CS6135 VLSI Physical Design Automation

Homework 2: Two-way Min-cut Partitioning

112062673 吳文雯

1. How to compile & execute:

(1) To compile, enter the following command in "HW2/src/":

```
$ make
```

An executable file "hw2" will be generated in "HW2/bin/".

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

```
$ make clean
```

(2)How to execute:

Usage:

$ ./hw2 <txt input_file> <out file>

E.g. in "HW2/bin/", enter the following command:

```
$./hw2 ../testcase/public1.txt ../output/public1.out
```

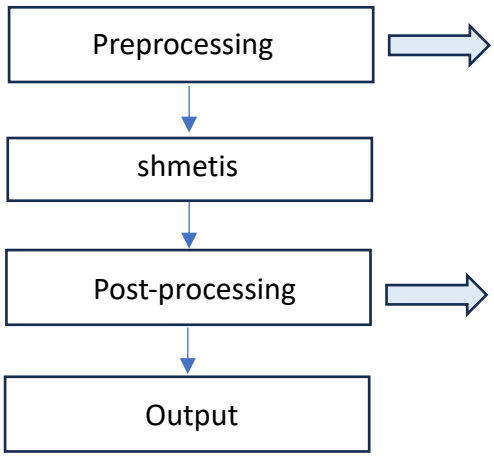
2. The final cut size and the runtime of each testcase

The following results are from HW2_grading:

```
grading on 112062673:
  testcase |     cutsize |     runtime | status
   public1 |         265 |        0.23 | success
   public2 |         841 |        4.21 | success
   public3 |        9732 |       42.98 | success
   public4 |         963 |        1.07 | success
   public5 |         945 |        7.96 | success
   public6 |        8451 |      181.19 | success
```

3. The details of the algorithm.

This is the outline of my program:



A. Read input file.
B. Store the information of the cells, nets, etc. by proper data structures.
C. Output hyperedge graph (hyper.hgr) as the input file for shmetis.

A. Get the hyperedge cut size and the partition areas generated by shmetis.
B. Read the file generated by shmetis.
C. Recalculated the area of the cells.
D. Move the cells from the over-used die to another die util the result is valid.
E. Update the cut size and the cells information.

I. Preprocessing:

- I use the following structs to store the information of technologies, dies, libs, nets, and cells.

struct Lib {
string name;
int width;
int height;
};
	struct Tech {
	string name;
	int numLib;
	int id;
	vector<Lib> libCells;
	map<string, Lib> libMap;
	};
		struct Die {
		string name;
		string techname;
		int util;
		int id;
		};

struct Net {
string name;
int numCell;
int id;
vector<Cell> netcells;
int dist[2]={0,0};
};
	struct DieSize {
	unsigned long long width;
	unsigned long long height;
	};
		struct Cell {
		string name;
		string libname;
		int id;
		vector<Net> netlist;
		};

- I also construct the following maps to get certain information quickly.

```
map<string, Tech> techMap;    //find the technology by its name
map<int, Die> dieMap;    //find the die by its id
map<string, Cell> cellMap;    // find the cell by its name
map<int, Cell> cellidMap;    // find the cell by its id
map<string, Net> netMap;    //fin the net by its name
```

  And the following vectors can retrieve all the cells and nets in the same order as the input file.

```
vector<Cell> cells;
vector<Net> nets;
```

- Then calculate the weight of each cells according to the technology adopted by Die A, and store them in an array.
- Construct hyperedge file “hyper.hgr” in the format of the inputfile (HGraphFile) for shmetis. The format is shown in Hmetis manual.

  So the input for shmetis is assumed that there is only 1 technology, the one adopted by Die A, and does not take the area constraints into consideration.

II. Shmetis

- Use the following code to make the command for shmetis to get the input file and provide the partition result as well as an output file “hyper.hgr.part.2.”

```
char cwd[1024];
string command;
if (getcwd(cwd, sizeof(cwd)) != NULL) {
    string hgrfile = string(cwd) + "/hyper.hgr";
    const char* shmetisEct = "../src/shmetis";
    command = string(shmetisEct) + " " + hgrfile + " 2 1";
    FILE* pipe = popen(command.c_str(), "r");
…
}
```

