ECE5960 Physical Design Algorithms Programming Assignment 3 Report

Cheng-Hsiang Chiu u1305418
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT-84112
{cheng-hsiang.chiu}@utah.edu

I. Introduction

In the assignmnet I implemented `minimum spanning tree` to connect all pins in a single net. I followed the idea in the webpage[1].

II. Encountered Challenge

The challenge I encountered was the long update time of the algorithm. I implmented Prim's Minimum Spanning Tree algorithm. The complexity of the algorithm is N^2. For a small graph, the runtime is acceptable. However, as the graph becomes much bigger, the runtime is exponentially increasing. Besides, every pin in the net can connect to any other pins. The graph is a fully-connected graph. Therefore, the biggest challenge is to reduce the runtime.

III. Way to Overcome Challenge

To overcome the challeng of long runtime, my idea was to only update the heighbor pins instead of all pins in the net. Because, pins that are located far from current node will be added to the minimun spanning tree in the future by its neightbor nodes. With the idea, I only updated the weights of the neighbor nodes and can reduce the runtime from 40 minutes to 1 second with respect to the biggest net.

IV. Availability

The whole implementation is available in the Github repository, link.

V. Build and Run Executable

There is a README in the uploaded compressed file and in the Github repository mentioned above. README provides very detailed information. In this section, I highlight the instructions to build and run the executable `st`. Please make sure CMake and clang++ is installed.

To build the executable `st`, use the following commands,

```
mkdir build
cd build
cmake ../
make
```

To run the executable with an input file, use the following command in folder `build`,

```
./st [input] [output]
```

Input	Wirelength	Runtime (second)
case1	323	0.003
case2	1249	0.001
case3	2708	0.001
case4	108654	0.002
case5	23387274	0.002
case6	405432649	0.004
case7	9426548476	0.008
case8	186116372386	0.015
case100000	1789631531172	0.143
case200000	3599278040539	0.282
case500000	90360690	0.61

TABLE I: Wirelength and runtime of each input.

In addition to running `st` with one input, I provide a script `run.sh` to run `st` and measure the runtime with all of the inputs. The script will be copied to folder `build` automatically when building `st`. In folder `build`, simply use the following commands,

```
chmod 744 run.sh
./run.sh
```

VI. Results

The initial positive and negative seuqnces were random sequences. The executable was built using clang++ v10.2 with -std=c++17 and -O3 on a Linux machine with Intel i7-9700K 8 Cores at 3.6 GHz and 32 GB RA. All results were obtained in a single run.

Table I shows the wirelength and runtime for each input.

References

[1] A. Kahng and I. Mandoiu. Rectilinear minimum spanning tree algorithms. [Online]. Available: https://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/RMST/