CSCE4013/5013 Project Description

General Instructions

For all projects, you need to implement an algorithm and test it with provided benchmarks. Two benchmarks are relatively easy to handle, and at least one benchmark is difficult. Your program needs to run on at least two provided benchmarks to secure a B for the project. Also, lab2 notes about allowed programming packages also apply to all projects. You should properly add comments and format the code to improve its readability. A standard naming format such as camel case is highly recommended.

Each team needs to submit a report in IEEE format with the following sections:
Introduction and Problem Formulation: Briefly discuss the CAD problem and your motivation
Methodology and Algorithm: Briefly introduce the idea behind the algorithm and include a time complexity analysis
Implementation: Detailed description of how your program implements the algorithm. Discuss several main functions used in your program and any optimizations you did to improve the runtime or memory footprint.
Experimental Results: Discuss the results obtained using the provided benchmarks. You can generate more benchmarks yourself and include them as well. You should at least show the performance (runtime, memory, etc) and quality of the solution you obtained (such as wirelength, area utilization rate, cutsize, etc).
Conclusion: You should summarize the project and what you learned from it in a short paragraph.

You can download the IEEE template from
https://www.ieee.org/conferences/publishing/templates.html

For two-member teams, you should also discuss contributions from each team members in the introduction. For graduate student, you need to discuss how visualization is implemented in your code, what GUI packages you used, etc. In this report, you should also include instructions of how to run your program in the implementation section.

At least a 2-page report is required for single-member teams, and a 3-page report is required for two-member teams. During the presentation, if your team has a graduate student, you are required to demonstrate the visualization results, and others need to demonstrate the command line interface. Package these files along with all of your source code and submit it on the course website.

Your final grade is calculated based on the following criteria:
Your packaged source code (10%)
A 2-3 page scientific report in IEEE format (10%)
Present and demonstrate your tool in the final week (10%)
Each team will receive the same score for all its members. Extra credit (up to 5%) may be given if extensions beyond the basic requirement are successfully implemented.

Final testing must be performed on the VLSI server. It provides a common platform to compare the runtime across different teams, and it has a much large memory space for large circuits. Again, start early. Running and testing your programs on large benchmarks can last 10+ hours if your algorithm is not efficient. You may even crash the server if your program has memory leakage issues (common for C implementations). So, anything can happen before the deadline.

The following instructions are specific to each project topic.

FM Partition

This project needs to implement the FM partition algorithm discussed in the partition chapter. The basic algorithm with bucket structure is required. The implementation of the algorithm must have a complexity of O(NlogN).

Instructions:

The detailed instructions for FM partition are mostly the same as lab2. Refer to lab2 instructions for input format, output format, input arguments, on-screen information and termination criteria.

The differences are:

FM needs to perform partition directly on a hypergraph.

Selecting best candidate needs to be performed in O(1) time.

In addition to those arguments mentioned in lab2, there is another argument (-r) to the program: area ratio X. X is the size of the left partition over the total size. Assume all cells have a unit area. E.g., if there are 10K cells, and 4K of them are in the left partition, then X=0.4. In the report, you need to show results with X=0.5 and X=0.4.

In the project report, you must include a comparison with your KL algorithm in lab3, using X=0.5. Only p2 and structP are needed for this comparison. Discuss the impact on both runtime and cut size.

Extra credit: Implement a 3-way partitioning based on FM. You must compute and report the cutsize between any pair of partitions. You can add additional arguments to specify area ratio if necessary.

Polish Expression Evaluation

This project needs to implement the floorplan evaluation using polish expression. This includes 1. Normalized PE validation checking, 2. A full evaluation of a new PE. The implementation must satisfy the time complexity requirement: 1. O(N), 2. O(NlogN). Each evaluation should compute the location of all modules and the size of the chip.

Your program should accept two input files as arguments. The first one in size format indicates the dimensions of all modules. The format is:

****

<width> <height>

…

****

The line number is the ID of that module.

The second argument in pe format contains the polish expression you need to evaluate:

****

<initial PE>

-<incremental PE after move 1 or 2>

+<incremental PE after move 3>

…

****

The first line indicates the initial PE. This PE is always valid but you need to perform a full evaluation. Starting from the second line, each line indicates an updated PE after a move. Lines starting with a minus sign are created after move 1 or 2, and no validation check is required. You need to perform an incremental evaluation of those moves. Lines starting with a plus sign indicates a move 3. You must perform the validation check on those moves. If the move is valid, perform floorplan evaluation and move on. Otherwise, the program should stop and reports the first location where the balloting property is violated.

For each floorplan evaluation, your program should generate the output as:

****

<width> <height> <module locations>

…

****

Each line corresponds to the same line in the PE file. If the move is invalid, leave the line as blank. <module locations> are specified as <x coordinates, y coordinates>. For example, if there are three modules of size 1x2, 1x1, 3x1, the output file may look like:

****

4 3 (0,1) (3,0) (0,0)

5 2 (0,0) (4,0) (1,0)

****

The location should be the lower-left corner of the allowed space for each module.

Extra credit: Incremental evaluation after a move (type 1, 2, 3) is performed on a PE. The time complexity should be O(logN). Instead of another full evaluation, the incremental evaluation should only compute the paths that need updates. You should compare the runtime comparison without or without incremental updates.

Sequence Pair Evaluation

This project needs to implement the floorplan evaluation using sequence pair. This includes 1. Constraint graph construction, 2. Redundant edge removal, 3. Location evaluation of SP. The implementation must satisfy the time complexity requirement: 1. $O(N^2)$, 2. O(N), 3. O(N). Each evaluation should compute the location of all modules and the size of the chip.

