CS6135 VLSI 實體設計自動化 Homework2

108062636 王勁程

如何編譯及執行

1. 編譯：

   將目錄切換至/HW2/src，輸入指令$ make，將會在/HW2/bin 資料夾產生一個可執行檔“FM_Partitioner”。

   若需移除請在/HW2/src 輸入指令$ make clean 即可。

2. 執行：

   將目錄切換至/HW2/bin 使用以下指令：

   $ ./FM_Partitioner -c <.cells 檔案> -n <.nets 檔案> -o <輸出結果檔案名稱>

   例如：

   $ ./FM_Partitioner -c ../testcases/p2-1.cells -n ../testcases/p2-1.nets -o p2-1.out

   執行結果 p2-1.out 將會生成在/HW2/output。

3. 自動編譯及執行：

   在"HW2"目錄下使用以下指令

   $ ./run.sh

   將自動編譯並讀取/HW2/testcases/目錄下 p2-1~p2-4(.cells / .nets)測資執行程式

   執行結果將輸出至../output 目錄

執行結果

	P2-1	P2-2	P2-3	P2-4
Cut size	60	1185	26319	66400
輸入/輸出時間	0.00	0.00	0.15	0.32
計算時間	0.00	0.29	218.93	610.80
執行時間(秒)	0.00	0.29	219.08	611.12

虛擬碼

輸入：

1. .cells 檔案

   每一行皆包含” cell 編號 cell 大小”

2. .nets 檔案

   NET n{net 編號) {在此 net 上的 cell 集合}

輸出：

cut_size #

A #Cell_in_groupA cell_ID

…

B #Cell_in_groupB cell_ID

Begin

Step1: initial partition

Step2: compute gains of all cells

Step3:

i=1 Select “base cell” and call it ci, If no base cell then exit.

A base cell is one which (i) has maximum gain; (ii) satisfies balance criterion;

Balance criterion:

$|area(A)-area(B)|<n/10$, where area(A) is the sum of all cell sizes in A, area(B) is the sum of all cell sizes in B, and n is the sum of all cell sizes in the circuit.

Step 4: lock ci and update gain

Step 5: If free cells≠null Then i=i+1; select next base cell; If ci ≠ null then Goto step 4;

Step 6: Output the result with minimum cutsize

End

問題

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

   是的，大致上都照著講義做。

2. Did you implement the bucket list data structure?

   是的，我使用 C++ STL vector 來實作兩個 bucket list，之所以會使用是因為他在找最大的 gain 時非常的快，而我的資料結構如下：

   vector<int> *bucket[2];

   bucket[0]= new vector<int>[pmax-pmin+1];

   bucket[1]= new vector<int>[pmax-pmin+1];

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

   我在每一次搬動 base cell 後皆有計算其 cut size 並跟歷史紀錄中最好的比較，如果比它更好，則更新最好的 cut size 存入變數 opt_cutsize，同時將對應的 cell 的分區資訊存入兩條 list 中。有了最小 cut size，就可以用初始 cut size 減掉 opt_cutsize 就是我的 maximum partial sum。

```
int c = compute_cutsize();
if(opt_cutsize > c)
{
    opt_cutsize = c;
    opt_block[0].clear();
    opt_block[1].clear();
    opt_num[0] = opt_num[1] = 0;
    for(int i = 1; i <= max_cell_num; i++)
    {
        if(cells[i].number != 0)
        {
            opt_block[cells[i].block].push_back(i);
            opt_num[cells[i].block]++;
        }
    }
}
```

4. 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?

比他們糟糕太多了！我覺得應該是在資料結構方面差異及平行化的使用與他們的差異造成執行時間的差異，又或者是我在做搬動的演算法做得並不好。而在 cut size 的部分，我認為應該也與一開始產生的初始分割有關，而在我的程式裡面是使用隨機分配的方式來進行初始分割。

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

   在我的程式裡面是使用隨機分配的方式來進行初始分割，目的就是為了快速，但似乎應該改善搬動的演算法會更有效。

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

   學到了 bucket list 這個資料結構以及 FM 演算法的整個流程也更加了解。

   在寫完程式準備執行時，我最常遭遇到的問題就是 segment fault，我去查了許多關於這個問題的發生原因，甚至有時候在我自己的電腦可以執行，放到工作站卻出現 segment fault。這個問題部會告訴我程式碼哪行出錯，所以設定了許多中斷點才找出問題點。

   再來還有 string 跟 char*之間的差異，一開始我在讀檔就出現這個問題，後來查了一些資料終於搞懂如何他們的原理。

   我還學會了 makefile 的相關語法以及 command line 的命令解析。

7. If you implement parallelization, please describe the implementation details and provide some experimental results.

   沒有做平行化。