CS6135 VLSI Physical Design Automation

Homework 2: Two-way Min-cut Paritioning

112062590 劉俊圻

1. How to compile and execute program, and give an execution example.

(1) Compile

進入 src/目錄，並且輸入 make 即可完成所有檔案的編譯，並在 bin/目錄底下產生 hw2 執行檔

e.g.

$ cd src

$ make

(2) Execute

直接使用 command 找到執行檔和相對應的輸入 testcase 來執行程式

e.g.

(in src directory)

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

$ ../bin/hw2 ../testcase/public2.txt ../output/public2.out

…

$ ../bin/hw2 ../testcase/public6.txt ../output/public6.out

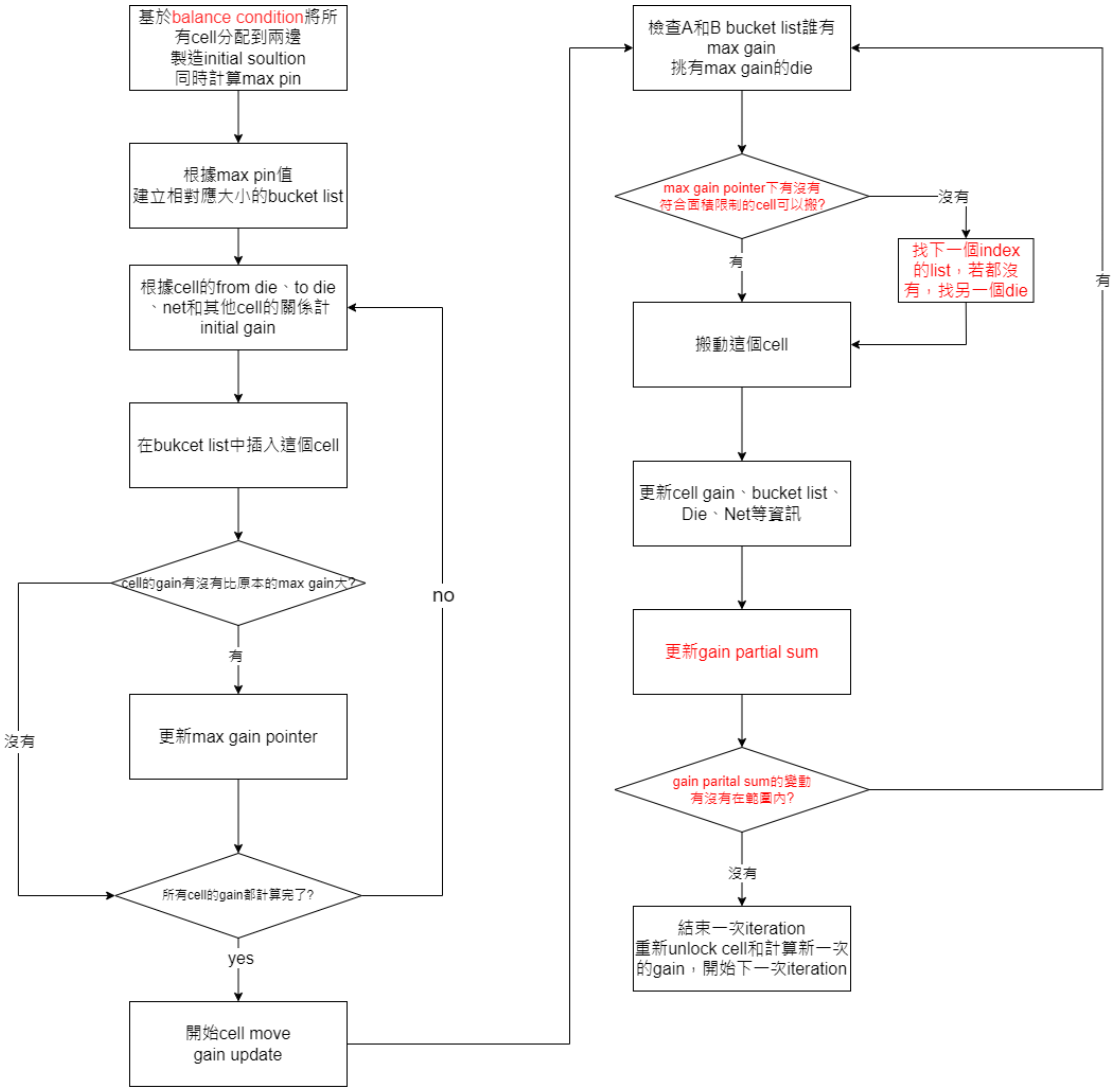
2. The final cut size and the runtime of each testcase

Test case	Final cut size	Runtime
public1.txt	339	0.09
public2.txt	3118	45.3
public3.txt	51201	290.43
public4.txt	2212	2.29
public5.txt	3055	99.75
public6.txt	328100	291.68

3. The details of algorithm.

- Fiduccia and Mattheyses. “A linear time heuristic for improving network partitions,” 19th Design Automation Conf.,1982.

