UCLA EE 201A -- VLSI Design Automation -- Winter 2018
Course Project: Automated Inter-Chip Pin Assignment

TA: Saptadeep Pal and Wei-Che Wang

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

- Run your experiments and do your work on SEASnet server `eeapps.seas.ucla.edu`
- You will work in teams of two students. The project has a competitive grading component.
  - How you coordinate your work is up to you. We recommend a version control system like Git.
  - Be careful not to reveal your project directory to other teams!
- All your work should be done in `/w/class.2/ee/ee201a/YOUR_SEASNET_USERNAME/`
  - Don't do the project in your SEASnet home directory (~).
  - You are responsible for backing up your code & data.
- You can get the latest version of the materials at `/w/class.2/ee/ee201a/ee201ota/material/project/`
  - Subject to updates over the quarter, but will be uploaded as separate tarballs (v1, v2, etc).
- **Due Dates:**
  - **1) Final Code Submission – Monday, March 12, 2018 @ 10:00:00 PM PT**
  - **2) Presentations w/TA-Verified Results – Thursday, March 15, 2018 (in class)**
- Start early! This project will be challenging. There are also expected license and server load issues from this as well as other courses in the last few weeks. You should try and get most work done before that.
- See the end for important submission instructions.

Project Overview

Routing at the global chip/system level connecting pins of various design blocks, soft/hard IP blocks or macros is often a critical part of the physical design flow. Due to the size and complexity of the design, each block or module is often individually processed to create a layout, and then combined with other macros in the system floorplan. However this approach leaves the pin assignment inside the block agnostic of the global net connectivity. This has adverse effects on performance, timing/skew matching as well as overall system area.

Your job is to research and develop a prototype pin assignment tool in C++ with the OpenAccess API. The input floorplan would have hard macro blocks (that can be treated as black-boxes for our purpose) with an initial default pin placement. Your tool would then need to allocate improved pin locations along the block peripheries to minimize routing wirelength, subject to various constraints described in later sections.

Your overall goal should be to learn how to approach open-ended electronic design problems through an automated solution. Ideally, you would produce a robust and useful tool that could be leveraged by others. This means the tool should not be designed to just barely satisfy the project requirements. Your approach should take a broader scope by incorporating relevant ideas from research literature. You would also need to budget time sufficient for an iterative design process, since the complete flow includes testing on a commercial P&R tool with significant runtimes due to design complexity.

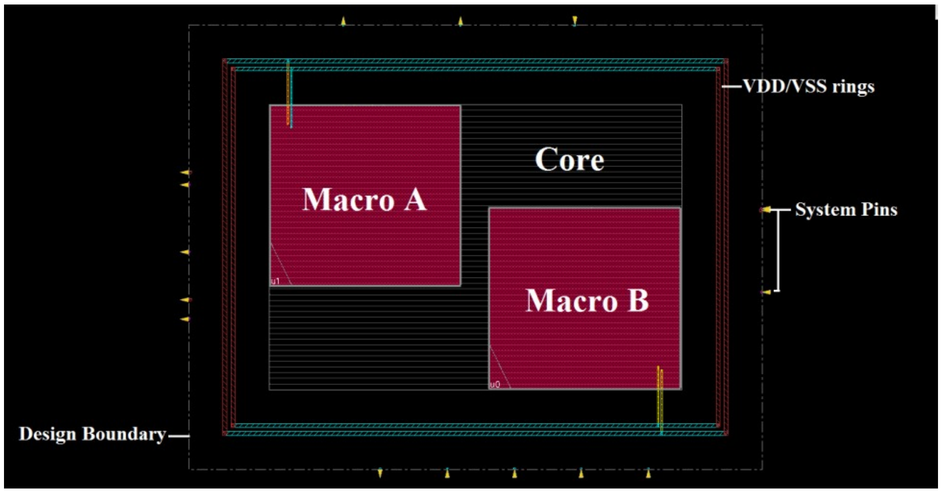
Your final solution should be well designed, implemented, and documented. The project also includes a presentation that is a significant component of your grade. For a good grade, make sure you leave enough time to do this well! All of these skills are essential to working in industry.

