VLSI Physical Design HW2 Report　　111062678 卓榮祥

➢ Compile and execute program

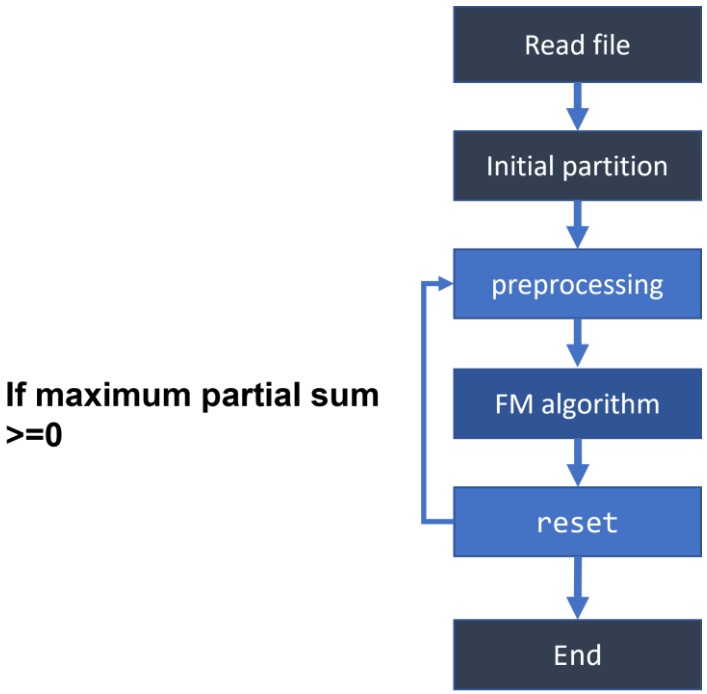
```
--How to Compile
  In "HW2/src/", enter the following command:
  $ make
  An executable file "hw2" will be generated in "HW2/bin/"

  If you want to remove it, please enter the following command:
  $ make clean

--How to Run
  Usage:
  $ ./hw2 <txt file> <out file>

  E.g., in "HW2/bin/", enter the following command:
  $ ./hw2 ../testcase/public1.txt ../output/public1.out
```

➢ The final cut size and the runtime of each testcase

testcase	Public1	Public2	Public3	Public4	Public5	Public6
cut size	436	4630	30831	2954	887	63919
runtime (s)	0.05	1.27	13.93	0.21	0.98	208.22

➢ The details of algorithm

The implemented algorithm is the FM algorithm, and its flowchart is as follows:



■ Read file

In this step, we read the input file and store various information such as the number of technologies, cell area, die size, in variables and data structures. Additionally, we create net arrays and cell arrays.

■ Initial partition

Initially, cells are placed in one of the sets while ensuring a balance condition is met:

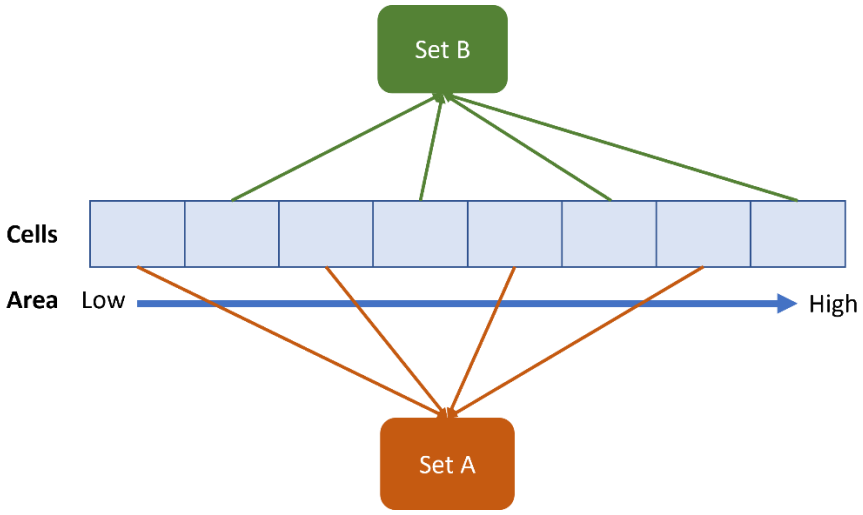
$$\frac{area(A)}{area(DieA)} \leq util(DieA) \text{ and } \frac{area(B)}{area(DieB)} \leq util(DieB)$$

However, as some input files may involve multiple technologies, and each cell's area may vary on different dies, a single partitioning method may not maintain balance in all cases. To address this, we have two partitioning approaches. If the program is unable to divide cells into two sets using the first partitioning method, it continues with the second method.

Partition 1: Place all cells in set B and arrange them in ascending order by cell area. Sequentially place them in set A in ascending order of area until the balance

condition is satisfied.

Partition 2: Cells are sorted by area in ascending order, and they are alternately assigned to set A and set B in the order of their areas.



■ Preprocessing

Calculate cell gains and create a bucket list. The bucket list is implemented using std::map<int, std::set<int>>. std::map can automatically sort the keys, making it easy to access the set of cells with the largest gain.

