109006240 劉其生, 110006267 張嘉俽

CS 516000 FPGA Architecture & CAD
Final Project Report

I. Contribution

For the contribution we do the code together.
109006240 劉其生 50% : Class cc, setoutput(), Legalization()
110006267 張嘉俽 50% : OpenFile(), toClass(), Legalization()

II. Introduction

In this assignment, we are required to make a program to do a legalization followed by a detailed placement given the inputs of global placement. The metrics of the performance of our programs is the HPWL equation which can be obtained through this equation.

$$HPWL_e = \max_{i \in e}\{x_i\} - \min_{i \in e}\{x_i\} + \max_{i \in e}\{y_i\} - \min_{i \in e}\{y_i\}$$

The sum of each of the netlists HPWL will then be the metric of our programs.

$$HPWL = \sum_{e \in E} HPWL_e$$

III. Brief Approach

Our main goal is to legalize and do detailed placement on the given resources and instances that have been placed globally. To do this first we need to pre-process the input into a data structure to help us manage the instances and resources and we choose to create custom classes for each instance and resources. These classes are then put into their respective vectors. First we wanted to handle the X because we need to put each instance on the correct block and since the block types are arranged in a column in which each column contains one type of block, we can then arrange the Y by simply stacking them. Since our first approach is to stack them, our first instance is going to be placed at the bottom. To tackle this issue, we added a sort function with input being the instances' X value to help with the iteration. The next step is to sort them by the Y value to ensure that the blocks that's supposed to be placed on the bottom stay on the bottom and instances the one supposed to be placed on the top stays on the top.

IV. Code Explanation

In this section, we are going to give a more detailed explanation behind the idea we described in the previous section. Our code can be divided into 2 big parts:

1. Storing Data

Before we implement our algorithm on the global placement, we need to store the information provided in this assignment. After discussing and trial and errors, we decided that a simple data structure will suffice. Since there are 3 files of data, each of these files is stored into:

i. Architecture -> a vector of class cc
ii. Instance-> a vector of class cc
iii. Netlist -> a vector of set of string

The cc class is made up of name, type, coordinate x & y, and a flag. As an instance of architecture, the flag is used to indicate whether the resource has been filled by an instance. On the other hand, the flag is not used in the Instances' instance.

The class cc has a few methods to ease our way into maintaining the program, such as print instance (*pa*), change coordinate (*ccoords*).

The netlist consists of strings of names. The vector on the outer layer indicates the group of netlists, the set indicating the individual netlist. We chose set because we hope to utilize the set function, namely *find*. However, in the later part it is not used since we don't really use netlist in our legalization process.

2. Legalization

After storing the data into the corresponding structures, we then start the legalization process. Before we go into the legalization function, we sort our instances based on the x-coordinate, then the y-coordinate. The purpose of this step has to do with the way we legalize the instances and will be explained further later.

Since the architecture of the FPGA is identical for all the testcases, we first make a string array(*ct*) indicating the type of resource in a particular x-coordinate. Since the left-most bottom x-coordinate of a resource is always an integer, we can use the index of the array to symbolize the left-most bottom x-coordinate of a column in the FPGA board.

As mentioned in the previous section, we're going to stack our instances from the bottom and rise vertically, therefore, we need another array(*bott*) indicating the number of instances that has already been stacked in the particular column. Similar as before, the index of the array will be used to indicate the position of the wanted row.

With these structures, we can continue with the legalization process. We will iterate through the vectors of instances based on the type of the current iterator, we will move the x to the left by 1 coordinate until it meets the matched type of resources. If x reaches 0, we will then move to the other side (right side). When we find the suitable value of x, we will then stack the resources based on the value stored in *bott*. We will then update the value of *bott* regularly after every iteration.

We also output the name of the instances, along with the name of the resource which has the same legal middle coordinates.