  我所採用的方法是上課時所介紹的 FM 演算法，但在實現 FM 的過程我有進行不同的調整



紅色的步驟是我和原演算法不一樣之處，以下是我對這些不同處和其他細節的實作

(1) Initial solution 的 balance condition

原 FM 演算法在生成 initial solution 時應該要符合 $\frac{|A|}{|A|+|B|} \cong r$ 的限制，但我的做法是先直接將 cell 優先分配給 DieA，先檢查分配過後是否符合 spec 中的面積使用率，若分配後不符條件，則分配給 DieB，否則 initial solution 不合法，後續的結果也難以生產合法的解

(2) max gain Pointer 底下是否有符合面積使用率的 cell?

原 FM 演算法中，必須找到符合(1)中的算是的 cell 才能搬動，但我的作法是只要能符合 spec 的使用率限制就好，並沒有特別判斷是否有符合原 FM 中所謂的平衡條件，而是優先以有 max gain 的 cell 為挑選對象

(3) 找下一個 index 的 list (若是 max gains index 沒有符合面積使用率的 cell)

原 FM 演算法中並沒有特別提及若是 max gain pointer 指著的 list 中都沒有符合條件的

cell 該如何挑選下個對象，我的作法是從同一個 Die 中的下一個 index 的 list 開始尋找。若是其他的 list 也都找不到符合條件的 cell，就找另一個 Die 的 max gain pointer

(4) max partial sum 和 Die 底下的 cell 紀錄

原 FM 演算法中會盡量將每一個 cell 都移動過，並且在過程中紀錄這些 cell 的 gain 和 Die 底下有哪些 cell 的狀態，並在最後決定 gain max partial sum 和恢復對應到的狀態，我在實作這個 Die 和 cell 資訊備份的過程中發現 runtime 會變得非常長，甚至在 public2 的 testcase 下，一個 iteration 就會來到將近 5 分鐘的時間，這方法顯然是行不通。

因此我改成每一次搬動 cell 之後就計算新的 partial sum，若是這個 partial sum 的結果比之前的 max partial sum 小於一定比例(max partial sum * 0.9 > new partial sum)，就終止這次的 iteration，並且用前一次 iteration 的狀態作為下一次的 initial solution，unlock 所有 cell 後重新計算 gain 和 bucket list。雖然這樣會錯過最佳的狀態，但 runtime 變得較為合理。

(5) 時間限制

原 FM 演算法應該要將所有的 cell 搬動後計算 max partial sum，回復至 max partial sum 的狀態，重複執行直到收斂到 gain 為 0 或以下，但我的程式在遇到大量的輸入時無法全部完成，甚至在採用(4)的方法減少 runtime 後在某些 test case 仍無法達到時間限制，因此我在程式中限制了程式執行時間上限。

4. Different method and result comparison

在確定了演算法的實作之後，我試了以下兩個不同配置的交叉組合來嘗試找到最小的 cut size

(1) 是否限制每一輪的 iteration 時間

在這組實驗中，我試了兩種不同方法，分別為限制每一輪 iteration 的時間以及不限制每一輪 iteration 時間，在前者，我限制了每一次的 move cell、得到 gain、重新 unlock cell、更新 cell gain…這些 statement 在每一輪能執行多久，若是超出時間限制，gain partial sum 還沒收斂，也會直接開始下一次的 iteration，並且在總執行時間限制內完成越多次 iteration 越好；另一個方案則是代表一次的 iteration 執行多久都沒關係，只要還沒像前面所提及的 partial sum 開始下降到一定程度，程式就繼續執行 move cell 和計算 gain partial sum，直到執行時間到限制，再輸出結果

