CS6135 VLSI Physical Design Automation

Homework 2: Two-way Min-cut Partitioning

Due: 23:59, April 1, 2025

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 disjoint groups $\boldsymbol{A}$ and $\boldsymbol{B}$, where each group of cells is put in a different die. The cost of a two-way partitioning is measured by the cut size, which is the sum of the weight of each net having cells in both groups. In this homework, you are asked to implement an existing algorithm or develop your own algorithm to solve the problem of two-way min-cut partitioning.

2. Problem Description

In this homework, we assume that two dies are implemented with different technologies, and you are asked to partition all cells into two groups each of which is put in a different die without violating the maximum area utilization constraint of the die. The size of each die is the same. The size of a cell could vary according to the technology of the die it is in. The two-way min-cut partitioning problem is defined as follows:

(1) Input:

- ✓ Technology and corresponding standard cell library
- ✓ Die size
- ✓ Technology and maximum area utilization for each die
- ✓ Cell and corresponding library cell
- ✓ Netlist

(2) Output:

- ✓ The final cut size and a partitioning result

(3) Objective:

Partition the circuit in two groups $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively corresponding to $\boldsymbol{DieA}$ and $\boldsymbol{DieB}$, such that the cut size is minimized subject to the following condition.

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

Here $\boldsymbol{area}(\boldsymbol{A})$ is the sum of all cell areas in $\boldsymbol{A}$, $\boldsymbol{area}(\boldsymbol{B})$ is the sum of all cell areas in $\boldsymbol{B}$, $\boldsymbol{area}(\boldsymbol{DieA})$ is the area of $\boldsymbol{DieA}$, $\boldsymbol{area}(\boldsymbol{DieB})$ is the area of $\boldsymbol{DieB}$, $\boldsymbol{util}(\boldsymbol{DieA})$ is the maximum area utilization of $\boldsymbol{DieA}$, and $\boldsymbol{util}(\boldsymbol{DieB})$ is the maximum area utilization of $\boldsymbol{DieB}$.

3. Input File

(1) The *.txt* file:

The .txt file specifies the input information. Here is an example:

```
NumTechs 2
// NumTechs number of technologies
Tech TA 3
// Tech technology name number of library cells
LibCell MC1 7 10
// LibCell library cell name library cell width library cell height
      ⋮

DieSize 40 30
// DieSize die width die height
DieA TA 80
// DieA technology of DieA max area utilization percentage of DieA
DieB TB 90
// DieB technology of DieB max area utilization percentage of DieB

NumCells 8
// NumCells number of cells
Cell C1 MC1
// Cell cell name library cell name
      ⋮

NumNets 6
// NumNets number of nets
Net N1 2 3
// Net net name number of cells on the net net weight
Cell C1
// Cell cell name
      ⋮
```

4. Output File

(1) The *.out* file:

It reports 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:

```
CutSize 1
// CutSize cut size
DieA 4
// DieA number of cells on DieA
C1
// cell name
      ⋮
DieB 4
// DieB number of cells on DieB
C2
      ⋮
```

