Problem A: X-value Equivalence Checking

Chih-Jen (Jacky) Hsu, Chi-An (Rocky) Wu, Ching-Yi Huang, and Chung-Han Chou
Cadence Design Systems, Inc.

Q&A

Q1. Are input, output and wire in the input file (golden. v and revised. v) only one line or more than one line?

A1. May be multiple lines.

Q2. Is each testcase only one output signal? If there are many output signal, I should check all output and write one answer or one answer for one output?

A2. Multiple outputs signals. One EQ/NEQ as the answer. NEQ if there is any output NEQ, otherwise EQ.

Q3. 1.If the wire is more than one line, which format of input file is it?

wire a, b, c, d, or wire a, b, c, d;
e, f, g; wire e, f, g;

2. n the 3.4 example of problem A, the expected output is
NEQ
in 1
a 1
b 0

However, there is another witness like
NEQ
in 0
a 1
b 0

Is it correct if there are many witness and I just output an arbitrary correct witness?

For instance, in 3.4 example, and my output is
NEQ
in 0
a 1
b 0

A3. Both are possible. Yes, arbitrary witness are okay.

Q4. AND, XOR, OR only have two inputs and one output, is right?

A4. Primitive gates (and, or, nand, nor, xor, xnor) will have multiple inputs (>=2) and single output in Verilog

format.
Primitive gates (not, buf) have only one input and one output in Verilog format.
For example: nand inst_name(o1, in1, in2 ,in3 ,…).

Q5. Can you provide some test examples?
A5. Yes, we will provide testcase soon.

Q6. Does the circuit contain loops?
A6. No, the circuits are all combinational circuit without loop.

Q7. Can we use the package, such as CUDD, that can be found online to solve this problem, or we can only solve this problem from scratch?
A7. Yes, no problem. Please make sure you pack library well and program can be executed in testing/evaluation machines.

Q8. Does the input format must first define input, then output, finally wire?
Or the definition of input, output and wire can occur alternately?
For example,
input a;
output b;
input c;
wire d;
A8. We don't guarantee the order.
It's possible to occur alternately.

Q9. Do the Verilog input, output, and wire have array?
Such as "input [7:0] in1, in2;".
A9. No. They are all represented in single bit:
input name1, name2,…;
However, please notice that Verilog identifier can be named into "\in[1] ". (They are still single bit).

Q10. The netlist in Verilog is as follows:
module ...;
input …;
output …;
wire <name0>, <name1>, …;
…
…
endmodule
My question is:

Does the wire declaration contain all the wires used in the module?

In other words, is it possible that there is a wire named WIRE_1 appearing in one gate but WIRE_1 is not been declared in wire <name0>, <name1>, …?

A10. All used wire names are in the declaration of “input”, ”output” or “wire”.

Q11. Could you please provide some benchmarks for reference?

A11. We provide 2 cases in this week.

There will be more cases in the future.

Q12. It's about the input format.

The document described that the netlist will be expressed in the following format:

<gate type> <name>(<name0>, <name1>, …);

However, I noticed that the released cases contain such descriptions:

buf(\O[19] , \675_Z[19]);

buf(\O[18] , \677_Z[18]);

buf(\O[17] , \679_Z[17]);

buf(\O[16] , \681_Z[16]);

buf(\O[15] , \683_Z[15]);

buf(\O[14] , \685_Z[14]);

buf(\O[13] , \687_Z[13]);

(No <name> in the descriptions)

Is it correct?

A12. Thanks for finding format issues.

We modify the cases. Please re-download them.

Q13. We have some questions on problem A as follows.

We see that order of inputs between golden and revised are different. Will the order affect the result?

If we take an example in problem A, the order input in both files is the same “input in a b”, our output is:

NEQ

in 0

a 1

b 0

If we chance the order input in the golden file as “input a in b”, our output is:

NEQ

in 1

a 0

b 0

Which output is correct? How can we evaluate our results?

A13. No. The order doesn’t affect the result of telling EQ/NEQ.

The inputs and outputs are matched by names.
Besides, when cases are NEQ, it may have multiple patterns.
Contestants need to make sure that their values can evaluate the NEQ result.

We already tell the simulation table and which kinds of output pins' values are NEQ in document, so contestants can implement the correct simulator to double check their results.
Besides, alpha/beta test stage will warn contestants if their results are wrong.

Q14. I have a question about the order of the ports, especially for _HMUX and _DC.
Is the order of the ports fixed in the following order?
_HMUX \U$1 (.O(\282), .I0(1'b1), .I1(\277), .S(\281));

_DC, // _DC \n6_5[9] (.O(\108), .C(\96), .D(\107));

A14. We modify the cases' format without pin names. Please re-download them.
Their connection order is the same as the module pin order in the document. e.g.
_DC <name> (n1, n2, n3);
n1 is connected to pin O.
n2 is connected to pin C.
n3 is connected to pin D.

Q15. I found that the number of inputs in some cases is different, such as case1.

