CS6135 VLSI Physical Design Automation

Homework 2: Two-way Min-cut Partitioning

Due: 23:59, November 6, 2022

1. Introduction

Let $\boldsymbol{C}$ be a set of cells and $\boldsymbol{N}$ be a set of nets. Each net connects a subset of cells. The two-way min-cut partitioning problem is to partition the cell set into two groups $\boldsymbol{A}$ and $\boldsymbol{B}$. The cost of a two-way partitioning is measured by the cut size, which is the number of nets having cells in both groups. In this homework, you are asked to implement **FM ALGORITHM** to solve the problem of two-way min-cut partitioning.

2. Problem Description

In this homework, we assume the two cell groups are implemented with different technologies (i.e., different standard cell library), the size of a cell will also vary according to the group it is in. The two-way min-cut partitioning problem is defined as follows:

(1) Input:

- ✓ The size of each cell for each group (.cells)
- ✓ A netlist for a circuit (.nets)

(2) Output:

- ✓ The final cut size and a partitioning result (.out)

(3) Objective:

Partition the circuit in two groups $\boldsymbol{A}$ and $\boldsymbol{B}$, such that the cut size is minimized under the constraint of $|\boldsymbol{area}(\boldsymbol{A}) - \boldsymbol{area}(\boldsymbol{B})| < \frac{|\boldsymbol{area}(\boldsymbol{A})+\boldsymbol{area}(\boldsymbol{B})|}{\boldsymbol{10}}$, where $\boldsymbol{area}(\boldsymbol{A})$ is the sum of all cell sizes in $\boldsymbol{A}$ and $\boldsymbol{area}(\boldsymbol{B})$ is the sum of all cell sizes in $\boldsymbol{B}$.

3. Input File

(1) The *.cells* file:

The .cell file specifies the cell name (unordered) and its size (a positive integer). Here is an example:

```
c1 1 2
// cellName cell size in group A cell size in group B
      ⋮
```

(2) The *.nets* file:

The .nets file specifies the netlist. Here is an example:

```
NET n1 { c2 c3 c4 }
// NET netName { cellName1 cellName2 … }
      ⋮
```

(3) The *.out* file:

Report the cells in each group and the cut size. You can run the "verify" program to check whether your result is legal or not. Here is an example:

```
cut_size 1
// cut_size cut size
A 4
// A number of cells in group A
c2
// cellName
     ⋮
B 4
// B number of cells in group B
c1
     ⋮
```

Please see the following example which shows the input files, a partitioning result, and the corresponding output files.

Example:

.cells

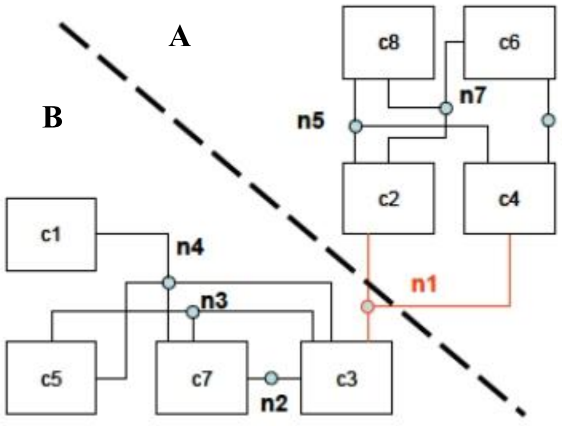
```
c1 1 2
c2 3 9
c3 2 6
c4 1 4
c5 1 3
c6 4 12
c7 2 4
c8 5 15
```

.nets

```
NET n1 { c2 c3 c4 }
NET n2 { c3 c7 }
NET n3 { c3 c5 c7 }
NET n4 { c1 c3 c5 c7 }
NET n5 { c2 c4 c8 }
NET n6 { c4 c6 }
NET n7 { c2 c6 c8 }
```

.out

```
cut_size 1
A 4
c2
c4
c6
c8
B 4
c1
c3
c5
c7
```



4. Language/Platform

(1) Language: C/C++

(2) Platform: Unix/Linux

5. Report

Your report should contain the following content, and you can add more as you wish.

