CS 5160-00 FPGA Architecture & CAD Fall 2020

(Due: Jan. 9, 2021)

1. Introduction

In this project, you are asked to **write a C/C++ program** to solve the placement problem for LUT-based FPGA after technology mapping. The goal is to place all instances (i.e., LUTs and 1-bit flip-flops) of a netlist onto CLB locations of a target FPGA with all PI/POs (i.e., primary inputs and primary outputs) already pre-placed so that the total wirelength is minimized. The wirelength of each net is estimated by its HPWL.

You are free to adapt any algorithm introduced in the class or reported in the literature or design your own algorithm. You may brainstorm good ideas with other students, but you must write your own code. You may also incorporate standard optimization tools (e.g. for weighted bipartite matching, network flow, integer linear programming, graph/hypergraph partitioning, etc.) from the public domain into your placement flow but you have to properly cite the source(s).

2. Assumptions

Figure 1 shows the locations of the CLBs and I/O pads. There are R rows and C columns of CLBs. We assume that the CLBs are evenly distributed and the distance between two adjacent CLBs is 1. The coordinate of the center of the lower left CLB is (1,1). The width and height of the FPGA are R+1 and C+1, respectively. The number of I/O pads on the left/right boundary of the CLB array is **P** where P may equal to **R**, **2R+1**, or **4R+3**. Similarly, the number of I/O pads on the top/bottom boundary of the CLB array is **Q** where **Q** may equal to **C**, **2C+1**, or **4C+3**. The distance between any two adjacent I/O pads are the same.

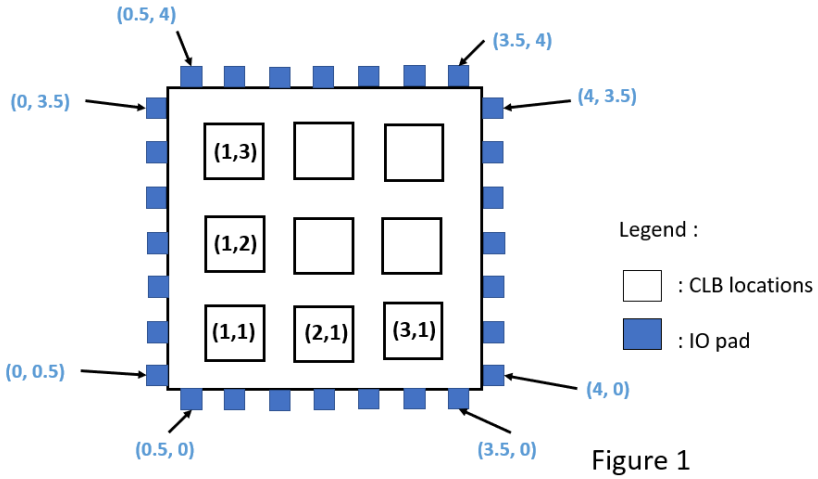

Figure 1

Figure 2 illustrates a circuit after technology mapping. The circuit consists of three LUTs and one FF. Each net connects two or more instances (i.e., LUT or FF), or connects one PI/PO with one or more

instances. For example, net n1 connects I1, L1 and L3.

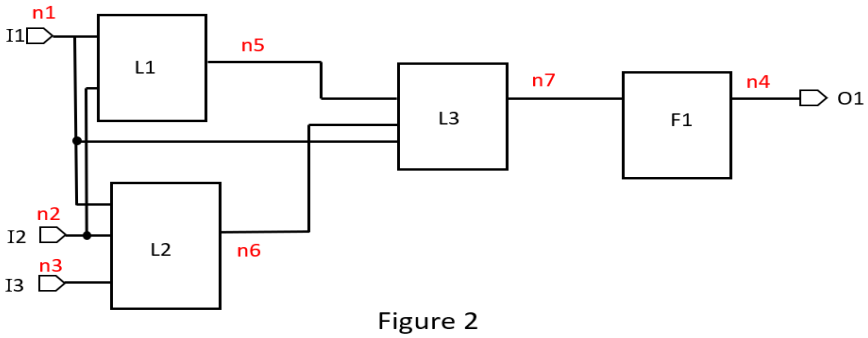

Figure 2

Figure 3 illustrates a feasible placement of all instances in figure 2 where all I/Os were pre-placed. L1 and L2 are placed onto CLB(1,2), and L3 and F1 are placed onto CLB(3,1). We assume that all pins of a CLB are located at the center of the CLB for simplicity. Hence, the HPWL of n1 denoted by the green bounding box in figure 3 is 4.5, the HPWL of n5 denoted by the red bounding box is 3, and the HPWL of n7 is 0. We assume that at most two LUTs and two DFFs can be placed onto the same CLB location.

