CS 516000 FPGA Architecture & CAD
Final Project (Due: Dec. 30, 2023)

This project is completely open in terms of how you solve the problem. (You may propose your own approach or follow previously proposed approaches in the literature. You may make use of libraries for combinatorial optimization in the public domain, if desired, but you have to properly cite the source(s).)
You may work individually or in a team of two people, you will receive some bonus points if you finish this project on your own.

Problem Description

Given an FPGA architecture *Arch* and a global placement solution of a netlist (V, E) on it where V represents the set of instances and E represents the set of connections between the instances. We want to perform legalization and detailed placement to find a final legal placement solution on *Arch* that minimizes the total HPWL of the nets.

Resource blocks have three different types, CLB, RAM, and DSP. Instances have four different types, IO, CLB, RAM, and DSP. We use the HPWL model to estimate the wirelength of a multi-pin net assuming each pin is located at the center of the corresponding instance.

There are several constraints the final solution must satisfy.

1. Each non-IO instance should be placed at a resource block of the matching type and the center of instance should be at the center of the resource block.
2. Each resource block can only accommodate one instance of the matching type.
3. All IO instances are fixed, which means that they are not movable.

Input Format

There will be three input files for each testcase which are the FPGA architecture file, the instance file, and the net file.

Each line in the FPGA architecture file lists the name, type, and center coordinate of a resource in the architecture.

Format:
<**resource name**> <**resource type**> <**resource center x coordinate**> < **resource center y coordinate** >

Ex:
RESOURCE1 CLB 1.5 0.5
RESOURCE2 CLB 1.5 1.5
RESOURCE3 CLB 1.5 2.5
RESOURCE4 CLB 1.5 3.5
RESOURCE5 CLB 1.5 4.5
RESOURCE6 CLB 1.5 5.5
RESOURCE7 RAM 2.5 1.0
RESOURCE8 RAM 2.5 3.0

RESOURCE9 RAM 2.5 5.0
RESOURCE10 DSP 3.5 1.0
RESOURCE11 DSP 3.5 3.0
RESOURCE12 DSP 3.5 5.0

Each line in the instance file lists the name, type, and initial center coordinate of an instance.

Format:
<**instance name**> < **instance type**> < **instance initial center x coordinate**> < **instance initial center y coordinate**>

Ex:
INST1 IO 0.5 1.5
INST2 IO 0.5 4.5
INST3 CLB 1.75 1.85
INST4 CLB 2.2 3.5
INST5 RAM 3.0 2.0
INST6 DSP 3.15 3.2

Each line in the net file lists the name of a net and the name of instances connected by this net.

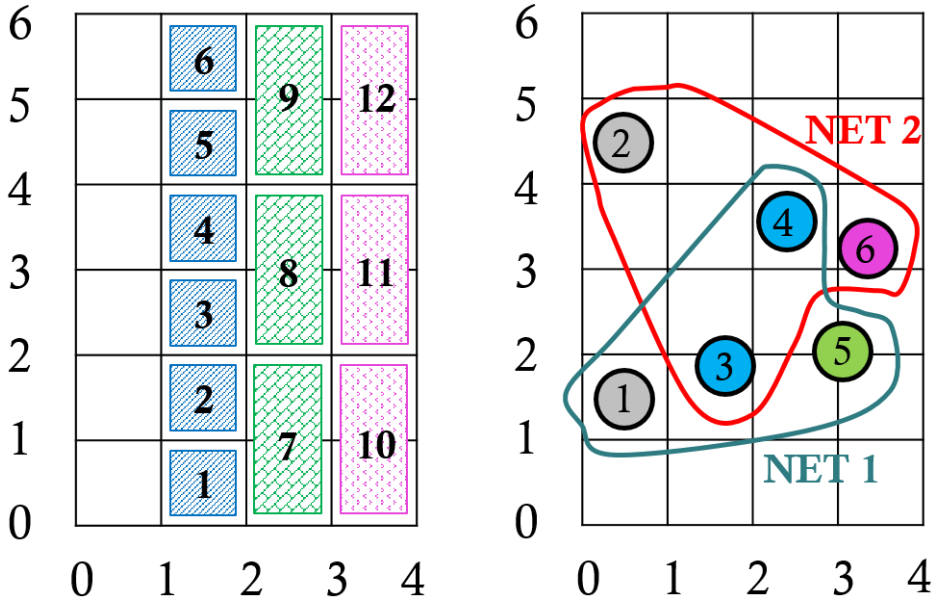
Format:
<**net name**> < **instance name**> < **instance name**> < **instance name**> …

Ex:
NET1 INST1 INST3 INST4 INST5
NET2 INST2 INST3 INST4 INST6

The following is an input example.