(1) Your name and student ID

(2) How to compile and execute your program, and give an execution example.

(3) The final cut size and the runtime of each testcase

P.S. You could use the command `time` to measure runtime.

e.g., `$ /usr/bin/time -p ./hw2 …`

```
$ echo "alias time '/usr/bin/time -p'" >> ~/.tcshrc
```

Re-log in then `$ time ./hw2 …` is equal to `$ /usr/bin/time -p ./hw2 …`

(4) Runtime = $T_{IO} + T_{computation}$. For each case, please analyze your runtime and find out how much time you spent on I/O and how much time you spent on the computation (FM Algorithm).

(5) The details of your implementation containing explanations of the following questions:

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

II. Did you implement the bucket list data structure?

✓ If so, is it exactly the same as that described in the slide? How many are they?

✓ If not, why? You had a better data structure? Or, is bucket list useless?

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

IV. What else did you do to enhance your solution quality (you are required to implement at least one method to enhance your solution quality) and to speed up your program?

V. If you implement parallelization (for FM algorithm itself), please describe the implementation details and provide some experimental results.

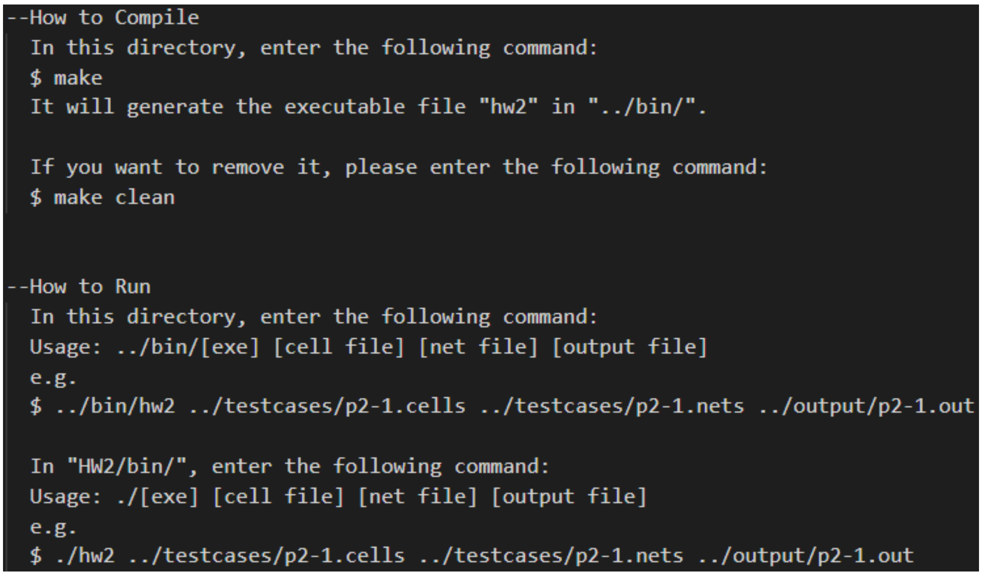
(6) What have you learned from this homework? What problem(s) have you encountered in this homework?

6. Required Items

Please compress `HW2/` (using tar) into one with the name `CS6135_HW2_${StudentID}.tar.gz` before uploading it to eeclass.

(1) `src/` contains all your source code, your `Makefile` and `README`.

➢ `README` must contain how to compile and execute your program. An example of `README` is like the following figure:



```
--How to Compile
  In this directory, enter the following command:
  $ make
  It will generate the executable file "hw2" in "../bin/".

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


--How to Run
  In this directory, enter the following command:
  Usage: ../bin/[exe] [cell file] [net file] [output file]
  e.g.
  $ ../bin/hw2 ../testcases/p2-1.cells ../testcases/p2-1.nets ../output/p2-1.out

  In "HW2/bin/", enter the following command:
  Usage: ./[exe] [cell file] [net file] [output file]
  e.g.
  $ ./hw2 ../testcases/p2-1.cells ../testcases/p2-1.nets ../output/p2-1.out
```

(2) `output/` contains all your outputs of testcases for the TA to verify.