The project may appear very daunting at first glance. However, as you familiarize yourself with the tool and assignment, it will get easier. Don't get discouraged if you are having difficulty meeting the design objectives – other students are likely also struggling! For this reason, you are not graded on an absolute points basis. Do your best!

Notes:

- *Read and understand the provided material including source code, scripts, testcases, constraints, and design flow as soon as possible.*
- *You will find it useful to refer to the OA tutorial from class and the official OA documentation.*
- *It is recommended that you adopt version control such as Git for collaborating with your teammate. If you are not comfortable with Git, you can read Mark's tutorial at https://markgottscho.wordpress.com/2015/01/24/a-brief-git-tutorial-for-collaborative-research/ . We also recommend beginners start with www.gitimmersion.com .*
- *Please use Piazza for asking (and answering) questions before coming to office hours.*
- *Since you are expected to work on the assignment in teams of two, do not provide solutions or compare results with other teams.*

Design Objectives



The Pin Assignment tool should accomplish the following:

- For each type of macro/block in a given testcase, determine a pin assignment solution such that the inter-macro wirelength is minimized. This is ***hierarchical pin placement*** since all instances of a particular macro should have identical pin assignment.
- You are allowed to add *at most one* extra physical copy of a pin around the periphery of the macro. This is ***multi-pin placement.*** Since hard macros are often integrated on passive substrates, once the signal leaves the macro boundary, there is no opportunity of buffering the signals. In case, only one pin location is allowed, some nets might be too long, and this would result in the signals dying down before they reach the destination macro. Multi-pin strategy allows such nets to be internally routed and buffered through the source macro and the nets would exit through the nearest-to-the destination pin locations.
    - Ensure that you maintain *hierarchical pin placement strategy*, i.e., the pin assignment for the instances of a macro should be same. You will finally notice that, top-level P&R would choose the best copy of the pin and in that case, different instances of a macro would have nets connected to

different copies of the same logical pin. You need not worry about that. The goal should be to ensure that all the instances of a macro get same physical locations of each pin before P&R.

- In case there is a constraint on maximum perturbation, for every pin, the Manhattan distance between the new and the old location must be within the amount of perturbation allowed. In case there are two copies of the pin, the total (or mean) perturbation should be minimized. For example, if a pin's original location is (x_1, y_1) and new location of two copies are (x_1^1, y_1^1) and (x_1^2, y_1^2), the total perturbation would be ($|x_1^1 - x_1| + |x_1^2 - x_1| + |y_1^1 - y_1| + |y_1^2 - y_1|$).
- The initial placement of the macros shouldn't be perturbed. However, you are allowed to rotate the macros without changing the location of the center of the macro. Note that, mirroring the block is not allowed.

These are the specific objectives for your Pin Assignment tool:

- Your tool should be robust in order to meet different sets of input rules across various test cases.
- The primary objective is to minimize wirelength after routing from Innovus based on your improved pin assignment. This means that your algorithm would need an effective routing/congestion prediction mechanism to evaluate different pin locations based on connectivity.
- You should minimize both maximum and mean net wirelength for improved performance.
- The macro pin placement must honor a specified minimum pitch (inter-pin distance). Pins must also be moved in discrete steps of the metal track spacing for the pin metal layer, as opposed to a continuous movement along the macro periphery. This will ensure DRC compliance. The initial input pin assignment will honor the defined minimum pitch.
- You can assume that all pins on all macros for a specific design will be on the same metal layer and thus have the same metal track spacing, specified in the input rules.
- Your algorithm should minimize total perturbation of pins from their default locations, since a standard flow would require an additional step to fix internal layouts of the sub-blocks if their pins have been externally moved.
- Your algorithm should also add extra copies of a pin only if required since additional copies would mean additional internal routing and power consumption. Try to minimize the number of additional pin copies.
- In the provided macros, you cannot touch internal layouts other than to alter pin locations. The system-level routing will use specific higher metal layers using an automated Innovus routine, whereas the macros will have internal routing in lower layers. You cannot move system pins at the top-level design boundary.
- The final routing result must meet basic layout versus schematic (LVS) and design rule checks (DRC) within Innovus. This means that your tool must not alter original connectivity, delete/rename essential pins, make pins inaccessible, place pins too close together to be routable, and so on.
- Your algorithm/heuristic should be well designed to have efficient runtime. Multiple iterations are permitted through the Innovus flow (maximum of 5) for an improved result, but this will have a hit on runtime.

