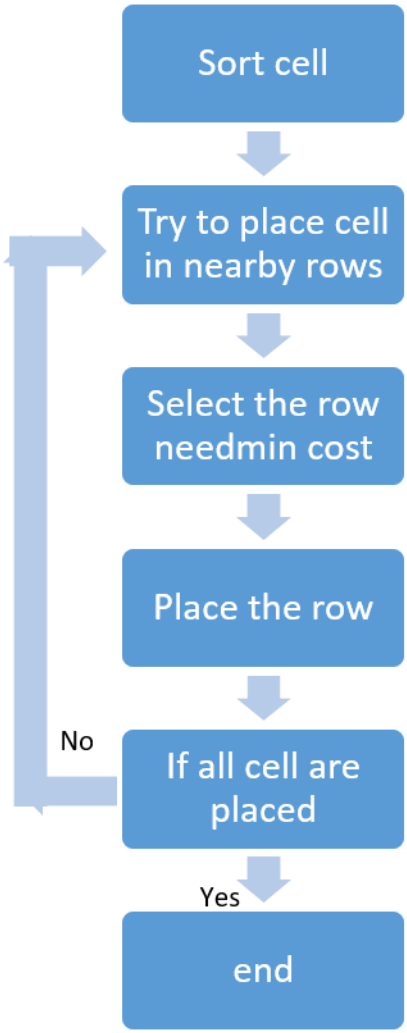
HW3 report

NTUEE

B03901152

陳景由

1.演算法流程 (Algorithm Flow) (6pt)



配合流程圖，將程式運作流程再加上文字敘述如下：

使用 abucus 演算法。

1. 將每一個 cell 嘗試著放到鄰近的 row 中，如果會和前面先放好的 cell 產生 overlap，就把他們全部當成一個 cluster，求出 local optimized 位置並放置。
2. 嘗試直到垂直距離變化的 cost 比前面嘗試過的 min cost 都還要大時，就不再放置，並將 cell 放在產生 min cost 的 row 中。
3. 重複 1.2 直到結束。

2.資料結構 (Data Structure) (6pt)

- Abcabus，一開始會將 CLegal 傳進去，之後所有有關的演算法都在這個 class 的 member function 裡面實作，不會影響到外面的資料結構。
  - _clusters: 儲存所有的 clusters。
  - _remain_widths: 每個 row 一開始初始的 remian_width = width，每塞入一個 cell，remain_widtg 都會減去 cell 的 width，如果該 row 要塞入一個新的 cell，但是 remain_width 不足就會無法塞入。
  - _ordered_moduleIDs: 一開始會將所有 cell sort 過，而 sort 則是根據 x-position+width/2。
  - _forward_clusters_sol: 會儲存順著從左方 legalize 過來的結果，做完順方向之後會作反方向的 legalize，並且判斷哪個方向的 cost 將會比較低，最後採取較低的排列當作答案。

```
class Abucus
{
private:
    vector<Cluster>             _save_clusters;
    double                      _save_remain_width;


    CLegal&                     _cLegal;
    Placement&                  _placement;
    vector<unsigned>            _ordered_moduleIDs;
    vector<double>              _remain_widths;
    vector<vector<Cluster>>     _clusters;
    vector<vector<Cluster>>     _forward_clusters_sol;
    double                      _forward_displacement;
    vector<CPoint>              _ordered_sol;
    double                      _boundaryRight;
};
```

- Cluster: 儲存 abucus 需要的相關資訊，記錄這個 cluster 帶有那些 cell。

```
class Cluster {
public:
    Cluster(){}
    Cluster(unsigned ID):
    _ID(ID),_w(0),_e(0), _q(0) {}
    unsigned _ID;
    double _w;
    double _e; // size of _modules
    double _x;
```

```
    double _q;
    vector<unsigned> _moduleIDs;
};
```

3. 問題與討論 (8pt)

1. Cost function:

Cost function 影響結果甚鉅，原本是只有考慮要排進去 cell 的 square displacement = (delta x)^2 + (delta y)^2。但是後來覺得這樣並沒有考慮到其他 Cell 被影響到的情況，因為如果在排進去的時候有 overlap，會產生新的 Cluster，其他 cell 也會被改變位置，因此後來決定使用 cost function = square displacement of 所有被影響的 cell + square displacement of new inserted cell * alpha。其中 alpha 是一個 user define 的 parameter，慢慢去調整它結果會有所變化，在本次作業中使用的 alpha = 1。

2. Sort by x coordinate:

有其他份論文說可以用 x + width/2 作 sort，結果會比接用 x coordinate sort 好上 3%~5%，影響到的結果就是當兩個 cellx coordinate 差不多的時候會選擇短的 cell 先作排列。

3. 順序:

從左到右和從又到左也會有所差異，也有約 3%的差異，但是實作上的困難來蠻不小的，幾乎是把所有的 function 重寫一次，我的方法把 width 變成負的，然後從右邊排起，每個 cell 右下角定成他的 global position。

4. 速度:

用 abacus 的速度非常快，用中位數去求解的方式會更快，如果之後要做 legal 應該可以使用其他結果更好但是會比較花時間的方式。