CS6135 VLSI Physical Design Automation

Homework 3 Report: Fixed-outline Slicing Floorplan Design

108062605 呂宸漢

1. 如何編譯及執行程式：

   進入 HW3/src/後，輸入 make 可編譯程式並輸出執行檔 hw3 至 HW3/bin/。

   如：$ make

   之後進入 HW3/bin/，輸入./hw3 <hardblock file> <net file> <pl file> <floorplan file> <deadspace ratio>即可執行程式。

   如：$ ./hw3 ../testcase/n100.hardblocks ../testcase/n100.nets ../testcase/n100.pl ../output/n100.floorplan 0.2

   注意：hardblock file 的副檔名須為.hardblocks、net file 的副檔名須為.nets、pl file 的副檔名須為.pl、floorplan file 的副檔名須為.floorplan。

2. wirelength and runtime for each case：

deadspace ratio: 0.2			
	n100	n200	n300
wirelength	201266	360086	495720
I/O time	0.004316	0.008714	0.008958
SA time for area	0.938868	2.92551	17.3382
SA time for WL	13.1677	64.9683	187.482
runtime	14.1109	67.9026	204.83

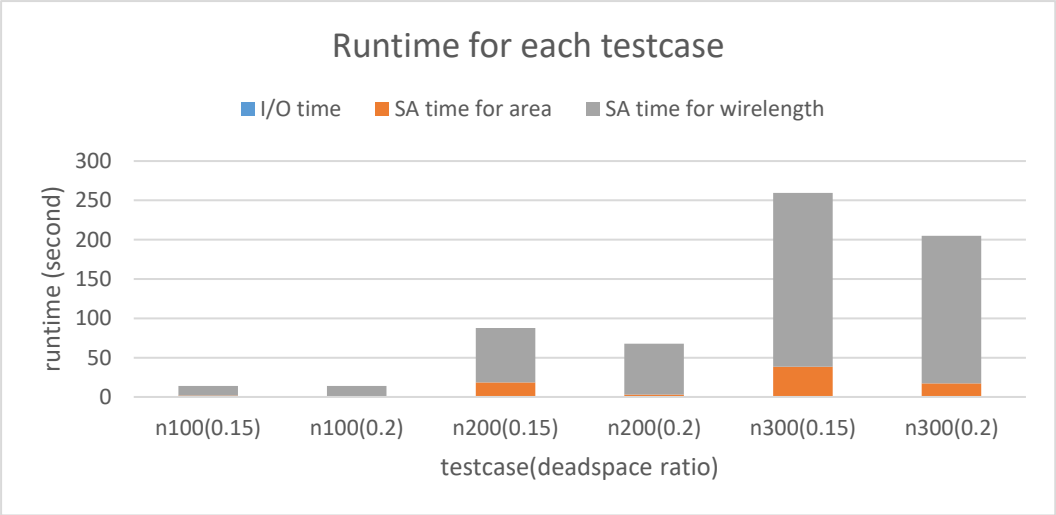
(時間以 clock_gettime 的 CLOCK_MONOTONIC 測量)

Table 1

deadspace ratio: 0.15			
	n100	n200	n300
wirelength	207309	367785	504903
I/O time	0.004352	0.007572	0.00921
SA time for area	1.59035	18.2755	38.3977
SA time for WL	12.5197	69.2965	220.901
runtime	14.1144	87.5795	259.308

(時間以 clock_gettime 的 CLOCK_MONOTONIC 測量)

Table 2



因為在 I/O 時會先讀取 hardblock file 將每個 hardblock 面積算出來並加總，在 record 內依照 roatation 前後的寬高做排序，並額外建一個 pin 記錄 block 中心點；再讀取 pl file 讀取每個 terminal 的位置並建立一個 pin 儲存；最後讀取 net file 記錄每個 net 有接到哪些 pin，以便之後方便計算 wirelength。

I/O time: *O(B) + O(P) + N*O(B+P) + O(B) = O(N*(B+P)) (B: number of hardblock, N: number of net, P: number of terminal pin)*

因為讀取 hardblock file 要 *O(B)*，讀取 pl file 要 *O(P)*，讀取 net file 並儲存 pin list 要 *N*O(B+P)*，寫入 floorplan file 要 *O(B)*。