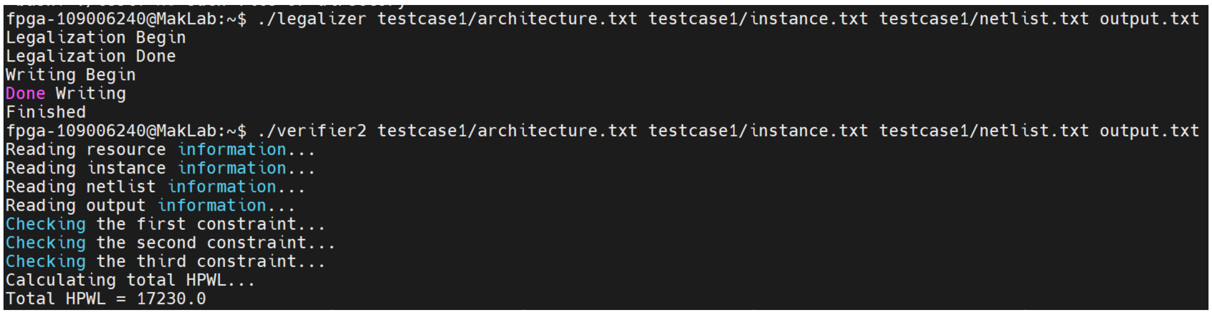
V. Result

No	Testcase	Version 1	Version 2 (Submitted)
1.	T1	20 176.0	17 230.0
2.	T2	18 202 806.0	13 690 267.0
3.	T3	275 802.5	218 861.5
4.	T4	78 193 940.0	80 188 381.5

109006240 劉其生, 110006267 張嘉俶

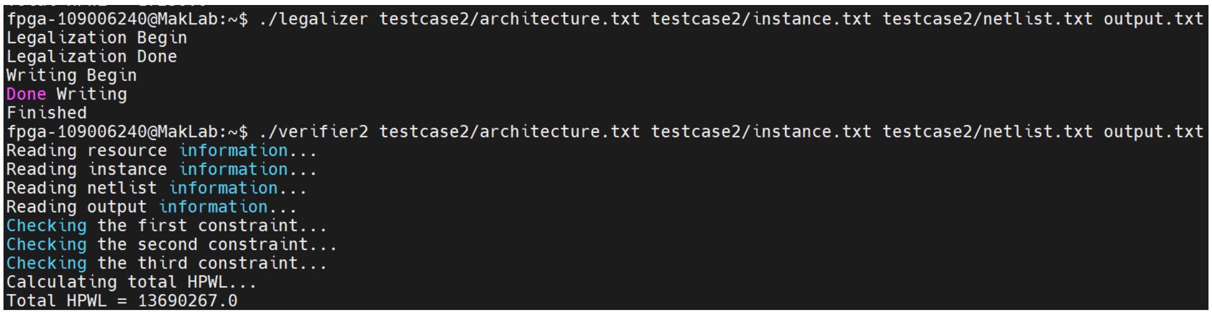
	TOTAL	**96 692 724.5**	**94 114 740.0**

VI. Attachment



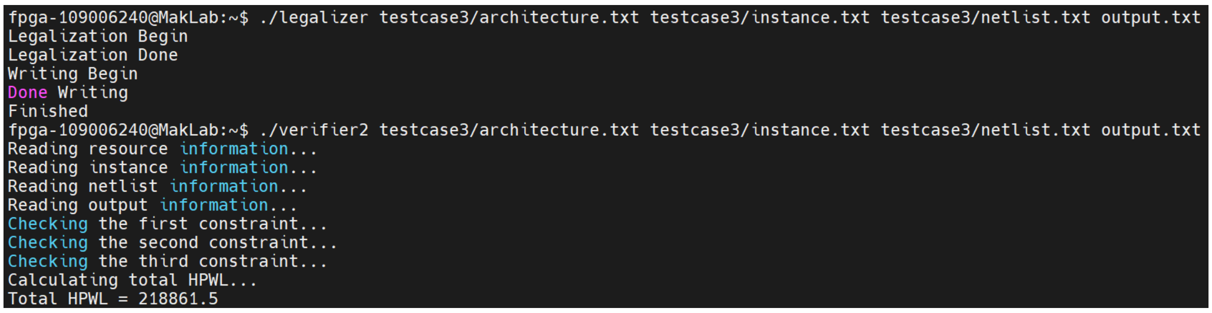
```
fpga-109006240@MakLab:~$ ./legalizer testcase1/architecture.txt testcase1/instance.txt testcase1/netlist.txt output.txt
Legalization Begin
Legalization Done
Writing Begin
Done Writing
Finished
fpga-109006240@MakLab:~$ ./verifier2 testcase1/architecture.txt testcase1/instance.txt testcase1/netlist.txt output.txt
Reading resource information...
Reading instance information...
Reading netlist information...
Reading output information...
Checking the first constraint...
Checking the second constraint...
Checking the third constraint...
Calculating total HPWL...
Total HPWL = 17230.0
```

