Andy, Yu-Guang Chen

EE6094 CAD for VLSI Design
Programming Assignment 1: Benchmark Translator
(Due: 23:59:59, 2022/03/17)

- Version 1: 2022.02.17 13:30:00

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). The ISCAS'85 netlist format was never formally documented; rather, it was distributed on magnetic tape along with a FORTRAN translator that would generate netlists in a few other formats. Nowadays the most popular hardware description type is Verilog. Therefore, the ISCAS'85 netlist should be translated into Verilog format for better usage in IC design flow. A Benchmark Translator, which is considered as a tiny EDA tool, is required to automatically perform file transformation. Therefore, in this programming assignment, **you are requested to implement a C++ program which can take ISCAS'85 netlist descriptions in its original format and translate them to the corresponding gate level Verilog format**. The transformed Verilog will then be sent to perform NC-Verilog simulation 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 [2].

To convert benchmark descriptions to corresponding format, a tiny CAD tool called Benchmark Translator is used. The Benchmark Translator takes the original descriptions of the benchmark as inputs, and translate them into corresponding form such as Verilog, VHDL, Spice, BLIF, and so on. An example of implementation ISCAS'85 benchmarks to VHDL can be found at [3].

Input file format

Your program will take ISCAS'85 original description (.isc) 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.

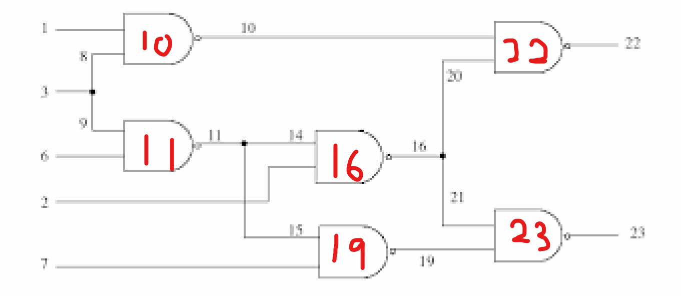
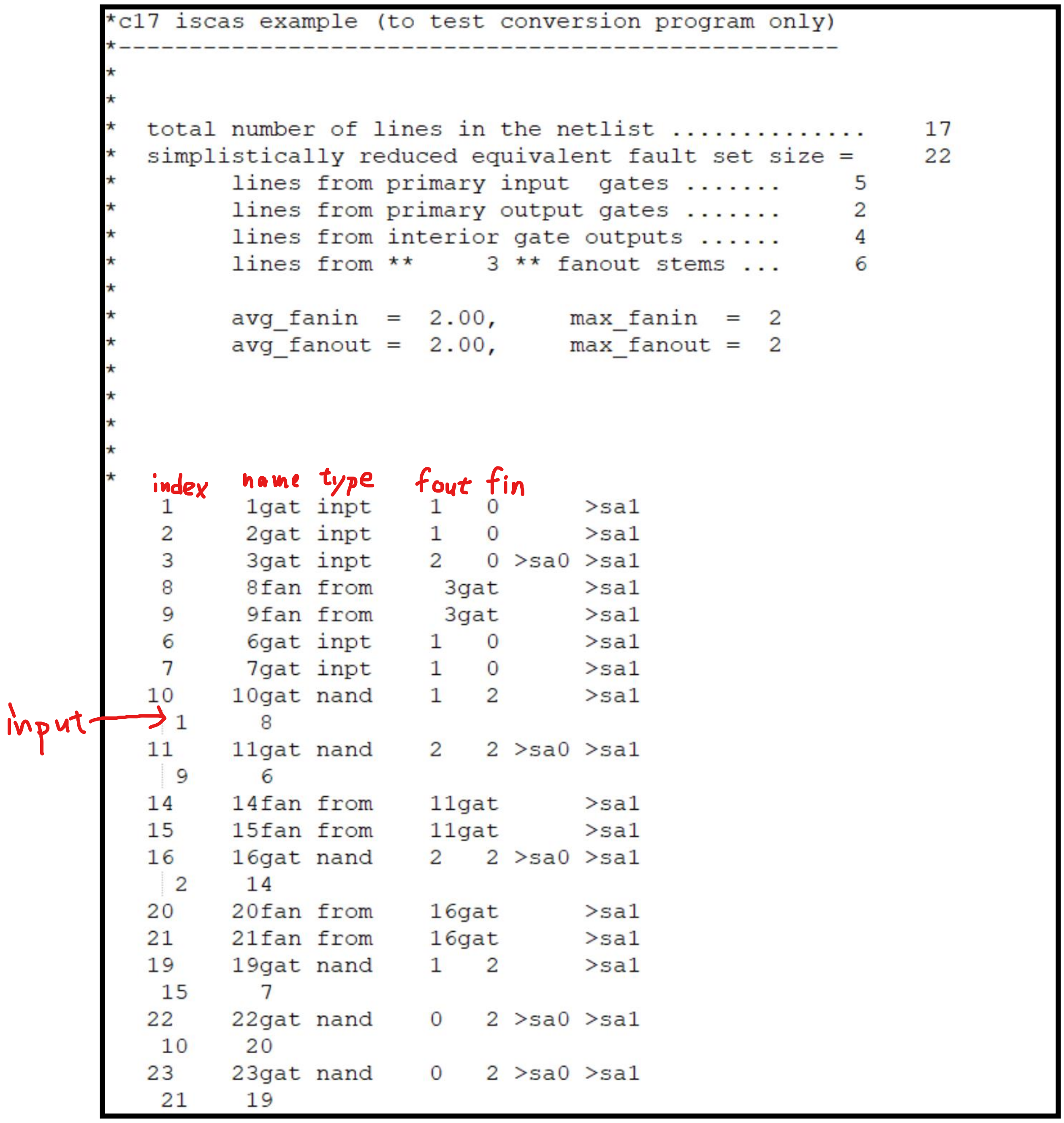