Computation: 即為 SA 的時間複雜度的兩倍

因為第一個 SA 是對 area 做優化，第二個 SA 是對 wirelength 做優化。

3. the smallest deadspace ratio I find in 20 minutes：

	n100	n200	n300
deadspace ratio	0.09	0.07	0.08
wirelength	295838	551755	828910
I/O time	0.003857	0.007498	0.009355
SA time for area	88.1576	191.186	694.58
SA time for WL	10.4612	58.3335	185.612
runtime	98.6226	249.527	880.201

(時間以 clock_gettime 的 CLOCK_MONOTONIC 測量)

Table 3

由 Table 1、Table 2 及 Table 3 可以看出，當 deadspace ratio 越小，SA 花費在 area 上的時間越長，反而 SA 花費在 wirelength 上的時間變短了，我認為是因為 deadspace ratio 越小，可以擺進去的方式越少，導致 feasible 的答案不好找，且能移動並優化 wirelength 的 hardblock 也越少，才會有這種現象。

4. the detail of my implementation：

以下列出我的程式與課程 SA 的不同處

a. initial solution 不同：

課程 SA 的 initial solution 是 12V3V…nV (n: number of hardblock)，可是在實際執行程式後發現，因為 initial solution 超出 outline 的很多，需要較長的時間才能找到可以完全放入的方式。因此我改用 width 計算 12V3V…的每列是否超出水平方向的 outline，如果超過就換行繼續擺放，這種方式可以盡量讓 initial solution 放進 outline 內，也較接近 feasible solution。

b. 在 SA for area 時 cost function 只考慮 area，在 SA for wirelength 時 cost function 再考慮 area+wirelength：

課程 SA 的 cost function 是直接考慮 area+wirelength，可是這樣即使在擺不進 outline 的時候也要花時間計算 wirelength，比較浪費時間，因此我把它改成兩的 part，一開始不考慮 wirelength，先找出可以擺進 outline 的方式，確定擺得進去後再優化 wirelength，雖然一開始有考慮兩者的 cost function 可能可以較快找到答案，可是要再優化 wirelength 會較花時間，尤其當根本塞不進去時更是，因此我才會採用 area 與 wirelength 分開的 cost function。

以下呈現不同 initial solution 在 n100 case deadspace ratio=0.2 的執行結果

	課程	my
wirelength	306436	207309
I/O time	0.00439	0.004352
SA time for area	150.485	1.59035
SA time for WL	10.7789	12.5197
runtime	161.268	14.1144

Table 4

5. compare with top 5 student's result：

雖然不應該分開來比較 wirelength 與 runtime，不過因為不知道 wirelength 及 runtime 的排名計算方式，我先分開來分析 wirelength 及 runtime，最後再綜合評分。

deadspace ratio: 0.15

wirelength				
	n100	n200	n300	積分
my	207309 (2)	367785 (1)	504903 (1)	4
1	209584 (3)	375829 (2)	522036 (2)	7
2	214282 (5)	395941 (5)	556247 (5)	15
3	210292 (4)	377553 (3)	545774 (3)	10
4	218352 (6)	402538 (6)	553960 (4)	16
5	204470 (1)	378612 (4)	562069 (6)	11

Table 5

Table 5 為 wirelength 的排名，雖然我在 n100 的 case 不是最短的，可是我在生下的 case 都是最短的，尤其是 n300 的 case 比其他結果短了至少 20000，以所有數據而言，我認為我贏過第一名。

deadspace ratio: 0.15

runtime				
	n100	n200	n300	積分
1	35.877 (5)	72.306 (3)	105.001 (2)	10
2	2.142 (1)	12.651 (1)	32.083 (1)	3
3	21.121 (3)	71.424 (2)	182.017 (3)	8
my	14.114 (2)	87.580 (4)	259.308 (6)	12
4	25.017 (4)	100.144 (5)	225.460 (5)	14
5	87.674 (6)	159.531 (6)	220.383 (4)	16

(時間以 clock_gettime 的 CLOCK_MONOTONIC 測量)

Table 6

Table 6 為 runtime 的排名，雖然我在 n200 及 n300 的 case 表現不佳，可是我在 n100 的 case 的排名第二，以所有數據而言，並用積分計算，我認為我贏過第四名。