(3) `bin/` contains your executable file.

(4) `CS6135_HW2_${StudentID}_report.pdf` contains your report.

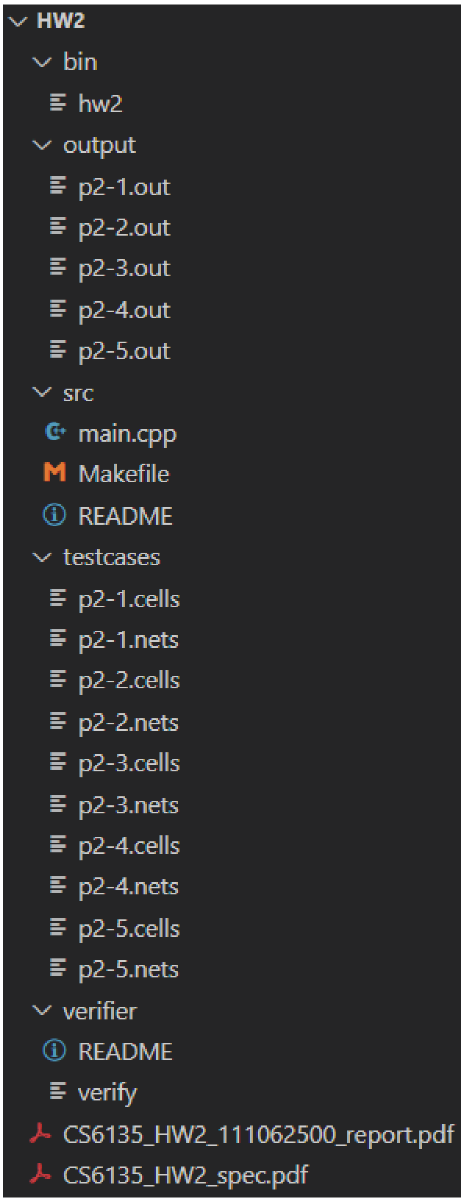
You can use the following command to compress your directory on a workstation:

```
$ tar -zcvf CS6135_HW2_${StudentID}.tar.gz <directory>
```

For example:

```
$ tar -zcvf CS6135_HW2_111062500.tar.gz HW2/
```

The file structure would be like the following figure:



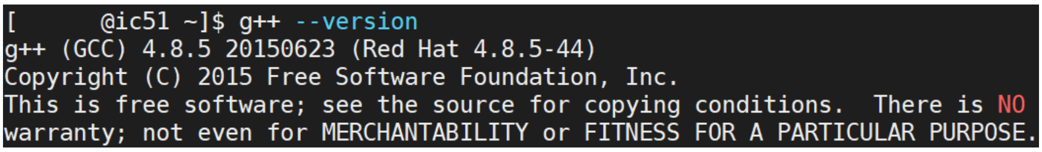
```
HW2
  bin
    hw2
  output
    p2-1.out
    p2-2.out
    p2-3.out
    p2-4.out
    p2-5.out
  src
    main.cpp
    Makefile
    README
  testcases
    p2-1.cells
    p2-1.nets
    p2-2.cells
    p2-2.nets
    p2-3.cells
    p2-3.nets
    p2-4.cells
    p2-4.nets
    p2-5.cells
    p2-5.nets
  verifier
    README
    verify
  CS6135_HW2_111062500_report.pdf
  CS6135_HW2_spec.pdf
```

7. Grading

- ✓ 80%: The solution quality (final cut size) of each testcase, hidden testcases included. This part will be evaluated with single thread version.
- ✓ 20%: The completeness of your report
- ✓ **5% (bonus)**: Parallelization