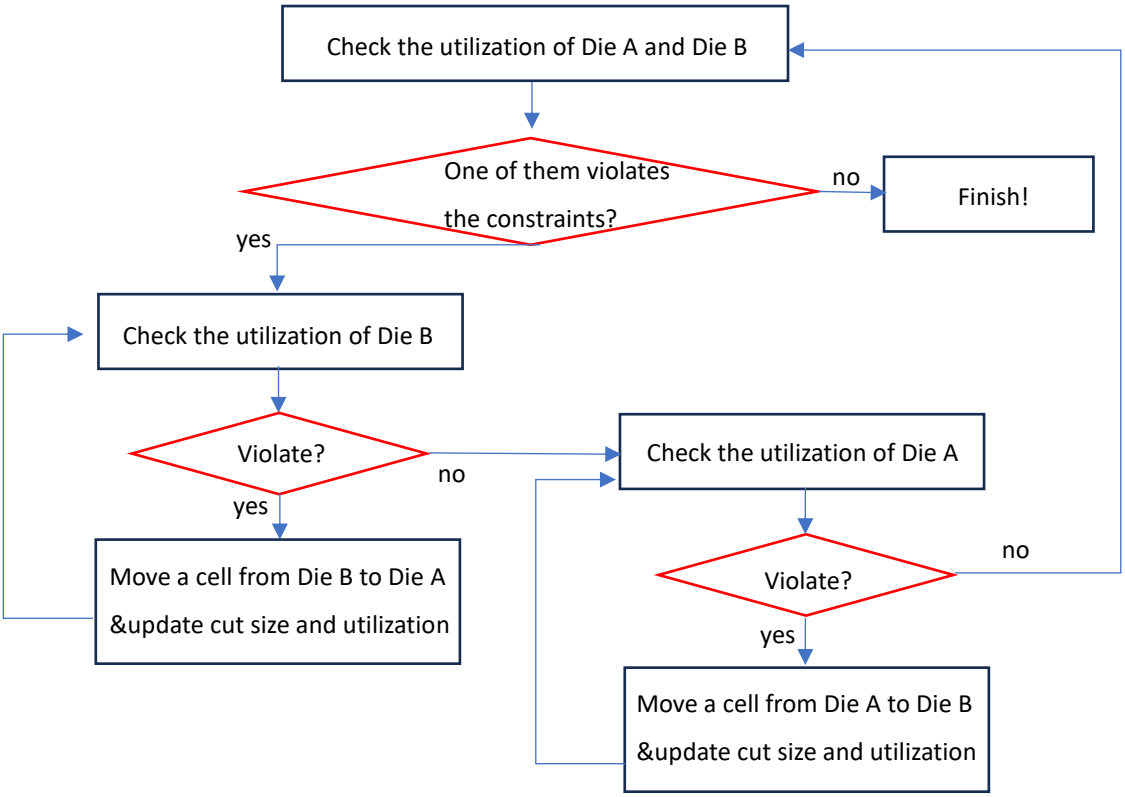
III. Post-processing:

- Use “regex_search” to get the hyperedge cut size, the area of the 2 partitions in the pipe.
- Since the size of the partitions generated by shmetis are mostly not the same, so I let the larger partition represent the die that has larger utilization constraint. And for those cells in Die B, their areas have to be recalculated according to the technology adopted by Die B. I use 2 arrays to store the area of all cells according to the technology adopted by Die A and Die B respectively.
- The following vectors are used to record which cells are on Die A and Die B.
  vector<tuple<int,int,int>> DiecellsA;
  vector<tuple<int,int,int>> DiecellsB;
  The elements of the tuple are cell id, area of the cell, and number of nets connected to the cell respectively.
- Compute the total area of Die A and Die B and their utilization at the current state.
- If the utilization constraints of the two dies are not both satisfied, the cells on the over-used side need to be moved to another die. The details are as follow.

  (i) Sorting:

  Here are 2 ways to sort the cells. (The sorting method will be determined by the feature of the input cells, which will be elaborated in section 4.)

  a. Sort the cells in decreasing order according to their area, and if the areas are the same, the one has fewer nets connected will be placed at front.
  b. Sort the cells in increasing order according to their nets, and if the number of nets are the same, the one has larger area will be placed at front.



(ii) Cells moving and cut size updating:

If Die B violated the utilization constraint, move the first cell in the vector "DiecellsB" to Die A.

Check those nets connected to that cell one by one. If the net has only 1 cell in Die B before movement, the gain is set to 1. If the net has no cell in Die A before movement, the gain is set to -1. (The way to compute gain is similar to the idea of the critical nets in FM algorithm.)

The gain contributed by all the nets connected to that first cell will be aggregated. (gains+=gain;) And the hyperedge cut will be updated as the original cut size minus those gains. (hyperedgeCut -= gains;)

(iii) Update the vectors and the total area and the utilization of Die A and Die B:

The first cell in "DiecellsB" will be pushed to the back of "DiecellsA", and it will be erased in "DiecellsB."

The total area of Die A will increase according to technology in Die A, and the total area of Die B will decrease according to technology in Die B.

If the utilization in Die B is still not satisfied, and following cells in "DiecellsB" will be moved to Die A and redo the operations in (ii) and (iii).