綜合上述結果，就總積分而言，我認為我應該是第一名。雖然我的 runtime 在 n200 及 n300 的 case 表現不佳，可是 wirelength 皆為最短，由 Table 2 可見，其實我最多在 40 秒內就可以擺進 outline 內，只是在優化 wirelengh 的部分耗費了較多的時間，不過也得到了不錯的 wirelength。

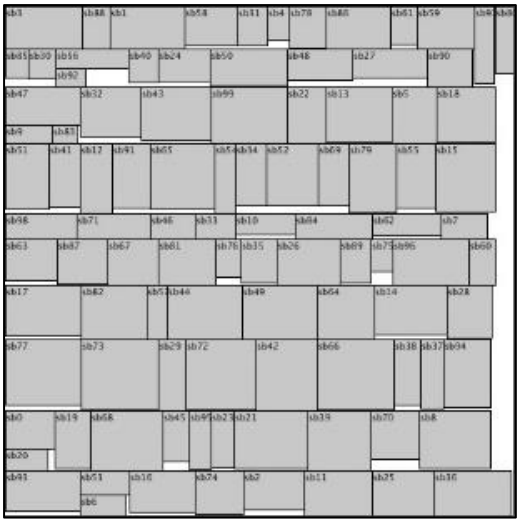
6. What have you learned from this homework?

   這次作業的 coding 不難，主要是 data structure 會影響 coding 的難度，我花了許多時間在想要怎麼建構 data structure 並實作 Stockmeyer 的方法，在幫忙 debug 的時候覺得幸虧自己有花時間在建構好的 data structure，才讓我可以方便寫出這次的程式。雖然有段時間因為有幾個 bug 沒找到導致一直擺不進去，不過在修好程式後，發現其實很多 random 都可放進去，不用是特別的 seed 才可以放進去，也表示這次開的 deadspace ratio 其實蠻寬的，另外是 SA 的參數也很重要，不同的初始溫度、每個溫度產出的答案數及溫度的降低的速率都需要調整，才可以找到 runtime 小且 wirelength 小的結果。其次是 cost function 的算法，如果不加 area 的 penalty 則會花較多的時間在 SA 上，這些都是自己要去嘗試的。

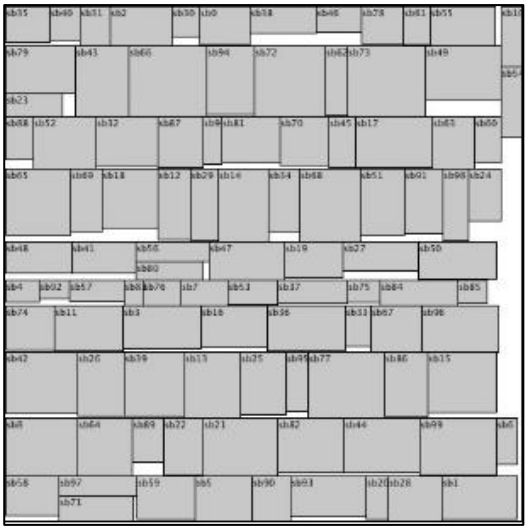
7. If you implement parallelization, please describe the implementation details and provide some experimental results.

   我沒有實作平行化的程式。

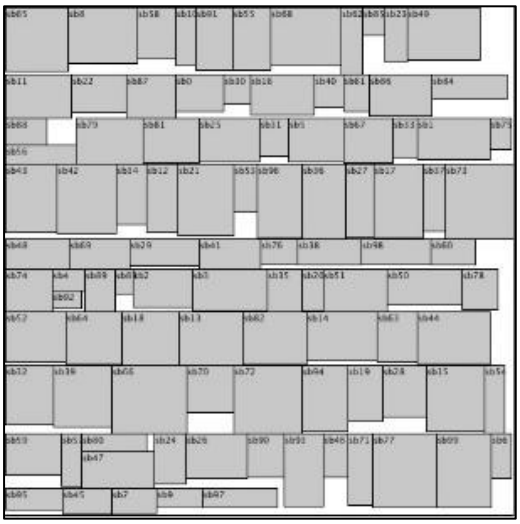
8. figure for each testcase with each deadspace ratio：