5. Language/Platform

(1) Language: C/C++

(2) Platform: Unix/Linux

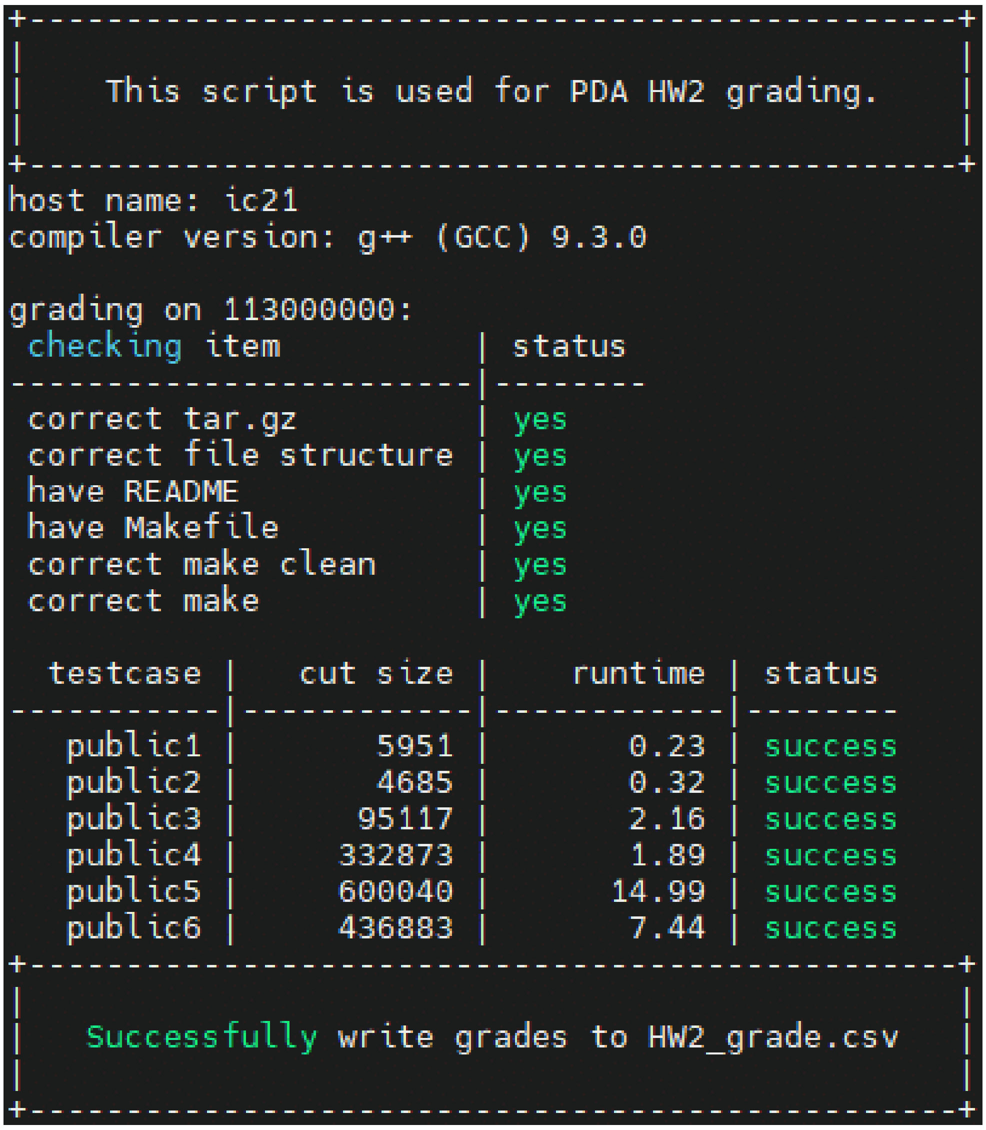
6. Report

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. Paste the screenshot of the result of running the **HW2_grading.sh** as the picture shown below.



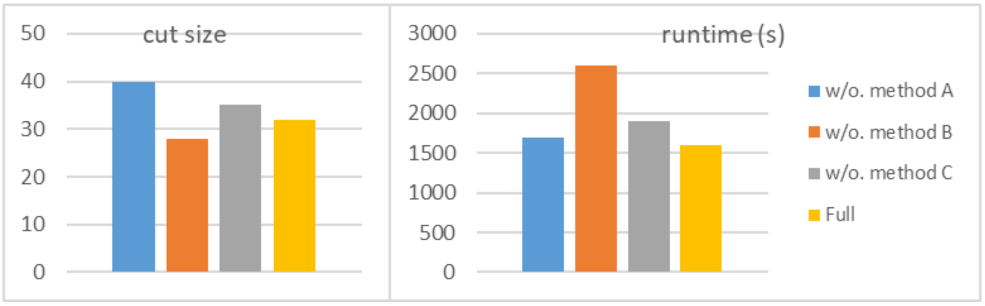
```
+---------------------------------------------------+
|                                                   |
|     This script is used for PDA HW2 grading.     |
|                                                   |
+---------------------------------------------------+
host name: ic21
compiler version: g++ (GCC) 9.3.0

grading on 113000000:
 checking item            | status
--------------------------|--------
 correct tar.gz           | yes
 correct file structure   | yes
 have README              | yes
 have Makefile            | yes
 correct make clean       | yes
 correct make             | yes

  testcase |    cut size |     runtime | status
-----------|-------------|-------------|--------
   public1 |        5951 |        0.23 | success
   public2 |        4685 |        0.32 | success
   public3 |       95117 |        2.16 | success
   public4 |      332873 |        1.89 | success
   public5 |      600040 |       14.99 | success
   public6 |      436883 |        7.44 | success
+---------------------------------------------------+
|                                                   |
|    Successfully write grades to HW2_grade.csv     |
|                                                   |
+---------------------------------------------------+
```

(4) The details of your algorithm. You could use flow chart(s) and/or pseudo code to help elaborate your algorithm. If your algorithm is similar to some previous work, please cite the corresponding paper(s) and reveal your difference(s).