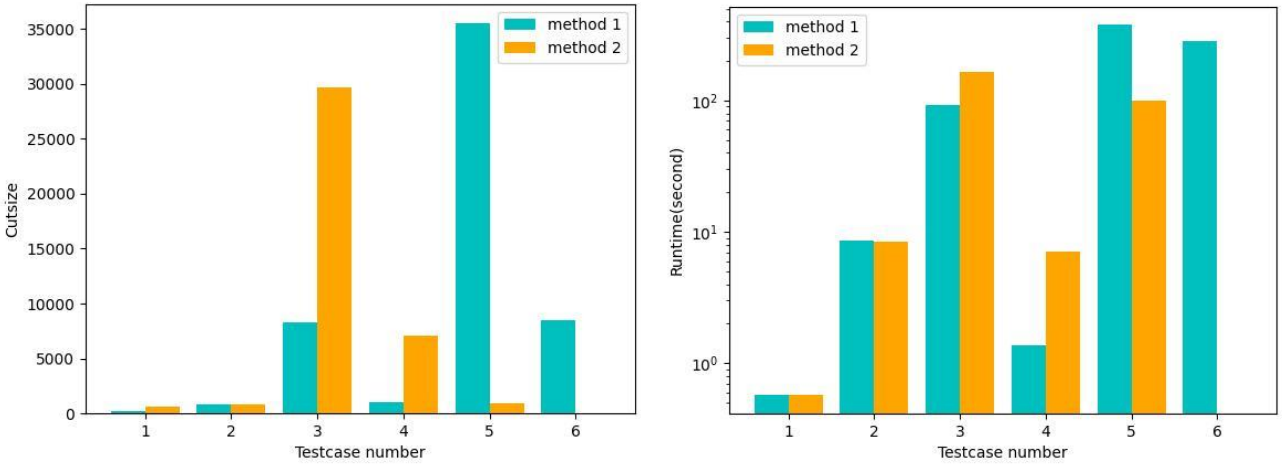
If the utilization in Die B is satisfied, the utilization in Die A will be checked. And if the utilization in Die A is violated, the cells in "DiecellsA" will be moved to Die B, which are the same operations in (ii) and (iii).

- The iterations will keep going until the utilization on the both dies are satisfied.

IV. Output:

- Sort the cells in “DiecellsA” and “DiecellsB” according to their cell id, so that later they will be write to the file in increasing order.
- Write the results to the output file in the format provided in the HW2 spec.

4. The results of different methods:



Ps. Method 2 cannot generate the result in 5 minutes for testcase 6, so there is no data.

The 2 methods are mostly the same, but they adopt different way to sort “DiecellsA” and “DiecellsB”.

- Method 1 uses a: Sort the cells in decreasing order according to their area, and if the areas are the same, the one has fewer nets connected will be placed at front.
- Method 2 uses b: Sort the cells in increasing order according to their nets, and if the number of nets are the same, the one has larger area will be placed at front.

According to the results, method 1 has better cut size for all the testcases except for testcase 5, and method 1 has similar or better run time for all the testcases except for testcase 5.

Conclusion: I should adopt method 2 for testcase 5 and adopt method 1 for the others.

Tricks:

- By observing the input of the 6 testcases, I found that the reason why testcase 5 has terrible result when adopting method 1 is that public5.txt contains over 30 cells whose area is more than 2% of the total die area. And method 1 will move the cells according to their area, but the movement of a single large cell will lead to utilization violation on another die. Thus, when there exist many large cells, it will need a lot of iterations of cells moving to get the valid result. However, the

more iterations it performs, the further the result will deviate from the cut size generated by shmetis (which should be closed to optimal solution when omitting area constraints), so the results will get worse.

- **Solution: if the input contains more than 10 cells whose area is larger than 2% of the die area, the program would use sorting method ”a” (method 1). Otherwise, use sorting method ”b” (method 2).**

5. Parallelization description:

I do not implement parallelization.

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

- Since I decide to use hMETIS in the first place, one of the challenges is how to make use of it? Therefore, I spend a lot of time reading the manual and understand the input and output of shmetis. This is the first time I write a program to interact with a commercial tool, and I finally learned how to make a command in the middle of my code as well as get the correct output from shmetis.
- One of the difficulties I met is that I tried to use the library of hMETIS at first. But after preparing all the parameters needed in the function “ HMETIS_PartRecursive,” I found that libhmetis.a seemed not compactable to the environment of the workstation, so my program cannot correctly be compiled and linked. Therefore, I rewrote my code to utilize shmetis executable file, and it also took me a lot of time to get the right way to interact with shmetis.
- Another difficulty is that shmetis cannot take area constraints into consideration, let alone adjust the area of the cells at runtime. So I have to think about how to get the correct answer with the help of shmetis. Finally. I decide to use shmetis to generate a initial partition and then adjust the cells to meet the utilization constraints by myself.

7. References:

a. George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. In *Proceedings of the Design and Automation Conference*, 1997.

b. Fiduccia and Mattheyses. “A linear time heuristic for improving network partitions,” 19th Design Automation Conf.,1982.