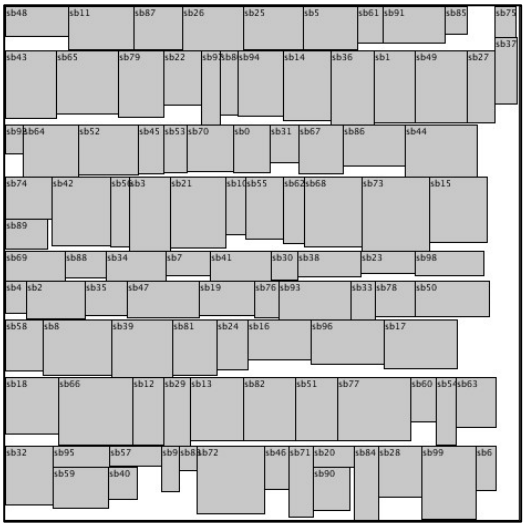
n100 ratio = 0.09



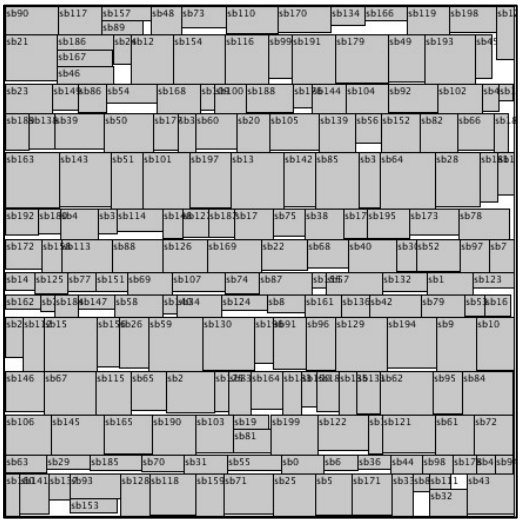
n100 ratio = 0.1



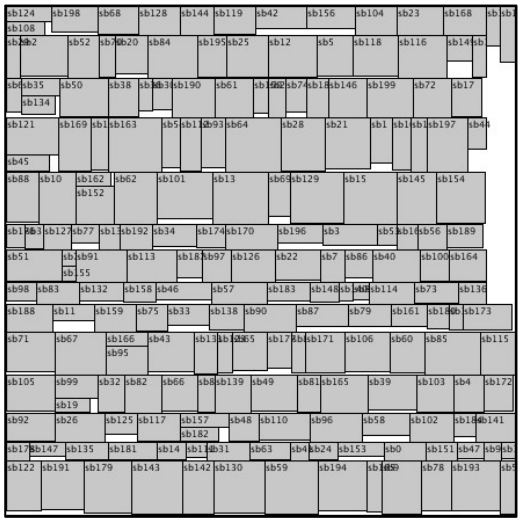
n100 ratio = 0.15



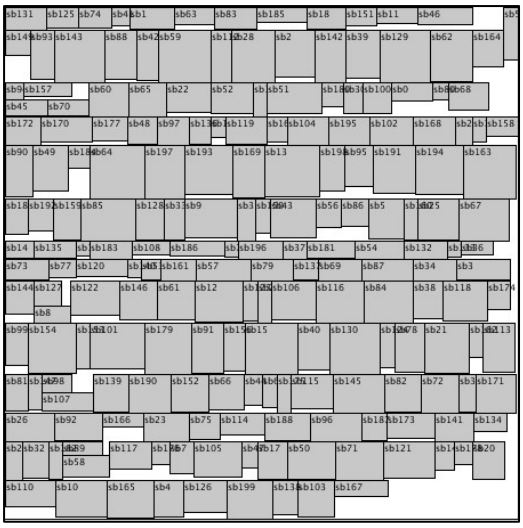
n100 ratio = 0.2



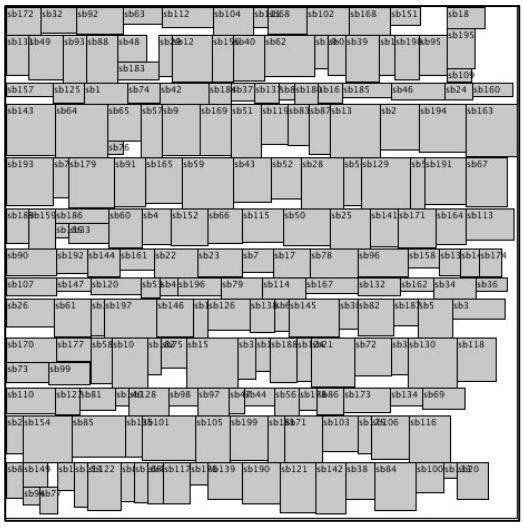
n200 ratio = 0.07