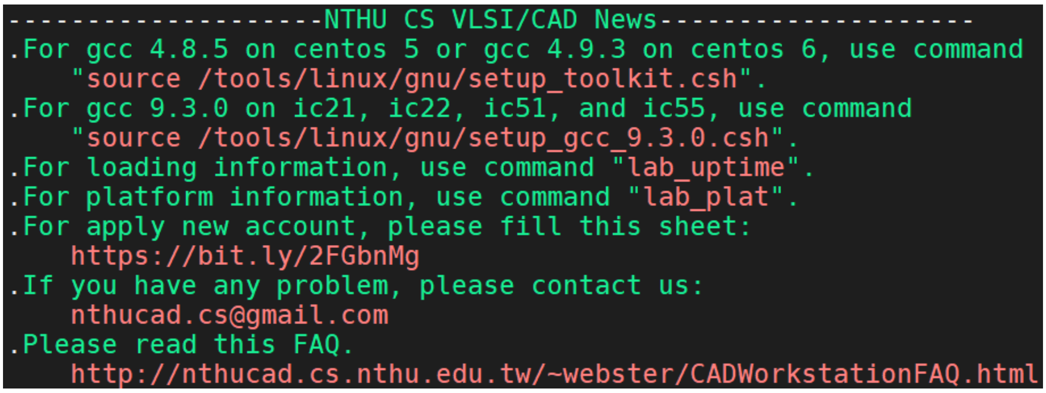
P.S. Using C++11, C++14, or C++17

The C++11 standard is implemented in GCC 4.8.1 and beyond. If you want to use C++14 or C++17, you need to use GCC 6.1 and beyond.



```
[      @ic51 ~]$ g++ --version
g++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44)
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

If you want to change the GCC version, you can follow the news of the login information and change your GCC version by yourself.



```
--------------------NTHU CS VLSI/CAD News--------------------
.For gcc 4.8.5 on centos 5 or gcc 4.9.3 on centos 6, use command
    "source /tools/linux/gnu/setup_toolkit.csh".
.For gcc 9.3.0 on ic21, ic22, ic51, and ic55, use command
    "source /tools/linux/gnu/setup_gcc_9.3.0.csh".
.For loading information, use command "lab_uptime".
.For platform information, use command "lab_plat".
.For apply new account, please fill this sheet:
    https://bit.ly/2FGbnMg
.If you have any problem, please contact us:
    nthucad.cs@gmail.com
.Please read this FAQ.
    http://nthucad.cs.nthu.edu.tw/~webster/CADWorkstationFAQ.html
```

In this way, you need to source it every time when you log in on a server. Instead, you can create a shell resource file called `.tcshrc` in the root folder and put the source command in it. Just enter the following command once, re-login, and then it will never bother you anymore.

```
$ echo "source /tools/linux/gnu/setup_gcc_9.3.0.csh" >> ~/.tcshrc
```

P.S. Using Boost C++ Library

The boost C++ library is installed on ic21, ic22, ic51, and ic55. If you want to use boost C++ library, you must add the following include path while compiling your source code.

```
-I /usr/local/include/boost/
```

For example:

```
$ g++ -O3 -std=c++11 -I /usr/local/include/boost/ main.cpp -o hw2
```

Notes:

- Make sure the following commands can be executed.
  - Go into directory “`src/`”, enter “`make`” to compile your program and generate the executable file, called “`hw2`”, which will be in directory “`bin/`”.
  - Go into directory “`src/`”, enter “`make clean`” to delete your executable file.
- Please use the following command format to run your program.

  ```
  $ ./hw2 *.cells *.nets *.out
  ```

  E.g.:

  ```
  $ ./hw2 ../testcases/p2-1.cells ../testcases/p2-1.nets
  ../output/p2-1.out
  ```
- If you implement parallelization, please name your parallel version executable file as “`hw2_parallel`” and name your sequential version executable file as “`hw2`”.

- Use arguments to read the file path. Do not write the file path in your code.
- Program must be terminated within 5 minutes for each testcase.
- Please use ic21, ic22, ic51 or ic55 to test your program.
- We will test your program by shell script with GCC 9.3.0 on ic51. Please make sure your program can be executed by **HW2_grading.sh**.
- Note that any form of plagiarism is strictly prohibited. If you have any problem, please contact TA.