Flow Inputs

This section describes some important inputs to the Pin Assignment tool-flow. You will be provided with some starter material and directory structure according to these specifications to help get started. The entire tool-flow will be automated using a push-button script relying on this directory and filename structure. Ensure that your tool works correctly in this exact flow – otherwise, your project cannot be graded.

The major inputs to the flow are described in Table 1 below:

Table 1. Input file type descriptions.

File Type	**Description**
System design DEF	Contains global floorplan, placed macros, placed system pins, and net connectivity (unrouted). No changes can be made here.
Block macro LEFs	Individual files for each unique macro used in the system. Contain entire physical layout information. Only editing pin locations is permitted.
Technology LEF (Nangate 45nm)	Describes physical information for various metal and via layers, and basic design rules such as metal track pitch and minimum same-net spacing.
Testcase-specific input rules	Described below.
Innovus Tcl Script	Used for an automated Innovus flow after improved pin assignment for complete routing, LVS/DRC checks, reports and output GDS generation. No changes can be made here.
System design SDC, macro Liberty models	Timing constraints and macro timing models necessary for the Innovus flow, not useful for pin assignment.

Test-case Specific Input Rules

Your pin assignment tool will be tested against a mix of benchmark designs, each with 3 different sets of input rules: minimum, random and maximum. Input parameters across the 3 rule classes would be design specific. A sample set of values is shown in Table 2 for demonstration.

Table 2. Sample set of Input rules.

Rule Name	**Min**	**Rand**	**Max**
Macro Pin Layer	5	5	5
Minimum System Routing Layer	6	6	6
Maximum System Routing Layer	9	? (Eg 8)	7
Macro Pin Movement Step Size (microns)	0.28	0.28	0.28
Minimum Macro Pin Pitch (centre-to-centre) (microns)	2	? (Eg 6)	10
Maximum Macro Pin Perturbation (Manhattan) (microns)	∞ (Inf)	? (Eg 100)	50

- Macro pin layer will remain constant for a design across all rule classes. Macro pin movement step size will be equal to the metal track spacing defined for the pin layer in the technology library and will not change across rule classes either.
- Minimum routing layer will remain constant for a design across all rule classes, and will always be above the internal routing layers inside the macro.
- Maximum pin perturbation shall be ∞ for minimum rules across all designs, indicating you can move them to any location on the boundary.
- Minimum and maximum rules will remain the same as released values. There will be a sample set of Random rule values provided, but these will differ during evaluation (within the min-max range).

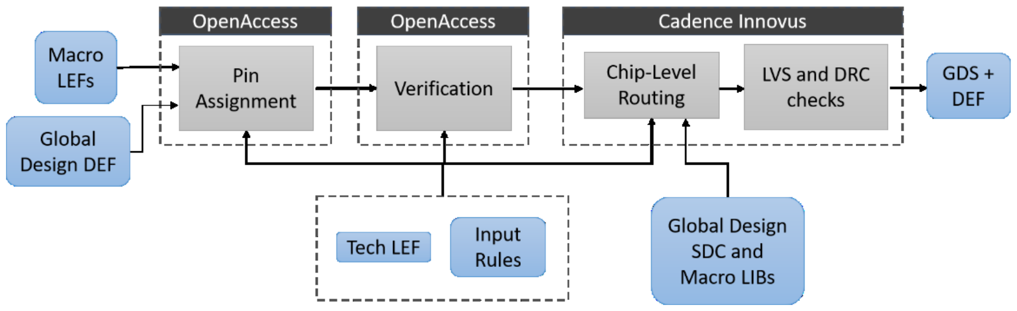
The input rule text file will be in the following form, separated by spaces, on the first line:

```
<MACRO_PIN_LAYER> <MIN_SYST_ROUT_LAYER> <MAX_SYST_ROUT_LAYER> <MACRO_PIN_MOVE_STEP>
<MIN_MACRO_PIN_PITCH> <MAX_MACRO_PIN_PERTURB>
```

