CS 5160-00 FPGA Architecture & CAD Fall 2019

Final Project
Due: Jan. 3, 2020

Power-Aware K-LUT Technology Mapping

1 Background

The combinational part of a circuit can be represented by a directed acyclic graph (DAG). Each node in the graph corresponds to a PI/PO/logic gate, and each edge represents a signal wire in the logic circuit. A primary input node has no incoming edge, and a primary output node has no outgoing edge. For a sequential circuit, we can break it into several DAGs while considering the inputs and outputs of registers as additional POs and PIs.

2 Problem

In this assignment, you have to **write a C/C++ program for K-LUT technology mapping that optimizes the power and the depth**. The depth is computed by the maximum number of levels of K-LUTs from a PI to a PO. Let r_i be the toggling rate of signal *i*. The power consumption of a mapping solution *M* is estimated by summing up the toggling rates of the LUT inputs for all LUTs of mapping *M*, i.e.,

$$\sum_{LUT\ k \in M} \Big(\sum_{i\ is\ an\ input\ of\ LUT\ k} r_i \Big)$$

You are free to adapt any algorithm introduced in the class or reported in the literature or design your own algorithm. You may also brainstorm good ideas with other students, but you must write your own code.

3 Evaluation

i. For each case, if the mapping solution is not valid, you will get **0** score on that test case.

ii. The score is based on the total toggling rate and the depth of your mapping result compared to other students when your solution is valid. Here is the equation for score calculation:

$$90 - 20 \times \left(\frac{\text{Your total toggling rate}}{\text{The smallest toggling rate among all teams}} - 1 \right) - 10 \times \left(\frac{\text{Your depth}}{\text{The smallest depth among all teams}} - 1 \right)$$

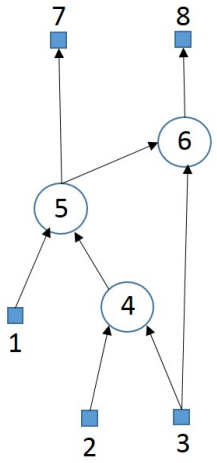
iii. Additionally, if your program takes more than **2** minutes to generate a mapping result, it fails on that test case.

4 Input format

Your program should be invoked using a command in the format below where K is the LUT size:
./program_name input_file_name K output_file_name
e.g. ./map alu4.aag 4 alu4.out (Please follow this format)

The input file describes a directed acyclic graph (DAG), and all the internal nodes (i.e., nodes other than the PI and PO nodes) in the DAG have exactly two inputs. The first line contains the string "agg" followed by three integers, *m, i,* and *o*. *m* denotes the total number of nodes in the DAG. *i* denotes the number of primary input nodes (PIs). *o* denotes the number of primary output nodes (POs). The following *i* lines list the node ID and toggling rate of each PI node. The next *o* lines list the node ID of each PO node and its unique fanin node. The remaining lines provide information about the internal nodes. For each internal node, its node ID is listed first, then the toggling rate of its output, followed by the node IDs of its two fanin nodes.
Here we show the content of a sample input file "simple.aag":

agg 8 3 2 //*m* = 5, *i* = 3, *o* = 2
1 0.1 //PI 1
2 0.12 //PI 2
3 0.06 //PI 3
7 5 //PO 7 with fanin 5
8 6 //PO 5 with fanin 6
4 0.09 2 3 //node 4 with two fanins 2 and 3
5 0.1 1 4 //node 5 with two fanins 1 and 4
6 0.08 3 5 //node 6 with two fanins 3 and 5



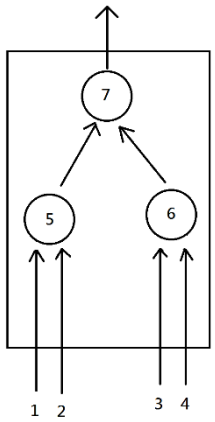
5 Output format

The output file format is described below. Every line in the output file represents a LUT. The first integer is the output node ID of the LUT, and the remaining integers are the IDs of all the fanin nodes of the same LUT.

For example, if the mapping contains a LUT with 4 input nodes as shown in the figure below, then the output line would be like

7 1 2 3 4 //first one is the output node
//and the following are the input nodes

(Note that the order of the input nodes is not important, so “7 3 1 2 4”, “7 4 3 2 1”, etc. are also acceptable.)



6 Project Submission

Source codes should be uploaded to iLMS. 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.

7 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.

8 Required Items

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

➔ Ex: Project_107891011.tar.gz

(1) src/ contains all your source codes, your Makefile and README
   - README must describe how to compile and execute your program

(2) output/ contains all your output files of all testcases

(3) testcase/ contains all testcases

(4) bin/ contains your compiled executable file

(5) Project_{StudentID}_report.pdf which is your report

9 Grading

- 90%: The solution quality
- 10%: The completeness of your report