Problem A: Learning Arithmetic Operations from Gate-Level Circuit

Chung-Han Chou, Chih-Jen (Jacky) Hsu, Chi-An (Rocky) Wu, and Kuan-Hua Tu
(Cadence Design Systems, Inc.)

Q&A

Q1. I have some questions about Problem A and need your help to check it.
The problem A said: Parallel computation with multiple threads or processes is not allowed.
Does it mean that we cannot use GPU? or cannot use machine learning?
Thank you very much.
A1. GUP is not allowed in this contest. However, you still can use machine learning algorithms by performing inference on a CPU.

Q2. I have a question about Problem A:
Do the designs only contain combinational logic? Thank you.
A2. Yes, there will only be combinational logic in the netlists.

Q3. We would like to know if there's any restriction on the problem size. Thanks.
(for example, the maximum number of gates or input/output nets)
A3. We do not make assumptions about the size of input netlist.

Q4. Will you provide a Verilog parser to Contestants to parse the netlists to be learned and make them concentrate on the learning algorithms?
A4. No, we won't provide Verilog parser to contestants

Q5. We found a way to rewrite any given circuit with constant cost (5). We would like to know if this is a valid solution and if the cost is really 5. Thank you for your consideration.

Given a combinational circuit,
we can represent the circuit using only NAND gates.
To rewrite the circuit with verilog operators,
we can simply concatenate the fanin nets of every NAND gate into 2 bit vectors,
and use the bit-wise NAND operation to complete the whole circuit.

The verilog code will look like the following,

```
assign { z1, z2, ..., zn } = ~( { x1, x2, ..., xn } & { y1, y2, ..., yn } );
```

and the total cost would be 5.

A5. Thanks for pointing out the shortcomings. We modify the cost of logic operators and concatenations to reflect bit-width or symbol number. Please refer to the updated problem description released on the website.

Q6. We have a question about the format of input file of problem A.
Could we assume the format of the input verilog files would be as same as the sample input files as follows:

module top(<inputs>, <outputs>);
    primary input declaration;
    primary output declaration;
    wires for primary inputs (same order with input declaration);
    wires for primary outputs (same order with output declaration);
    intermediate wires declaration;
    gate-level circuits;
endmodule;

and will primary inputs named after in1, in2, in3.....
primary outputs named after out1, out2, out3.....?

It would be a big help on parsing the input files if we can make the assumption.
Thank you very much!

A6. Contestants are not allowed to make any assumption on variable names of input/output ports, internal wires, and gate names.