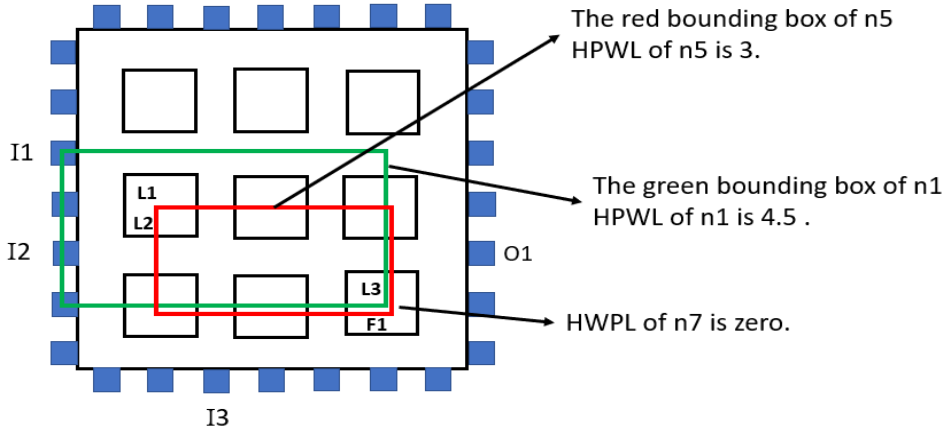

Figure 3

3. Goal and Constraints

- Goal: Place all instances onto CLB locations to minimize the total wirelength subject to the pre-placed PI/POs.

$$min\left(\sum_{net_i \in all\ nets}\left(HPWL_{net_i}\right)\right)$$

- Constraint: at most two LUTs and two DFFs can be placed onto the same CLB location.

4. Input File

There are two input files for each test case. The first input file with ".info" extension specifies the target architecture, some information of the target circuit, and the coordinates of pre-placed PI/POs. In the .info file, line 1 specifies values of **C** and **R** in order. Line 2 specifies values of **Q** and **P** in order. Line 3 specifies the number of primary inputs (***I***), and line 4 to line 3 + ***I*** list all IDs (each

is the form of "I+number") of primary inputs on the first column, the x-coordinates and y-coordinates of them on the second and third columns, respectively. Line 4 + ***I*** specifies the number of primary outputs **(O)**, and line 5 + ***I*** to line 4 + ***I*** + **O** list all IDs (each is the form of "O+number") of primary outputs, the x-coordinates and y-coordinates of them on the second and third columns, respectively. Line 5 + ***I*** + **O** specifies the numbers of LUTs and DFFs (**M** and **N**, respectively). The following **M** lines lists the IDs (each is the form of "L+number") for LUTs. The following **N** lines describe the IDs (each is the form of "F+number") for DFFs.

For example, the content of the .info input file for figure 1 and figure 2 is as follows.

```
CLB_Dim 3 3
Num_I/O_Pad 7 7
Num_PI 3
I1 0 2.5
I2 0 1.5
I3 1.5 0
Num_PO 1
O1 4 1.5
Num_Inst 3 1
L1
L2
L3
F1
```

The second input file with ".nets" extension describes information about of the nets of the target circuit. Line 1 specifies values of **N** which denotes the total number of nets. For the i-th net, line 1 + i shows its ID (each is the form of "ni"), its source, and all of its sinks separated by whitespaces. For example, the content of the second input file for figure 2 is shown below.

```
7
n1 I1 L1 L2 L3
n2 I2 L1 L2
n3 I3 L3
n4 F1 O1
n5 L1 L3
n6 L2 L3
n7 L3 F1
```

5. Output File

Your placement result should be written into an output file with ".placement" extension. Line 1 to line **M** list the name, the x- and y-coordinates of each placed LUT. Finally, line **M** + 1 to line **M + N** list the name, the x- and y-coordinates of each placed FF. (There should be a whitespace separating each item on the same line.)

The output file corresponding to figure 3 is shown below.

L1 1 2
L2 1 2
L3 3 1
F1 3 1

6. Evaluation

- For each benchmark, if your placement result violates the aforementioned constraint, the quality score on that benchmark will be 0.
- If your program takes more than **30** minutes to generate a placement result, it fails on that benchmark.
- Any plagiarism will automatically result in a **0** grade for the project.
- The quality score is based on the total HPWL of your placement result compared to other students when your solution is valid. Here is the equation for score calculation:

$$100 - 25 \times \left(\frac{\text{Your HPWL}}{\text{The smallest HPWL among all teams}} - 1 \right)$$

7. Project Submission

Source codes and an executable file should be uploaded to eLearn. Please include a Makefile for compiling your codes and a Readme file for relevant information in using your codes. You also need to submit a report which describes and analyzes the algorithm you used.

8. Environment Issue

The official evaluation platform will be an Ubuntu workstation, gcc and g++ compilers are available. We do not use Windows platform for evaluation, so please include your Makefile.

9. Required Items

Please compress Project/ (using tar) into one with the name Project_{StudentID}.tar.gz before uploading to eLearn.

➔ Ex: Project_109891011.tar.gz

(1) src/ contains all your source codes, your Makefile and README

- README must describe how to compile and execute your program

(2) outputs/ contains all your output files with ".placement" extension of all benchmarks
(3) benchmarks/ contains all benchmarks
(4) bin/ contains your compiled executable file
(5) Project_{StudentID}_report.pdf which is your report

10. Grading

- 50%: The solution quality
- 50%: The completeness of your program and report