Picture 1. Testcase1



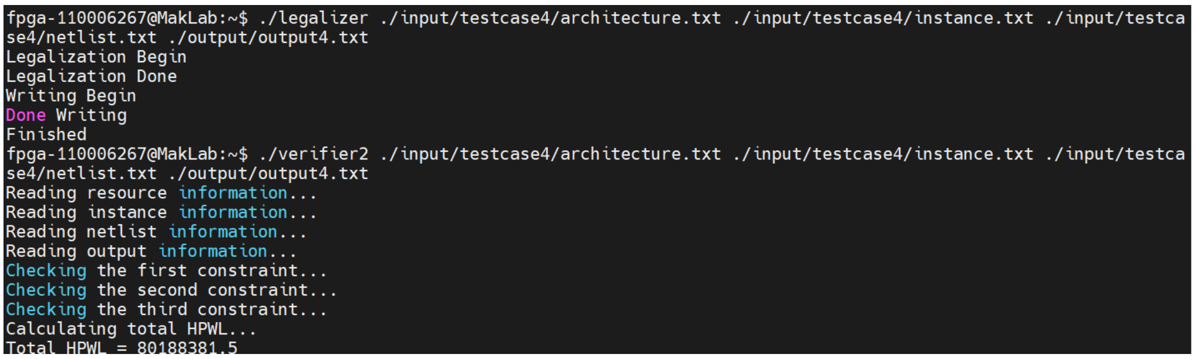
```
fpga-109006240@MakLab:~$ ./legalizer testcase2/architecture.txt testcase2/instance.txt testcase2/netlist.txt output.txt
Legalization Begin
Legalization Done
Writing Begin
Done Writing
Finished
fpga-109006240@MakLab:~$ ./verifier2 testcase2/architecture.txt testcase2/instance.txt testcase2/netlist.txt output.txt
Reading resource information...
Reading instance information...
Reading netlist information...
Reading output information...
Checking the first constraint...
Checking the second constraint...
Checking the third constraint...
Calculating total HPWL...
Total HPWL = 13690267.0
```

Picture 2. Testcase2



```
fpga-109006240@MakLab:~$ ./legalizer testcase3/architecture.txt testcase3/instance.txt testcase3/netlist.txt output.txt
Legalization Begin
Legalization Done
Writing Begin
Done Writing
Finished
fpga-109006240@MakLab:~$ ./verifier2 testcase3/architecture.txt testcase3/instance.txt testcase3/netlist.txt output.txt
Reading resource information...
Reading instance information...
Reading netlist information...
Reading output information...
Checking the first constraint...
Checking the second constraint...
Checking the third constraint...
Calculating total HPWL...
Total HPWL = 218861.5
```

Picture 3. Testcase3



```
fpga-110006267@MakLab:~$ ./legalizer ./input/testcase4/architecture.txt ./input/testcase4/instance.txt ./input/testca
se4/netlist.txt ./output/output4.txt
Legalization Begin
Legalization Done
Writing Begin
Done Writing
Finished
fpga-110006267@MakLab:~$ ./verifier2 ./input/testcase4/architecture.txt ./input/testcase4/instance.txt ./input/testca
se4/netlist.txt ./output/output4.txt
Reading resource information...
Reading instance information...
Reading netlist information...
Reading output information...
Checking the first constraint...
Checking the second constraint...
Checking the third constraint...
Calculating total HPWL...
Total HPWL = 80188381.5
```

Picture 4. Testcase4