Your program should accept two input files as arguments. The first one in size format is the same as the one used in PE evaluation. See above. The second one in sp format is defined as:

****

<initial positive SP> <initial negative SP>
<incremental positive SP> <incremental negative SP>
…

****

The first line is the initial SP you need to evaluate. Starting from the second line, each SP updated based on a move of the previous SP. The move may swap two modules in one or both positive and negative sequence pairs. If one of the sequences is not changed, it is indicated by a dash.

The output requirement is the same as in PE. Since there is no validation checking, you should always generate an output for each line in the sp file.

Extra credit: Implement an incremental evaluation which only updates the affected modules. Also, identify a sub-graph in the constraint graph and only reevaluating the changed portion. Compare the runtime in your report.

ILP Floorplanning

This project needs to implement the ILP floorplanning with rotatable modules and soft modules. For soft modules, overestimation method is used, as in homework 3.

Your program should accept the input file name (in fp format) as the first argument.
The format of this file is:

```
*************************************************************
*HARDBLOCKS
<module> <width> <height>
*END

*ROTATABLEBLOCKS
<module> <width> <height>
*END

*SOFTBLOCKS
<module> <area> <aspect_lower_limit> <aspect_higher_limit>
*END
*************************************************************
```

Module names are unique. The aspect ratio is defined as the width over height.

You can choose to use any LPsolvers available on the VLSI server. Generally, matlab, glpk and lpsolve are provided. Your program should calculate the floorplan size and the area utilization. Print this information on the screen. You can ignore the soft blocks in the basic implementation.

Your program should generate an output file containing the floorplan results:

```
*************************************************************
*HARDBLOCKS
<module> <xlocation> <ylocation>
*END

*ROTATABLEBLOCKS
<module> <xlocation> <ylocation> <width> <height>
*END

*SOFTBLOCKS
<module> <xlocation> <ylocation> <width> <height>
*END
*************************************************************
```

XY locations are from the lower left point of the module.

Extra credit: Implement the soft block handling. Your program should implement both under-estimation and over-estimation approximations to handle these soft blocks. You can add another program argument (-m) to switch between these methods. Further, you should compare the results using different approximation methods in the report.

Steiner Routing

Implement the basic algorithm with the naïve search of the Steiner points. An example is provided in the supplement reading material. For GUI, show the routing trees for individual nets as well as the combined ones that show the overall routing graph usage.

A single input argument is needed for the filename. File format is:
****
1st line: <Grid size> <No. of nets>
All other lines: <net name> [<points> …]
Each subsequent line is one net. The first field is the unique net name, followed by numbers in parenthesis () which are coordinates of the points (cells) in that net.
The first node in each net is the source node.

****

E.g., 10_30.nets: The grid size is 10x10, and there are 30 nets to be routed. The first point in each net is the source node for that net.

Your program should calculate the Manhattan wirelength for each net, and compute the total wirelength. Print this information on screen. Your program should also generate one output file showing the routing result. The output file format is:
****
1st line: <Grid size> <No. of nets>
All other lines: <net name> [<wire segments> …]
The wire segment has four coordinates, (x1, y1), (x2, y2). The segment is from (x1,y1) to (x2,y2). E.g, net N1 which is from (1,1) through (2,2) to (3,2) should be written as:
N1 [(1,1),(2,2) (2,2),(3,2)]
****

Simulated Annealing Based Non-Slicing Floorplanning

Implement the basic algorithm and produce floorplanning results on all of the benchmark circuits. Some additional information is provided for you to calculate wirelength. But you are only required to minimize the floorplan area. If you consider the wirelength in your algorithm, it can be considered an extension to the basic algorithm, but only if you finished the basic algorithm first. No extension will be considered if your basic algorithm with only area considerations does not work. For GUI, show the annealing progress using floorplan snapshots at various temperature. (like the example slide in floorplan part2)

The first input argument should be the path of the input file in the fp format:
****
*HARDBLOCKS
<module> <width> <height>
*END

*TERMINALS
<terminal name> <x-coordinate> <y-coordinate>
*END

*NETS
- <net name> <number of connections (say N)>
connection_1
connection_2
....
...
..
connection_N
*END
****

You are to ignore all connections to the IO terminals in your wirelength calculation.

Your program should generate one output file showing the floorplan results:
****
*HARDBLOCKS
<module> <xlocation> <ylocation>
*END
****
XY locations are from the lower left point of the module.

You are required to perform parameter tuning with the SA algorithm to achieve the best quality result. You should at least implement the move based on swapping modules in the sequence. You should try with different initial temperatures, cooling speed, and the possibility of accepting negative moves. You should also set a reasonable termination temperature based on the parameters you choose. You need to explain the strategy you used to select these parameters in the report and explain their impacts on runtime and result quality using some examples. Note that the final aspect ratio of the floorplan should be close to 1. Results with an aspect ratio larger than 2 or a utilization lower than 0.75 are considered to have bad quality. Results with an aspect ratio higher than 2.5 or with a utilization lower than 0.65 are not acceptable and will be treated as failing on the benchmark. If your program considers wirelength, you should also tune your optimization target function to balance area and wirelength. Your program should print the following information on screen: the floorplan size, aspect ratio, and total area utilization. Also, if your program calculates wirelength, print it as well.

Extra credit: 1. You can add additional custom moves to further improve quality and runtime. However, you must document those additions in your report and compare it against the basic

implementation. Extra credit may be considered only if a significant utilization improvement (10% improvement for utilization under 75% and 5% for utilization above 75%) is found and properly explained in the report. 2. Include wirelength in the optimization target. You can add a program argument (-w) as a weight factor to balance the area and wirelength targets. The default value with the weight factor to be zero indicates no wirelength consideration at all. You can use half-perimeter bounding box as the wirelength model. In the report, you must compare results with or without wirelength consideration.