■ FM algorithm

Find the cell with the largest gain that also satisfies the balance condition. Update the gains of other cells and then find the maximum partial sum among the cells with a list of largest gains. Move the cells that form the maximum partial sum between sets A and B.

■ Reset

During the execution of the FM algorithm, some data values change, such as the total area of sets and net distribution. These values need to be recalculated.

Finally, check if the maximum partial sum is less than or equal to 0. If this condition

is met, terminate the algorithm; otherwise, continue the calculation.

➢ Tricks I use to enhance the solution quality

■ Storing data using std::unordered_map

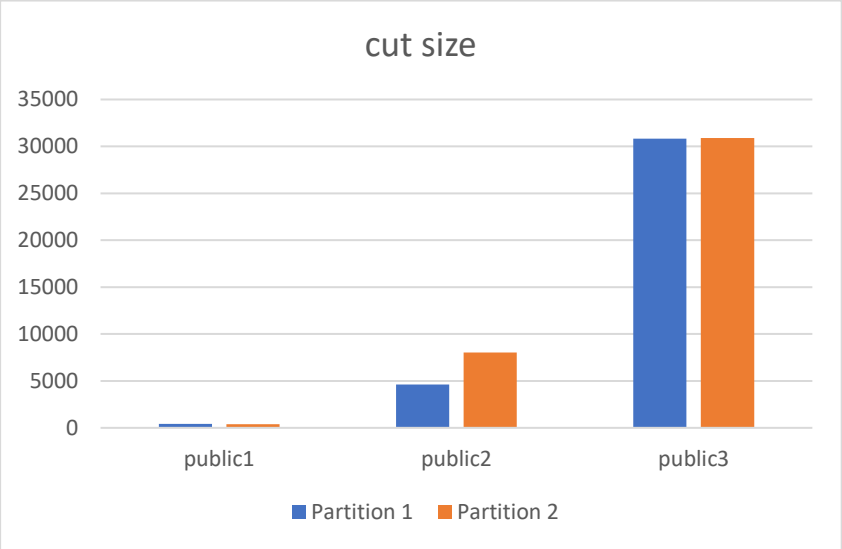
Can access Net content and Cell content and so on through **std::unordered_map** in O(1) time complexity.

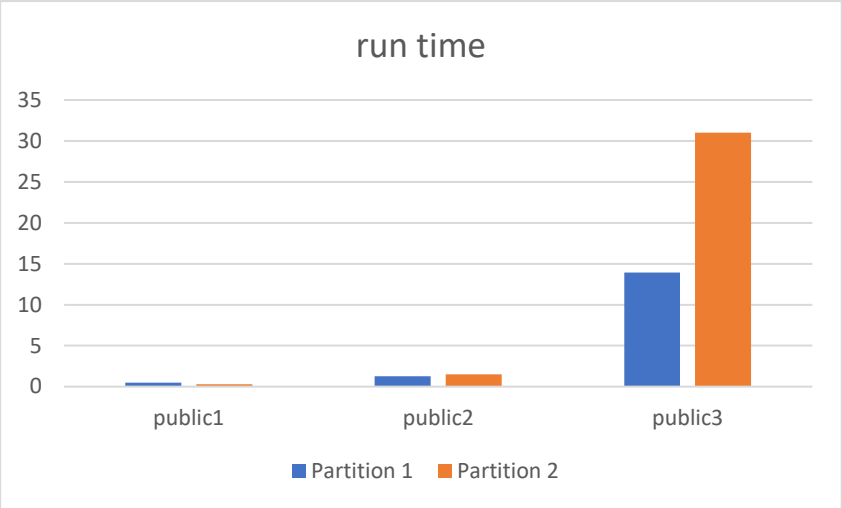
```
std::unordered_map<int, Net*> Nets;
std::vector<Cell*> Cells;
std::unordered_map< int, std::set<int> > cell_array;
std::unordered_map< int, std::set<int> > net_array;
std::set< Cell* > first_tech_set;
std::set< Cell* > sec_tech_set;
std::unordered_map<int, Cell*> Cells_hash;
```

■ Initial partition

I executed the methods mentioned above on the public testing data for both **Partition 1** and **Partition 2**. I found that the final cut set produced by Partition 1 performed better. Therefore, I have prioritized the use of Partition 1 in the program. If it's not possible to split cells using Partition 1, then I turn to using Partition 2.

In summary, I determine the priority order by testing each partition method and comparing the results.





➢ Learned from this homework

This assignment was time-consuming, especially in the initial phase where I had to organize and store data from the input file, including details about cells, nets, and sizes. Implementing the algorithm also had its challenges, like handling cells with different sizes under various technologies, which sometimes led to errors. Additionally, achieving a balanced partition during the initial phase was tricky due to differences in cell areas.

What I learned from this assignment is the crucial importance of a good initial partition. A well initial partition greatly influences the quality of final cut sets. Although I started a bit late on this assignment, given more time, I'd explore using hMETIS to improve the initial partition.