In gf.v, the input port "\I[7]" has been used in:
nor \U$8 (\89 , \I[0] , \88 , \I[2] , \I[3] , \I[4] , \I[5] , \I[6] , \I[7]);
nor \U$6 (\92 , \91 , \I[1] , \I[2] , \I[3] , \I[4] , \I[5] , \I[6] , \I[7]);
nor \U$4 (\94 , \I[0] , \I[1] , \I[2] , \I[3] , \I[4] , \I[5] , \I[6] , \I[7]);
buf \U$2/A[7] (\102 , \I[7]);
But in rf.v, this input port has never been used.

Would you please explain to me that this way is right?

A15. "\I[7]" is the input port of "rf.v".
The port number of gf.v and rf.v are equal by our parser.
Can you show which port doesn't exist?
If you take about using nets as drivers, any net is allowed to be not used.
The situation doesn't violate the format.

Q16. I noticed all inputs and outputs are declared in the wires in ProblemA_0201.pdf.

module top(in, a, b, out);
input in, a, b;

output out;
wire n2, n1, en1, en2b, en1b, out, b, a, in;

Will this happen in the test cases?

A16. Inputs are allowed to be declared as wires.
However, we don't guarantee that inputs must be in wire's declaration.

Q17. We did some analysis on the testcases that you have released.
In all six cases, we found that there doesn't exist any _DC gate in those rf.v(revised circuit) files.
Can we assume that there wouldn't be any _DC gate in rf.v(revised circuit) file?

A17. No. It is possible to have _DC gates in both sides for industrial applications.

Q18. According to the Q&A 13, contestants can implement the correct simulator to simulate if the answer is correct. However, we can't check whether the answer is EQ or NEQ. Can you provide whether the answer is EQ or NEQ of those released cases?

A18. In alpha test, contesters will know if their results are wrong. Alpha test will include public cases.

Q19. Will the hidden test-cases have exactly the same text format as the test-cases we have? I would like to know because this will affect our parser.

A19. The format of cases is described in the document. All cases match this format.

Q20. We wondering if the organizers can provide more test benches. A suite of test cases would not only helps us validate our parser, sweeping through potential issues, but also can drive the development for performance.

A20. We will provide more cases after alpha result.

Q21. We are a bit confused regarding the results of the Alpha Submission. We thought there would be hidden test cases that we would be notified whether we passed them or not. Is that the case? Or do we just get the results of the provided test cases only?

A21. In alpha stage, there is no hidden case.
Besides, it is possible that some new hidden cases only appear in final stage.

Q22. Can you provide us with the result (EQ or NEQ) of each case?

A22. We already publish the answer of alpha cases in website:
http://iccad-contest.org/2020/Problem_A/2020CAD_Alpha%20Test%20results_Problem%20A.pdf
The answer of beta cases (cases10~18) will be published after beta test.

Q23. We are confused about the rank of the Alpha Submission. Are the results of different cases of the same rank all belong to the same team? Or do the results of the same rank in different cases may belong to different teams?

A23. Not the same team. The rank is not the team's rank.
It is the best solving time for each case, which can come from different teams.
The candidates of solving time are collected by teams which don't give wrong answer in all cases.

Q24. This is regarding the beta test submission. What is the limit on the total memory usage for the program?

A24. It should not be more than half of the total memory of the test machine.

Q25. There are two testcases files released : case2 and case9.
in case2, 2 testcases uses _DC gates and _HMUX gates with the format as follow :
_DC gate_name(.O(output_name) , .C(input1_name) , .C(input2_name));
_HMUX gate_name(.O(output_name) , .I0(input0_name) , .I1(input1_name) , .S(select_name));
however, in cases9, 9 testcases uses
_DC gate_name(output_name , input1_name , input2_name);
_HMUX gate_name(output_name , input0_name , input1_name , select_name);
can I suppose that in our final contest, the format will always be like cases9 ? it would be easier to parse the verilog code.
or should I suppose that the testcases will use either one of two formats (I hope not D:) ?
I read the Q&A, there is Q14 asking question like my question. but it seems that the testcase "case2" has not been modified.

A25. The format of all cases is the format of cases 9 testcases. There won't be format using ".O".
The modified format of 2 cases in the webpage link "case2" are already in webpage link "cases".
"case2" is out-of-date.

Q26. Could you please provide the results by the team rank?
For example, the number of solved cases and the corresponding CPU time by a team.
I know the results have been announced.
However, the results are ranked according each case instead of team.

A26. Because the final evaluation will have other hidden cases, including very-hard NEQ cases,
The team rank will be affected by hidden cases in the final evaluation.
Hence, we only show the best runtime of each case for reference.

Q27. Could you please provide some hard-NEQ benchmarks for reference? According to the Beta test, all the released NEQ benchmarks can be solved within a few seconds. We think the hard-NEQ benchmarks can help us refine and improve the algorithm/program we're currently developing. Obviously, these hard-NEQ benchmarks mean a lot to the X-value equivalence checking problem.

A27. The very-hard-NEQ cases are hidden cases for final evaluation.
We cannot release them. Contestants need to guarantee the correct theory of their algorithm.
What we MAY do is to create some non-real cases, which contestants also can create by themselves.