Figure1. c17 circuit structure



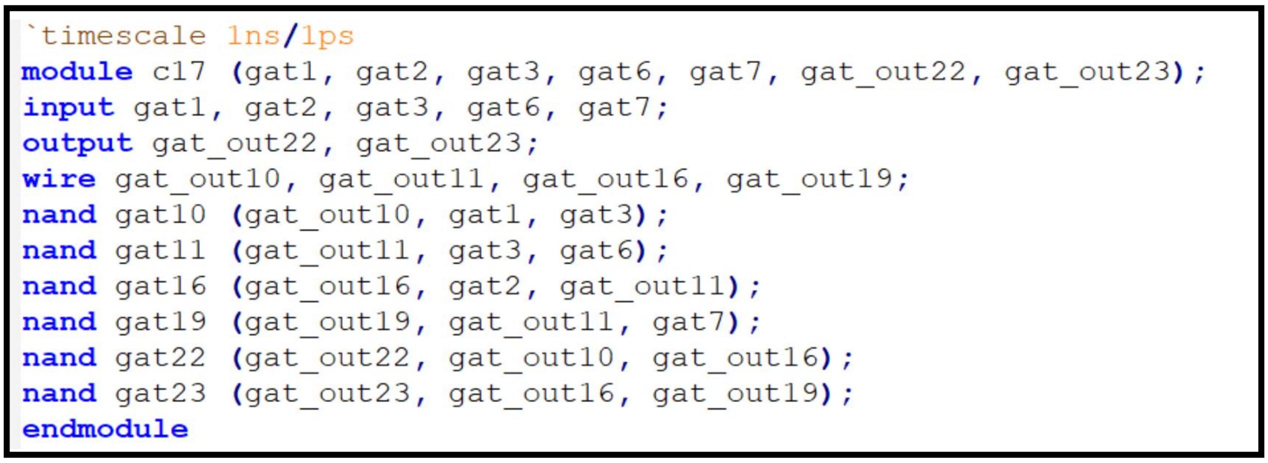
```
*c17 iscas example (to test conversion program only)
*---------------------------------------------------
*
*
*  total number of lines in the netlist ..............     17
*  simplistically reduced equivalent fault set size =      22
*        lines from primary input  gates .......     5
*        lines from primary output gates .......     2
*        lines from interior gate outputs ......     4
*        lines from **     3 ** fanout stems ...     6
*
*        avg_fanin  =  2.00,     max_fanin  =  2
*        avg_fanout =  2.00,     max_fanout =  2
*
*
*
*
*   index     name type     fout fin
     1       1gat inpt    1    0      >sa1
     2       2gat inpt    1    0      >sa1
     3       3gat inpt    2    0 >sa0 >sa1
     8       8fan from     3gat       >sa1
     9       9fan from     3gat       >sa1
     6       6gat inpt    1    0      >sa1
     7       7gat inpt    1    0      >sa1
    10      10gat nand    1    2      >sa1
input ->  1     8
    11      11gat nand    2    2 >sa0 >sa1
      9      6
    14      14fan from    11gat       >sa1
    15      15fan from    11gat       >sa1
    16      16gat nand    2    2 >sa0 >sa1
      2     14
    20      20fan from    16gat       >sa1
    21      21fan from    16gat       >sa1
    19      19gat nand    1    2      >sa1
     15      7
    22      22gat nand    0    2 >sa0 >sa1
     10     20
    23      23gat nand    0    2 >sa0 >sa1
     21     19
```

Figure2. c17 circuit description

Figure 2 shows the original description (.isc) file of c17. For detail explanations, please refer to [1].

Output file format

Your program will generate a synthesizable gate-level Verilog (.v) file. Figure 3 shows the output of the above example.



```
`timescale 1ns/1ps
module c17 (gat1, gat2, gat3, gat6, gat7, gat_out22, gat_out23);
input gat1, gat2, gat3, gat6, gat7;
output gat_out22, gat_out23;
wire gat_out10, gat_out11, gat_out16, gat_out19;
nand gat10 (gat_out10, gat1, gat3);
nand gat11 (gat_out11, gat3, gat6);
nand gat16 (gat_out16, gat2, gat_out11);
nand gat19 (gat_out19, gat_out11, gat7);
nand gat22 (gat_out22, gat_out10, gat_out16);
nand gat23 (gat_out23, gat_out16, gat_out19);
endmodule
```

Figure3. c17 circuit in Verilog format

The first line starts with "`timescale 1ns/1ps" for simulation purpose. The second line uses "module" as key word, follows by the benchmark name. Then, all the inputs and outputs are printed with the increasing order with the node address. Then the inputs/outputs are specified. After that, all the wires (internal connections) used in the benchmark are specified. Then the circuit descriptions with different gates are shown. Finally, a keyword "endmodule" appears to identify the ending of the Verilog file. Note that the generated file needs to follow the Verilog Standard of IEEE P1364-2005 [4].

Verification

We will use **Cadence NC-Verilog simulator** [5] to verify the correctness of your output. A benchmark will be provided for testing the Verilog file you generated. If the generated file can pass the simulation comparison, you will get the point. Three public cases will be provided for your reference, and several hidden case 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.isc 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 黃仕成 (a96hgbc659@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] Collection of Digital Design Benchmarks, available on-line: https://ddd.fit.cvut.cz/www/prj/Benchmarks/index.php?page=download

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

[4] 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

[5] Cadence® NC-Verilog® Simulator Help, available on-line: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.433.1841&rep=rep1&type=pdf