(5) What techniques did you use to improve your solution's quality? Additionally, analyze how they contributed to the improvements. Plot the effects of those different settings like the ones shown below.



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

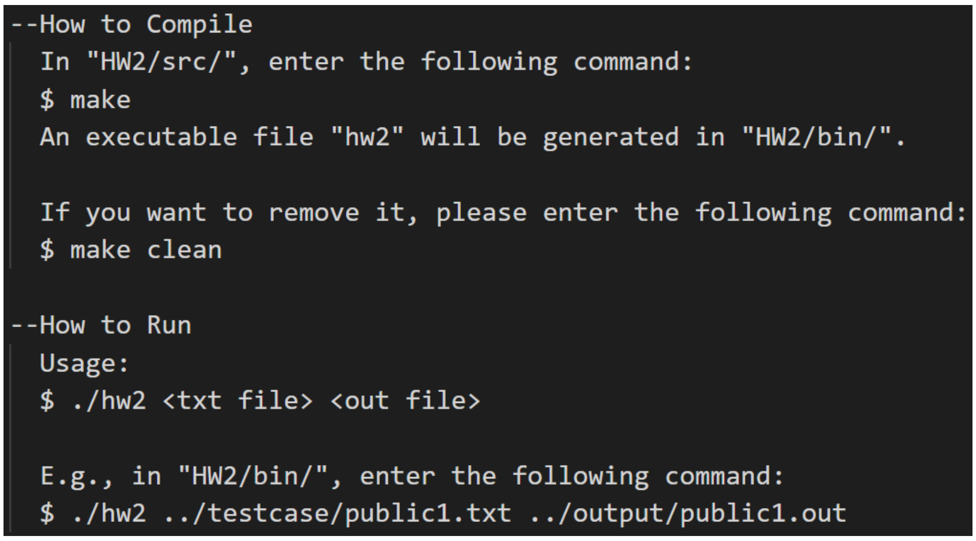
(7) What have you learned from this homework? What problem(s) have you encountered in this homework?

7. 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 "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
```

(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.

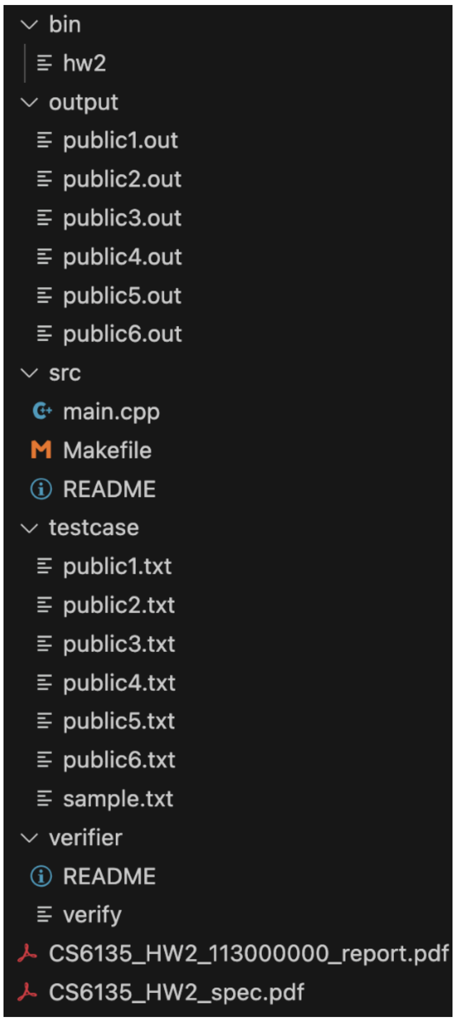
You can use the following command to compress your directory on a workstation:

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

For example:

```
$ tar -zcvf CS6135_HW2_113000000.tar.gz HW2/
```

The folder structure would be like the following figure:



```
bin
  hw2
output
  public1.out
  public2.out
  public3.out
  public4.out
  public5.out
  public6.out
src
  main.cpp
  Makefile
  README
testcase
  public1.txt
  public2.txt
  public3.txt
  public4.txt
  public5.txt
  public6.txt
  sample.txt
verifier
  README
  verify
CS6135_HW2_113000000_report.pdf
CS6135_HW2_spec.pdf
```

8. Grading

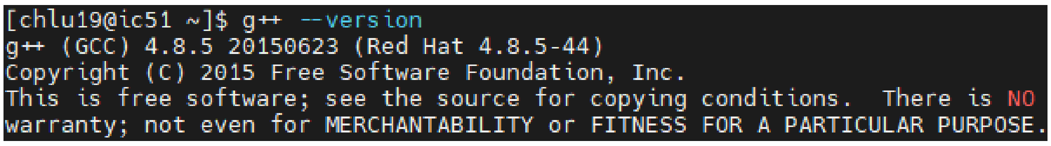
- ✓ 50%: Outperform the baseline in public testcases. The cut size of baseline for each public testcase is listed below. For hidden testcases, you only need to generate a valid result.

Testcase	Cut Size
public1.txt	10001
public2.txt	5809
public3.txt	98569
public4.txt	333356
public5.txt	632749
public6.txt	484267

- ✓ 30%: The solution quality (final cut size) of each testcase, hidden testcases included. This part will be evaluated with the sequential version of your program.
- ✓ 20%: The completeness of your report
- ✓ **5% Bonus**: Parallelization.

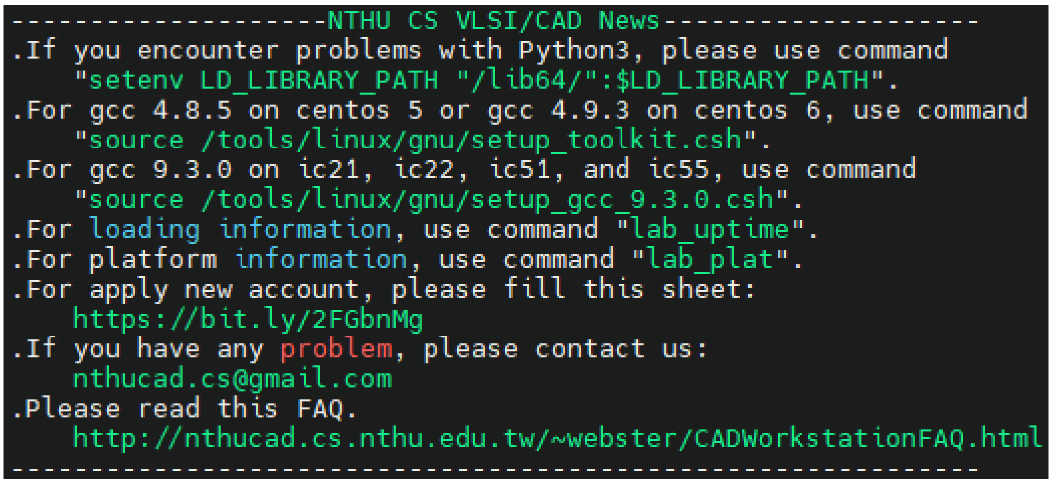
P.S. Using C++11, C++14, or C++17

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.



```
[chlu19@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--------------------
.If you encounter problems with Python3, please use command
    "setenv LD_LIBRARY_PATH "/lib64/":$LD_LIBRARY_PATH".
.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 *.txt *.out
  ```

  E.g.:

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

- If you implement parallelization, please name the executable file of your parallel version as "`hw2_parallel`" and name the executable file of your sequential version as "`hw2`".
- Use arguments to read the file path. Do not write the file path in your code.
- Program must be terminated within 3 minutes for each testcase.
- Please use ic21, ic22, ic51, or ic55 to test your program.
- We will test your program by a shell script with GCC 9.3.0 on the servers mentioned above. Please make sure your program can be executed by **HW2_grading.sh**. If we cannot compile or execute your program by the script, you will get 0 points on your programming score.
- Note that any form of plagiarism is strictly prohibited, including the code found on GitHub and the code from any student who took this course before. If you have any problem, please contact TA.