A sample input rule file that corresponds to the minimum rule values in Table 2 would be:

```
5 6 9 2.8 2 Inf
```

Tool Flow



- Each benchmark design will consist of a global design DEF and SDC, LEFs and LIBs for each unique macro in the design, and 3 sets of input rules.
- The global design DEF, macro LEFs and technology LEF will be translated to an OA database.
- Pin assignment will be run in OpenAccess, where only moving macro pins would be permitted.
- The assignment output will be run through a provided OA verification routine to ensure legality of the design according to given constraints.
- The OA database will be imported into Innovus for routing. The global design SDC and macro LIBs will also be supplied here (unchanged).
- The routed output will be checked for DRC and LVS within Innovus. The final design outputs will be in DEF and GDS.
- You are permitted to run the Pin Assignment – Innovus flow iteratively for an improved output, with a maximum of 5 runs. Please read the project material README for more information.

Performance Metrics

Your Pin Assignment tool will be evaluated on several different metrics described in this section. A significant part of your grade will be based on your team's relative rank on the project metric compared to your classmates.

There are a few initial designs provided in the starter material, and more will be added on in the coming weeks. A few other benchmarks will remain blind for final evaluation. Each benchmark (released + hidden) will be evaluated against the 3 sets of design rules (minimum, random, maximum).

The score for each testcase (a particular benchmark evaluated for a particular rule set) will be a weighted sum of the following parameters.

- w_{max} = max[0, 1 – (max net wirelength) / (max net wirelength without pin assignment)].
- w_{mn} = max[0, 1 – (wirelength mean) / (wirelength mean without pin assignment)].
- p = max[0, 1 – (mean pin perturbation) / { (mean macro perimeter) / 2 }).
  - *Perturbation is computed as the minimum Manhattan distance between the original and final pin locations along the macro boundary. In case there are two copies of a pin, the mean of the perturbations of the two copies of the pin from the original pin location would be considered.*
  - *Zero perturbation (i.e. no pin moved) will also result in p=0.*
- m = 1 – ((total number of pins - total # pins in original macro)/(total # of pins in original macro))

- e = max[0, 1- (your runtime/10 seconds)].
  - *Runtime will be measured using the Linux "time" command on a specific eeapps server.*
  - *Multiple iterations are permitted through the Innovus flow for improving your result, with a maximum of 5 runs. The runtime for your OA source code will be summed up across all iterations.*
  - *The runtime counted for this metric will be the total pin assignment runtime excluding time taken for Innovus runs. If your run looks like: PinAssign1 -> Innovus1 -> TextParse1 -> PinAssign2 -> Innovus2, then the runtime would be time(PinAssign1) + time(TextParse1) + time(PinAssign2).*

Each evaluation also carries the following verification metrics:

- a = 1 if the new positions of the blocks are same as the old positions, 0 otherwise (Rotation allowed).
- b = 1 if hierarchical pin placement is obtained, 0 otherwise.
- c = 1 if the routed output passes Innovus DRC and LVS, 0 otherwise.
- p_{min} = 1 if all macro pins satisfy the minimum pitch constraint, 0 otherwise.
- p_{max} = 1 if all macro pins satisfy the maximum perturbation constraint, 0 otherwise.
- d = 1 if all the pins have at most two copies, 0 otherwise.

The total score s for each testcase will then be computed as:

$$s = a*b*c*d*[\ p_{min} *p_{max}\ (\ 1*w_{max} + 2*w_{mn}\ ) + p_{max}(\ 2*p\ + 2*m\ ) + 3*e\ ]$$

The total pin assignment score will then be given by:

$$P = \text{sum}(s \text{ over all testcases})$$

The final project metric score will be given by:

$$S = P + r \qquad \text{where}$$

- r = 10 if your code is well organized and commented (easy to read and understand); 0 otherwise

Thus, the maximum theoretically achievable score is achieved with the Innovus output passing DRC and LVS, satisfying minimum pin pitch and maximum pin perturbation constraints, having 0 mean and 0 max wirelength, negligible pin perturbation, and instantaneous runtime. Thus, total testcase score s is out of 10. Regardless of your tool's performance, you can get points for code readability and organization (up to 10 points). Thus, the total possible points are out of (10 * #testcases) + 10.