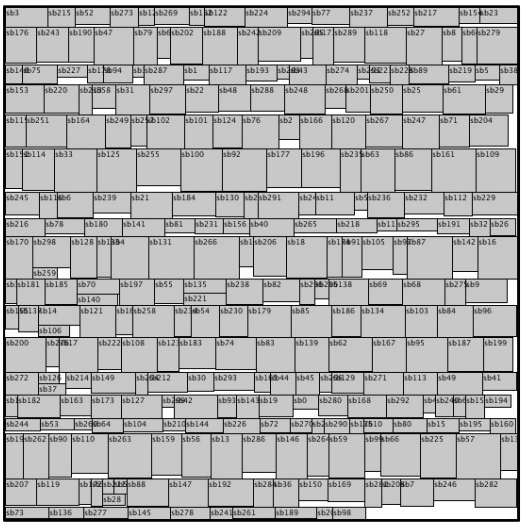
n200 ratio = 0.1



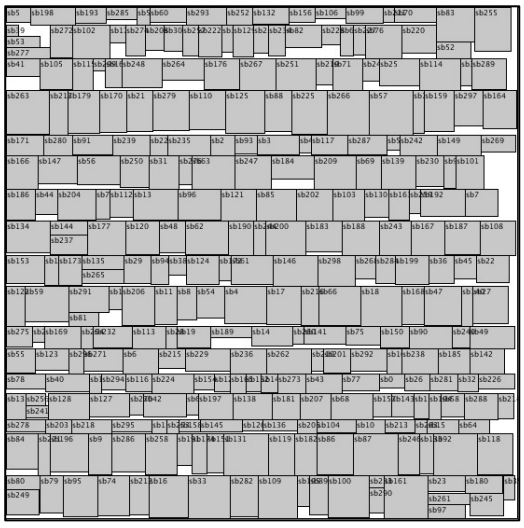
n200 ratio = 0.15



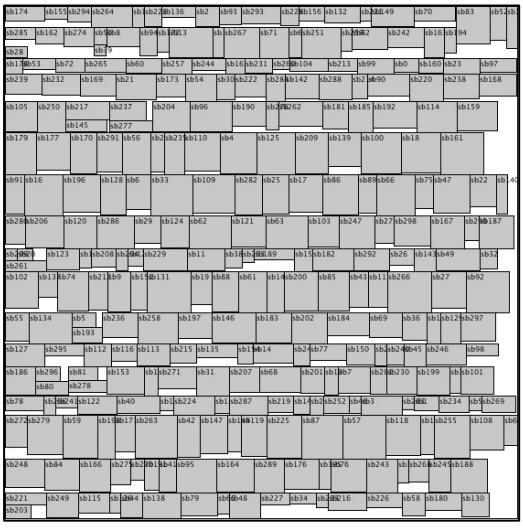
n200 ratio = 0.2



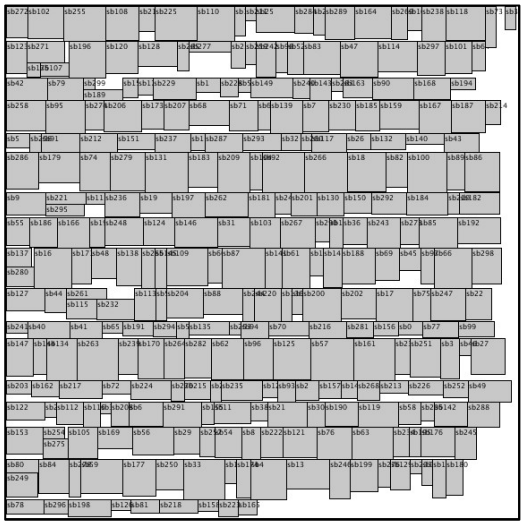
n300 ratio = 0.08



n300 ratio = 0.1



n300 ratio = 0.15



n300 ratio = 0.2