CS6135 VLSI Physical Design Automation

Homework 2: Two-way Min-cut Partitioning

(1) Name:陳博琳

Student ID:112030504

(2) --- How to compile ---

command: $ make

To generate the executable file "hw2" in "HW2/bin/"

command: $ make clean

To remove the executable file in "HW2/bin/"

--- How to run ---

In this directory , enter the following command:

command: ../bin/<exe> <testcase file> <output file>

ex. :

$ ../bin/hw2 ../testcases/public1.txt ../output/output1.out

In "HW2/bin/" , enter the following command:

command: ./<exe> <testcase file> <output file>

ex. :

$ ./hw2 ../testcases/public1.txt ../output/output1.out

(3)

testcase	Best cutsize	runningtime
Public1	251	1.15
Public2	4078	171.13
Public3	10838	292.21
Public4	4495	112.91
Public5	7893	295.48
Public6	17313	294.09

$ /usr/bin/time -p ../bin/hw2 ../testcase/public5.txt ../output/output5.out

To get each real, user, sys time. The time recording in sheet is recording real time.

(4) Use FM algorithm to partitioning the cell，if the exchange is not match to limit condition that won't swap this cell. Finding out the cell from the max gain bucket list. There is no big difference between my algorithm and FM algorithm in class.

- use push back to record all step's result in FM algorithm.
- Use map<(int)cell id,(char)die name> to store the partition set in two die

three method to initial partitioning

I. Random to put all cell in one die

II. Put all cell into DieA until it full, after that put remain cell in DieB.

III. As much as possible to separate the cell in two dies

```
g(i) ← 0;
F ← the "from block" of Cell i;
T ← the "to block" of Cell i;
for each net n on Cell i do
    if F(n)=1 then g(i) ← g(i)+1;
    if T(n)=0 then g(i) ← g(i)-1;
```

```
Algorithm: Update_Gain
1 begin  /* move base cell and update neighbors' gains */
2 F ← the Front Block of the base cell;
3 T ← the To Block of the base cell;
4 Lock the base cell and complement its block;
5 for each net n on the base cell do
   /* check critical nets before the move */
6    if T(n) = 0 then increment gains of all free cells on n
     else if T(n)=1 then decrement gain of the only T cell on n,
     if it is free
      /* change F(n) and T(n) to reflect the move */
7     F(n) ← F(n) - 1; T(n) ← T(n)+1;
     /* check for critical nets after the move */
8    if F(n)=0  then decrement gains of all free cells on n
     else if F(n) = 1 then increment gain of the only F cell on n,
     if it is free
9 end
```

(5)

Initial partition phase

I design three methods in the initial partition. I think if I have more time to try on method1 it may produce better case to me. But if we can only compile one time, I will choose method2. It can conduct the best case steadily.

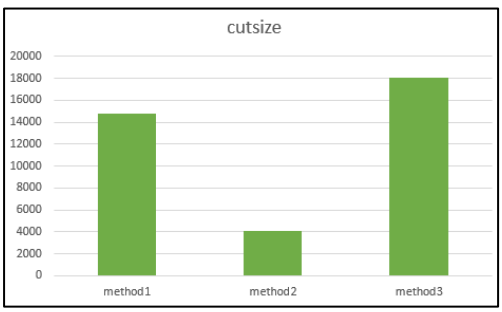
Cutsize

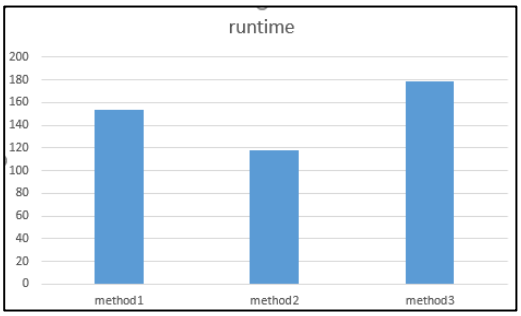
Choosing public2 to verify the cut size. we found that method2 implement the largest number of best cases. When we put all cell in one die as much as possible that we will have more area to exchange.

Runtime

Choosing public4 to verify the runtime. Method2 is the fastest one to finish the iteration.

In conclusion initial partition is a very I important section in this project. Method2 is the best choice no matter in runtime or cutsize, furthermore it can guarantee the result quality.





(6) I don't implement parallelization

(7) Conclusion

- Although I don't implement the parallelization, but I think if there are two cell and both of their swap won't bother to each other's connecting cell.
- After this project I realize the difference between map and vector, these tools are very helpful in partitioning

- Cutdown the iteration number to reduce the running time. I check if it is suitable or not to put the cell into the die, but it can’t complete even if initial partitioning.