25.25	16	332	2216	45646

3. Runtime = $TIO + Tcomputation$. For each case, please analyze your runtime and find out how much time you spent on I/O and how much time you spent on the computation (FM Algorithm).

IO time				Computation Time			
2-1	2-2	2-3	2-4	2-1	2-2	2-3	2-4
0.00	0.02	0.25	0.47	0.00	0.08	3.37	24.75



4. The details of your implementation containing explanations of the following questions:

I. Where is the difference between your algorithm and FM Algorithm described in class? Are they exactly the same?

Yes.我實作的方法跟講義上的幾乎一樣，只差別在一輪移動 cells 時，會真的更動他的位置。當一輪結束後，我會選取最好的 partial sum 把其之後，多做的移動更改回來。

II. Did you implement the bucket list data structure?

Yes.我實作的方法跟講義上的幾乎一樣。一整個 Bucket List 大小是，所有 Cells 當中連接最多 Pin 的數量的兩倍加一。

只差別在 Bucket List 本身，一格是存放一個 Cell 的指標，而這些 Cell 指標是真實指向一開始創建的 Cells，而 Bucket List 要將 Cells 連接起來，也是讓 Cells Structure 並內部有往前指與往後指的指標，直接指向在 Bucket List 前後的 cell。

III. How did you find the maximum partial sum and restore the result?

```
MoveToB(ctmp);
changed_cells.push_back(ctmp);
CurrentPartial+=(*ctmp).gain;
if(CurrentPartial > MaxPartial){
    MaxPartial = CurrentPartial;
    target_step = move_step;
}++move_step;
```

在一輪當中，透過移動 Cells，每一次都可以得到移動的那個 cell，其原先帶有的 gain 值。而一整輪下來會移動很多 cells，因此一輪的開始必須初始一個值為 0，在本程式裡為 CurrentPartitial，去記錄這些得到的 gain 的加總。並在途中，使用 MaxPartitial 去紀錄目前為止最大的 CurrentPartital。並且更新這個最大值的出現，是在哪一步(target_step)。而 target_step 代表著，一整輪會有很多 cells 被交換到另一個區域去，在 target_step 這步可以得到目前最好的 cutsize，也就是這步下 cells 分布在 A/B 區域的狀況，可以獲得目前最小的 cutsize。因此下一輪開始，我們必須回復到當時 Cells 在 A/B 的分布情況。而本程式的計算方法是每次移動一個 cells，會真的將他的所屬區域換到另一區，同時把這個 cell 的指標 Push 到 Changed_cells(vector)裡。在一整輪結束時，為了保持 target_step 當下 cells 在 A/B 區域的分布狀況。target_step 之後都是不要的交換，因此要再次把它回復成原始所屬的 region。

IV. Please compare your results with the top 5 students' results from last year and show your advantage either in runtime or in solution quality. Are your results better than them?

Top 5 of last year!!!

rank	2-1 runtime	p2-2 runtime	p2-3 runtime	p2-4 runtime	p2-1 cutsize	p2-2 cutsize	p2-3 cutsize	p2-4 cutsize
1	0.001	0.014	1.058	10.982	8	197	1275	46719
2	0.006	0.29	431.588	1090.998	6	255	2633	42135
3	0.015	0.423	29.066	772.805	6	511	1564	45419
4	0.041	1.307	679.534	1471.833	6	221	1630	46323
5	0.002	0.04	0.97	35.163	23	225	9398	46432

runtime				cutsize			
2-1	2-2	2-3	2-4	2-1	2-2	2-3	2-4
0.01	0.10	3.65	25.25	16	332	2216	45646

Advantages : 在時間上，我幾乎贏過大部分的 run-time。
Weakness : 在 cutsize 的優化上，我並沒有特別的顯著。我想我的 bottle net could do to improve the result in the future?

V. What else did you do to enhance your solution quality or to speed up your program?
Speed:

我特別將 Bucket List 裡的做法，改成直接在 Cells 的 structure 裡新增前後指標，去代表這個 Cell 目前在 Bucket List 裡的位置。這個做法可以讓更動 Bucket List 變得很快速。

為了讓 Net 在找內部連接到的 Cells 時(反向亦然)，可以快速找到需要的 Cells，我是儲存 cell 的 id 在 Net 裡，當需要查找 cells 時就使用 id 去查找存放 cells 的 map(以 id 為 key)，而這個 Map 是 unordered map 其差找時間為 O(1)

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

指標能隨便指，會爆掉。最好在編寫時，就要寫一個小型的測資，測試自己的指標會不會指錯。