Your absolute numeric score is not the determining factor. We will use the relative differences between teams' scores to determine your rank and final project grade. The competition part of the project constitutes 10% of your overall course grade.

Potential Updates

The project description and released material are subject to updates over the coming weeks:

- An OA verification script will be provided to ensure legality of the design.
- OA and python scripts will be provided for automated evaluation of all performance metrics.
- More benchmark designs will be added beyond the initial few. You should expect them to get significantly more complicated than the initial simple testcases and should thus keep your algorithm adaptive.
- A few bonus capabilities might be added late in the quarter for extra credit. These could potentially include enabling system pin relocation, and feedthrough pins within macros.

Midterm Status Report

During the 8th week, we will like an informal status report in the form of a PDF uploaded to CCLE. This should briefly and concisely state at what stage you are in the project, what your approach is, and any problems you are having. You should also list things that you believe you can resolve by the submission time. The report should be of 1 page (including References) in the standard 2-column IEEE Transactions format. This report is worth 10% of your overall course grade.

Final Presentation

After you have submitted your final code, you will present your approach and TA-verified results in a short 7+3 minute presentation during Week 10. This should demonstrate your high-level flow, key design choices with reasoning, and a table of all (TA-verified) results. You should prepare no more than 10 slides. The presentation is worth 20% of your overall course grade.

Project Submission

You should submit your tool's complete source code (the `src/` subdirectory only) based on the provided material directory structure, along with your modifications of specific scripts where customization was permitted. Do not include benchmark test cases, temporary files, compiled object files, etc. A complete submission should thus contain:

- `src/` subdirectory
- Customized `run_pinassign_and_innovus.py`
  - Even if you use the default version, include it in your submission.
- Customized SConstruct
  - While it is preferable and entirely possible to compile everything within `scons`, you may include extra Makefiles for compiling external tools if you absolutely have to.
  - In the worst case, your entire build should be compile-able with <u>one</u> `scons` + <u>one</u> `make` run on the eeapps terminal, without any errors.
- README file to build and run.
  - Carefully list any considerations the TA will need to get your code working.

<u>Be absolutely sure that you can run the complete flow using the provided run_all.py script without problems</u>. Otherwise, your submission cannot be graded!

SUBMISSION INSTRUCTIONS (Read carefully)

- All necessary code must be tarballed and submitted as a single archive file on `eeapps.seas.ucla.edu`. Presentation slides (PowerPoint or PDF format) and midterm progress report (PDF) should be posted on CCLE by the respective deadlines.
  - For final code submission, create a directory named as follows: `GroupID_Lastname1-Firstname1_UID1_username1_Lastname2-Firstname2_UID2_username2_Project/`. For example: `Group1_Bruin_Joe_123456789_joe_Bruin_Josie_987654321_josie_Project/`
    - Include all files mentioned in the Project Submission section.

    - Compress and archive this directory using tar/gzip to have a single submission file named `GroupID_Lastname1-Firstname1_UID1_username1_Lastname2-Firstname2_UID2_username2_Project_pinXXXX.tar.gz` (Make up a 4-digit numeric PIN of your choice to substitute for `xxxx`). For example: `Group1_Bruin_Joe_123456789_joe_Bruin_Josie_987654321_josie_Project_pin1234.tar.gz`
    - Submit tarball by copying it to `/w/class/ee201a/ee201ata/submission/project/`
    - IMPORTANT: Make sure all files you submit, as well as the tarball, have full read and execute permissions to group and others or we may not be able to grade your lab. Do this using `chmod -R go+rx FILE_OR_DIRECTORY`
- **Late submissions will not be accepted** (write/execute permissions to the submission directory will be removed). 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.
- 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.2/ee/ee201a/ee201ota/submission/project` 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.
- **Test your submission before the deadline!**