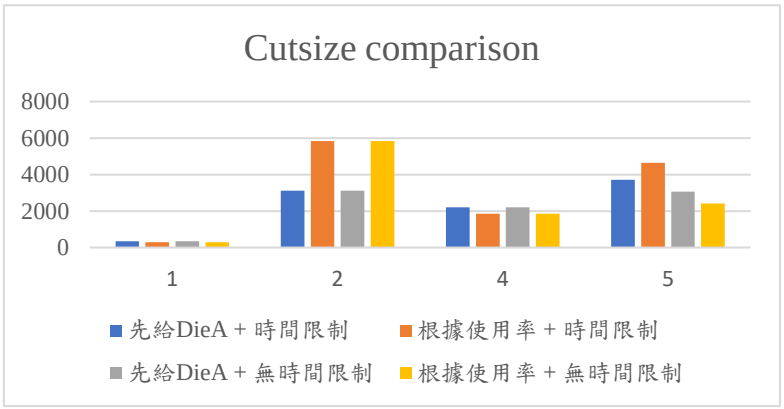
(2) initial solution 的生產條件

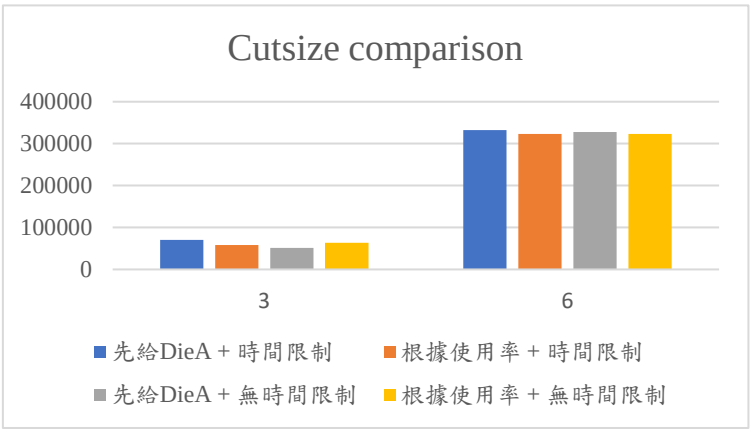
在這組實驗中，我試了兩種不同的 cell 的在兩個 die 上的分配方法來生成 initial solution，第一個是優先將 cell 分配給 DieA，若是 DieA 塞不下，則將這個 cell 分配給 DieB；另一個則是先檢查 DieA 和 DieB 的面積使用率，面積使用率較小且能將 cell 放入的 die 優先分配到 cell

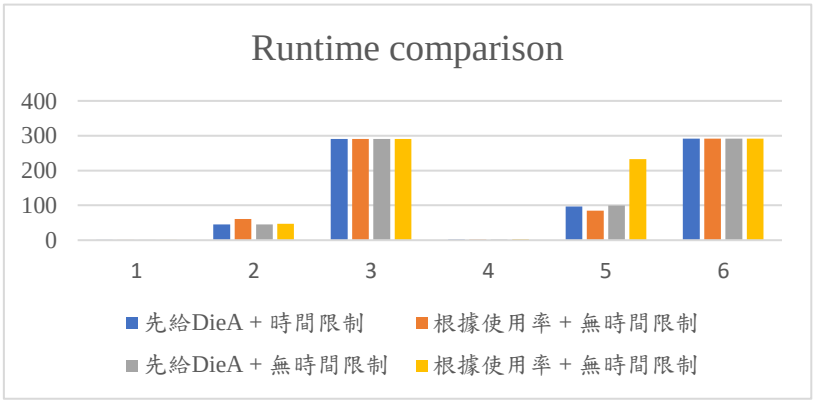
(3) 實驗結果

可以看到在 cutsize 較小的 test case(1, 2, 4, 5)中，優先給 DieA 的 initial solution 生成方法有較優的表現，先給 DieA 配合 iteration 無時間限制方案則略優於先給 DieA 搭配時間 iteration 限制方案的表現，但在 cutsize 較大的問題中，四種方法的結果都差不多，因此最後我挑選的方法為先給 DieA＋ 無時間限制的方法

在 runtime 方面這幾個方法都沒有太大的差別，除了在根據使用率+時間限制的情況下，public5 的 text case 會突然時間暴增，但也在我設的執行時間限制內，其他的無法在 5 分鐘內跑完的 test case 則是被我設定的時間規範，所以還是以 cutsize 結果作為方法的挑選依據







5. Parallelization

我並沒有實作平行化

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

在這份作業中我遇到各式各樣的困難，從資料結構的發想、STL 的挑選、initial solution 的生成方法、上面提到的 max gain 下的 cell 不符合條件的話該怎麼辦的機制…尤其是在資料結構的發想上，Lib Cell、Cell、Tech、Net、Die 的關係實在錯綜複雜，這也讓後續的程式開發遇到困難，發生上面提到的備份太耗時的問題，我只能選擇一個折衷的方案。但也是因為這樣讓我學會在寫程式實進行縝密的思考，到底什麼資料結構和迴圈的執行是必要的或可以簡化的，除了學習到 EDA 相關的演算法之外，更讓我熟悉 C++開發的一些原理和慣用法，希望這次的開發經驗能讓後續的作業越來越順利。