UCLA EE 201A -- VLSI Design Automation – Winter 2024
Lab 3: Incremental Placement Using OpenAccess

TA: Dedeepyo Ray

****** IMPORTANT ******

- Run your experiments and do your work on SEASnet server eeapps.seas.ucla.edu
- All your work should be done in `/w/classproj/ee201a/YOUR_SEASNET_USERNAME`
    - DO NOT do the lab in your SEASnet home directory (~). There isn't enough disk space.
    - You are responsible for backing up your code & data.
- All necessary files can be found in `/w/class.1/ee/ee201o/ee201ot2/2024_labs/lab3`
- **See last page for important submission instructions**
- **Due Date: Tuesday, Feb 17th, 2024 at 11:59:59 P.M. Pacific Daylight Time (PDT)**

Notes:

- *You will find it useful to refer to the OA tutorial from class and the official OA documentation.*
- *Use the skeleton C++ code provided in the Lab 3 materials directory listed above – it is nearly identical to the Lab 1 skeleton. You will likely find it helpful to refer to your own Lab 1 solution as frequently as needed to develop your solution for this lab. We've provided a shell script to import the design into an OA database for your convenience.*
- *You should do Problem 1 first, as Problem 2 relies on it working correctly.*
- *Please use Piazza for asking (and answering) questions before coming to office hours. However, you are expected to work on the assignment alone, so do not provide solutions or compare results with your peers.*

For this assignment, you will develop an incremental placement algorithm and implement it in C++ using the OpenAccess (OA) API.

Problem 1 (4 points): Determine the total wirelength for all nets.

This lab builds on your work from Lab 1, where you computed the half-perimeter wirelength (HPWL) of logic nets in a design. Write a small C++ routine to determine the HPWL for *all* nets in the design, including power, ground, clock, floating, etc. Justify your assumptions as necessary, like you did in Lab 1. For nets with more than two endpoints, use the half perimeter of the smallest bounding box that includes all endpoints. Output the *total* HPWL of the design.

In the report, describe your algorithm for computing total HPWL in words and include the total HPWL for the design.

Problem 2 (6 points): Incremental placement.

Write a C++ routine that incrementally improves the placement of a design by *swapping* the locations of individual blocks/gates to reduce the total HPWL. To simplify this problem, please limit the swaps to identical cells only and with exactly the same instance orientation. So, for an `INV_X1` cell instance with `oaOrient = 2` (i.e., 180 degree rotation), the algorithm may swap it only with another `INV_X1` cell instance with exactly the same orientation. You cannot move instances in any way besides swapping.

There are many ways to solve this problem -- feel free to be creative! One easy way would be to first determine the improvement in total HPWL with all possible individual swaps of instance pairs. Then, make the best choice swap and "lock" the pair as well as any connected instances so that they cannot be swapped again. Repeat until there are no more possible swaps that are beneficial. Output the best achievable total HPWL after your incremental placement algorithm, and the number of swaps it took to achieve it.

In the report, describe your algorithm and your approach, your HPWL after incremental placement, the number of swaps it took to achieve this, and the total execution time for your program.

Grading Notes: 2 out of the 6 points on Problem 2 will be awarded based loosely on your ranking in the class on the following metric: total HPWL $(DBU)^2$ * runtime (seconds), where a lower metric score is better. Note that not attempting incremental placement or HPWL improvement will get 0/6 on this problem even if you have very good runtime (say, by instantly exiting the program in main(), which would defeat the purpose of the assignment).

SUBMISSION INSTRUCTIONS (Read carefully)

- All necessary code, data, and report files must be tarballed and submitted as a single archive file on eeapps.
  - For final submission, create a directory named as follows: `Lastname-Firstname_UID_Username_Lab3/`
  - Inside this submission directory, you must include exactly three files:
    - `Lastname-Firstname_UID_Username_Lab3.cpp` (source code for lab)
    - `Lastname-Firstname_UID_Username_Lab3` (compiled binary file for lab)
    - `Lastname-Firstname_UID_Username_Lab3.pdf` (lab report – answer any questions here)
    - `Lastname-Firstname_UID_Username_Lab3_results_submission.txt` (write your answers in here for automatic parsing, same as lab2).
  - Compress and archive this directory using tar/gzip to have a single submission file named `Lastname-Firstname_UID_Username_Lab3_pinXXXX.tar.gz` (Make up a 4-digit numeric PIN of your choice to substitute for `XXXX`)
  - Submit tarball by copying it to `/w/class.1/ee/ee201o/ee201ot2/submission/lab3/`
    - IMPORTANT: Make sure all files you submit, as well as the tarball, have full read and execute permissions to group and others or we will not be able to grade your lab. Do this using `chmod -R go+rx FILE_OR_DIRECTORY`
  - **Late submissions will not be accepted** (or possible – read/write/execute permissions to `submission/lab3/` will be disabled at the deadline). If you cannot finish the entire assignment on time, submit whatever you completed. **No submission = no points.** You are welcome to overwrite your submission as many times as you like before the deadline. However, please only use the filename format that is provided. Duplicates of the form "v1, v2, v3, new, newest," etc. will be ignored. If you accidentally submitted your tarball multiple times using different PINs, only the latest timestamp will be considered.
  - Some students may worry that their work could be visible to other students. We try to reduce this chance by disabling read permissions on `/w/class.1/ee/ee201o/ee201ot2/submission/lab3/` directory so that other students cannot list its contents. Thus, they will not know the filename and cannot open your submission unless you give them your full name, student ID number, SEAS username, and arbitrary 4-digit PIN that you substituted for `XXXX` earlier on the tarball. This also means you will not be able to list the directory contents to see if your own submission worked – you can only write to it! In short, please do not give your classmates this information or collaborate on homeworks. The PIN can be whatever 4-digit number you want -- it is just to reduce the likelihood of another student guessing your full file submission path. You can change it for every lab submission if you like.
  - Submission example step-by-step:

```
$ cd /w/class/ee201a/mgottscho
$ mkdir Gottscho-Mark_203555232_gottscho_Lab3
$ cp foo.cpp Gottscho-Mark_203555232_gottscho_Lab3/Gottscho-
  Mark_203555232_gottscho_Lab3.cpp
$ cp bar
  Gottscho-Mark_203555232_gottscho_Lab3/Gottscho-Mark_203555232_gottscho
  _Lab3
$ cp report.pdf
  Gottscho-Mark_203555232_gottscho_Lab3/Gottscho-Mark_203555232_gottscho
  _Lab3.pdf
$ chmod -R go+rx Gottscho-Mark_203555232_gottscho_Lab3/
$ tar -czf Gottscho-Mark_203555232_gottscho_Lab3_pin1543.tar.gz
  Gottscho-Mark_203555232_gottscho_Lab3/
$ chmod go+rx Gottscho-Mark_203555232_gottscho_Lab3_pin1453.tar.gz
```

```
$ cp Gottscho-Mark_203555232_gottscho_Lab3_pin1543.tar.gz
  /w/class.1/ee/ee201o/ee201ot2/submission/lab3/Gottscho-Mark_203555232_
  gottscho_Lab3_pin1543.tar.gz
```