Andy, Yu-Guang Chen

EE6094 CAD for VLSI Design
Programming Assignment 1: Benchmark Translator
(Due: 23:59:59, 2022/03/19)

Introduction

When evaluating quality of different algorithms in CAD, it is important to use a set of representative circuits to perform fairness comparison. The representative circuits should cover most of the behaviors / structures in various types of ICs to provide credible comparisons. The ISCAS'85 benchmark [1] circuits are ten combinational networks provided to authors at the 1985 International Symposium on Circuits And Systems (ISCAS). They subsequently have been used by many researchers as a basis for comparing results in the area of test generation. To provide fair comparison between different types of circuit descriptions. A Benchmark Translator, which is considered as a tiny EDA tool, is required to automatically perform file transformation for better usage in IC design flow. Therefore, in this programming assignment, **you are requested to implement a C++ program which can take ISCAS'85 netlist descriptions in its Verilog format and translate them to the corresponding circuit benchmark file**. The transformed output file will then be sent to perform combinational equivalence checking (CEC) to verify the correctness.

Background

Benchmarks are used to provide fair comparisons between different algorithms/CAD tools during circuit design and optimization process in various fields such as power, area, performance, and reliability. For digital designs, there are some well-known benchmark sets used by most of the researchers. You can find them at website.

To convert benchmark descriptions to corresponding format, a tiny CAD tool called Benchmark Translator is used. For example, the Benchmark Translator takes the original descriptions of ISCAS'85 benchmark as inputs, and translate them into corresponding VHDL can be found at [2].

Input file format

Your program will take a synthesizable gate-level Verilog (.v) file as an input. We use an example, a small, six-NAND-gate circuit, known as "c17", to explain. The gate level diagram of c17 is shown in Figure 1. Figure 2 shows the Verilog (.v) file of c17. For detail explanations, please refer to [3].

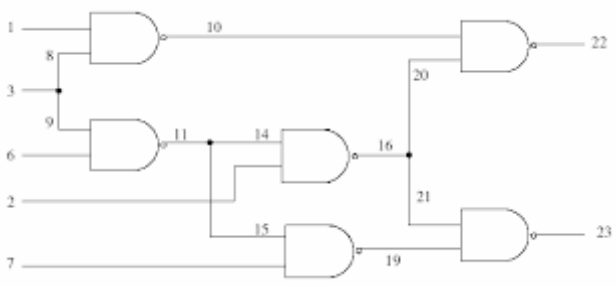

Figure1. c17 circuit structure

```
module c17 (N1,N2,N3,N6,N7,N22,N23);
input N1,N2,N3,N6,N7;
output N22,N23;

wire N10,N11,N16,N19;

nand NAND2_1 (N10, N1, N3);
nand NAND2_2 (N11, N3, N6);
nand NAND2_3 (N16, N2, N11);
nand NAND2_4 (N19, N11, N7);
nand NAND2_5 (N22, N10, N16);
nand NAND2_6 (N23, N16, N19);

endmodule
```

Figure2. c17 circuit in Verilog format

```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )

INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)

OUTPUT(22)
OUTPUT(23)

10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

Figure3. c17 circuit in benchmark file format

Output file format

Your program will generate a benchmark (.bench) file. Figure 3 shows the output of the above example. The first 5 lines start with # are the comment lines which provide basic information of the benchmark circuit, including circuit name, number of input/output ports, number of inverters and logic gates. Then after a blank line, all the input/output ports are specified as one per line. Note that input and output ports need to be separated by a blank line. After that, a blank line appears again, follows by logic gate connection information.

Verification

We will use Combinational Equivalence Checking (CEC) which is one of the useful functions of **ABC (A System for Sequential Synthesis and Verification) [4]** to verify the correctness of your output. Then several benchmark circuit files will be provided for testing the output file you generated. If the generated file can pass the comparison, you will get the point. Three public cases will be provided for your reference, and several hidden cases as well as the three public cases are used to evaluate your program.

Requirement

1. You have to write this program in C or C++. No open source codes are allowed to use. (i.e., you MUST implement the tool by yourself). You can use any data structure to realize your program.

2. All files should be submitted through ee-class. You have to submit **a source code file** named as *StudID*_PA1.cpp (ex: 9862534_PA1.cpp) and **a report** named *StudID_Name*_PA1_report.pdf (ex: 9862534_陳聿廣_PA1_report.pdf). Note that the only acceptable report file format is .pdf, no .doc/.docx or other files are acceptable. **BE SURE to follow the naming rule mentioned above. Otherwise, your program will be not graded.**

3. I will verify your program on a workstation (info will be released later) with the following command:
   $ StudID_PA1.out c17.bench c17.v
   where the first term is the executable file, the second term is the input file name, and the third term is the output file name.

4. We don't restrict the report format and length. In your report, you have to at least include:
   (1) How to compile and execute your program; (You can use screenshot to explain)
   (2) The completion of the assignment; (If you complete all requirements, just specify all)
   (3) The hardness of this assignment and how you overcome it;
   (4) Any suggestions about this programming assignment?

Grading

The grading is as follows:

(1) Correctness of your code: 50%
(2) Readability of your code: 10%
(3) The report: 10%
(4) Demo session: 30%
(5) Bonus (at most): 10%

Please submit your assignment on time. Otherwise, the penalty rule will apply:

- Within 24hrs delay: 20% off
- Within 48hrs delay: 40% off
- More than 48hrs: 0 point

Contact

For all questions about PA1, please send E-mail to TA 金昌明 (abcd29417557@gmail.com)

Reference

[1] David Bryan, “The ISCAS '85 benchmark circuits and netlist format,” available on-line: https://ddd.fit.cvut.cz/www/prj/Benchmarks/iscas85.pdf

[2] Neša P. Tomić and Mile K. Stojčev, “ISCAS-85 Netlist Translator into VHDL Code,” in ICEST 2004.

[3] IEEE P1364-2005 standard, available on-line: https://www.eg.bucknell.edu/~csci320/2016-fall/wp-content/uploads/2015/08/verilog-std-1364-2005.pdf

[4] ABC: A System for Sequential Synthesis and Verification, available on-line: https://people.eecs.berkeley.edu/~alanmi/abc/abc.htm