First file	RESOURCE1 CLB 1.5 0.5 RESOURCE2 CLB 1.5 1.5 RESOURCE3 CLB 1.5 2.5 RESOURCE4 CLB 1.5 3.5 RESOURCE5 CLB 1.5 4.5 RESOURCE6 CLB 1.5 5.5 RESOURCE7 RAM 2.5 1.0 RESOURCE8 RAM 2.5 3.0 RESOURCE9 RAM 2.5 5.0 RESOURCE10 DSP 3.5 1.0 RESOURCE11 DSP 3.5 3.0 RESOURCE12 DSP 3.5 5.0
Second file	INST1 IO 0.5 1.5 INST2 IO 0.5 4.5 INST3 CLB 1.75 1.85 INST4 CLB 2.2 3.5 INST5 RAM 3.0 2.0 INST6 DSP 3.15 3.2
Third file	NET1 INST1 INST3 INST4 INST5 NET2 INST2 INST3 INST4 INST6

The above input corresponds to the following architecture and global placement solution of a netlist.



Architecture Instances and Nets

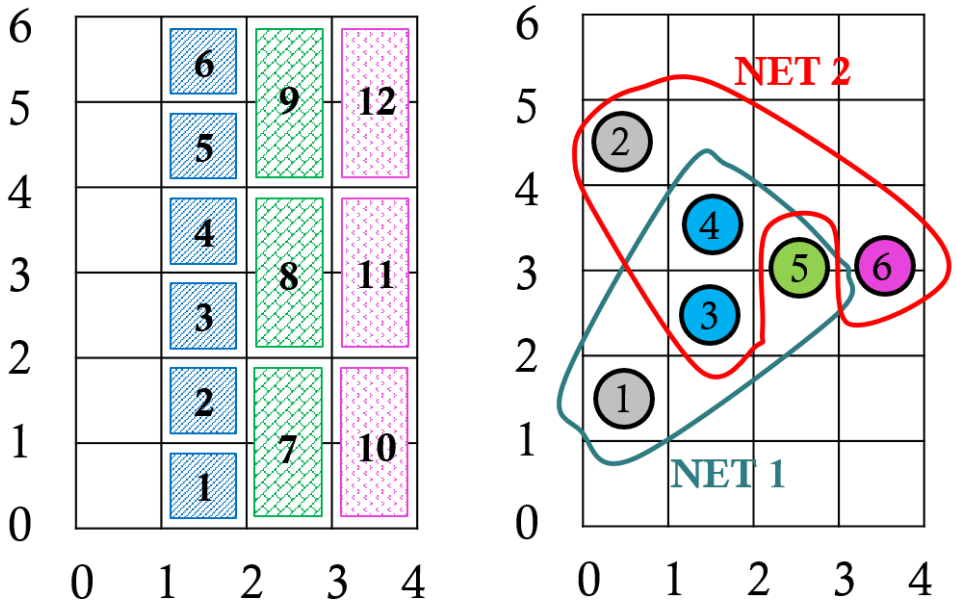
Output format

The number of lines in the output file should be equal to the number of non-IO instances, and each line lists the name of an non-IO instance followed by the name of the resource it is placed at.

Format:
<**instance name**> < **resource name**>

Ex:
INST3 RESOURCE3
INST4 RESOURCE4
INST5 RESOURCE8
INST6 RESOURCE11

The corresponding final legalized placement solution is shown in the right figure below.



Architecture Instances and Nets

Evaluation

The total HPWL of testcases would be calculated for grading. For instance, in the example above, the HPWL of NET1 is 4 and the HPWL of NET2 is 5, so the total HPWL in this example is 9.

For each testcase, if the output result violates any constraint mentioned in the first section, or the runtime of your program exceeds 10 minutes, it fails on this testcase.

Any plagiarism will result in a 0 grade for the project.

Submission

The source codes should be uploaded to eeclass. Please include a Makefile for compiling your codes. The codes are required to be written in C/C++, and be compiled on a Linux platform.

In addition, upload a report describing the details of your approach. If it is a team work, each member has to explain what he has done and the percentage of his contribution.

Name the executable file "legalizer" and make sure your program can be executed by running the command:

$ ./legalizer <first input file path> <second input file path> <third input file path> <output file path>

For example,

./legalizer ./input/testcase1/architecture.txt ./input/testcase1/instance.txt ./input/testcase1/netlist.txt ./output/output1.txt

All codes should be compiled by running the command "make". You don't need to submit the executable code, and the command "make clean" should delete all the files generated by the command "make".

Grading

1. The completeness of your program and report (20%)
2. The solution quality, including hidden testcases (80%)

The quality score is based on the total HPWL of your solution compared to other students when your solution is valid (quality score of different testcases would be calculated independently). Here is the equation for score calculation:

$$100 - 35 \times \min(\mathrm{F}_1, \mathrm{F}_2)$$

where

$$\mathrm{F}_1 = \frac{Your\ total\ HPWL - The\ smallest\ total\ HPWL}{The\ largest\ total\ HPWL - The\ smallest\ total\ HPWL}$$

and

$$\mathrm{F}_2 = \frac{Your\ total\ HPWL - The\ smallest\ total\ HPWL}{The\ smallest\ total\ HPWL}$$