CS313000 Introduction to Computer-Aided Design of Integrated Circuits

Programming Assignment 1

Deadline: 2022/04/01 23:59

1. Introduction

You are requested to implement a two-level logic optimization program. In this program, you have to reduce the literal count of given Boolean equations. Assume these equations are all written in sum of product (SOP) form. You can solve this problem using Quine-McClusky approach you have learned in class. Also, any heuristic methods in your algorithm are acceptable. Although the literal count is the major consideration of this program, the runtime of your program is also another important factor.

2. Input file format (*.in*)

The first two lines list the total number of variables and the number of product terms, respectively. The next n (n = total number of product terms) lines list the literals of this product term. Each of these n lines contains several characters, which represent the logic value (0, 1, or -) of the variables in the product term. The hyphen (-) means the corresponding variable is don't care. For example, a line *10--11* represents that *ab'ef* is in the on-set of the input Boolean space. The number of the variables shown at the first line will not exceed 26. The letters used will appear in the same order of English letter. For example, if the number of the variables is 6, the letter representing these variables would be *abcdef*.

Sample input:

```
6                // total number of variables = 6
7                // total number of product terms = 7
101110           // product term = ab'cdef'
-10101           // product term = bc'de'f
10-01-
--0111
10-111
100-1-
1-0111
```

3. Output file format (*.out*)

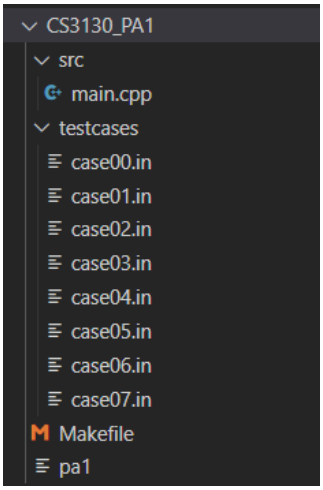
The first line lists the literal count. The second line lists the number of product terms remained in the optimized equation. The next m (m = total number of remained product terms) lines list all the remained product terms after optimization.

Sample output:

```
11          // literal count = 11
3           // total number of product terms = 3
10--1-      // optimized product term = ab'e
-101-1
--0111
```

4. Implementation

A sample directory is provided with a file name, `CS3130_PA1.tar.gz`. Please decompress it using the command `$ tar -zxvf CS3130_PA1.tar.gz`, then a directory named `CS3130_PA1/` with a file structure like the following figure will be produced.



You have to write this program in C/C++ and a Makefile for compiling the program. TAs will compile your source code and run all the test cases (7 released test cases and 3 hidden cases) by the following commands.

```
$ cd CS3130_PA1
$ make
$ ./pa1 <.in file> <.out file>
```

(e.g., `$ ./pa1 case00.in case00.out`)

So, make sure that your program follows the format and accepts the corresponding arguments.

5. Submitted items

Two files should be summited to NTHU eeclass,

- `PA1_{StudentID}.tar.gz`
- `PA1_{StudentID}.pdf`

Please compress `CS3130_PA1/` into one tar file with the name `PA1_{StudentID}.tar.gz` by running the command, `$ tar -zcvf PA1_{StudentID}.tar.gz CS3130_PA1/`. It must contain the **SOURCE CODES in C/C++** under `src/` and a **MAKEFILE**.

A **REPORT** with a file name `PA1_{StudentID}.pdf` is also needed, in which you should contain (1) your name and student ID, (2) how to execute your program, (3) introduction of the data structure and the algorithm you used and (4) other details of your implementation. The page limit of the report is 10.

6. Grading policies

Your program should be scalable for large cases. This assignment will be ranked and scored according to the quality of the experimental results and the runtime. Specifically, the percentage is as follows,

- Accuracy 60% (Ranked by the number of literals)
- Runtime 20%
- Report 20%

For each case, the ACCURACY and the RUNTIME will be graded as follows,

Result	Accuracy	Runtime
Ranks top 35%	60	20
Ranks between 35%-70%	50	15
Ranks lower 70%	30	10
Functionally not equivalent (NEQ)	10	10
Same results as input	0	0
Segmentation fault or **TLE**	0	0

TLE (time limit exceeded): runtime longer than 2 hours.

Other Notes

- If you have any question, please post it on the Discussion board of NTHU eeclass instead of using email since there may be someone else having the same question.
- Your program should be single-threaded. Parallel computation with multiple threads or processes is not allowed.