IWLS 2023 Programming Contest

Alan Mishchenko, UC Berkeley

It has been recently observed that the cost of silicon wafers has started to grow steeply in the 3rd quarter of 2021. Please note the hockey stick shape of the second graph here https://morethanmoore.substack.com/p/tsmc-financial-year-2022 As a result, the design area contributes more to the design cost, and there is increasing interest in high-effort area optimization methods, which is the topic of this year's IWLS Programming Contest.

Please note that the contest this year is an extension of the contest held at IWLS 2022 (https://github.com/alanminko/iwls2022-ls-contest). Please read the description: https://www.iwls.org/iwls2022/contest/IWLS_2022_Programming_Contest.pdf

Similar to the last year, the participants are given 100 truth tables representing Boolean functions. The goal is to synthesize two circuits for each truth table: (1) the traditional and-inverter-graph (AIG), as in the last year's contest, and (2) the xor-and-inverter graph (XAIG) composed of any two-input gates, including exclusive-or gates. In the former case, the cost of a circuit, as before, is the number of internal two-input and-nodes. In the latter case, the cost is the number of any two-input gates, including xors.

The resulting score of each participant will be determined by the same formula as in the last year's competition. There will be three winners who submit functionally equivalent AIGs/XAIGs with: (i) the best score of the traditional AIGs, (ii) the best score of the XAIGs, and (iii) the best overall score. Functionally incorrect circuits as well as not submitted circuits, as before, will receive the zero score.

The question is, how to represent the resulting XAIGs. The known AIGER format (http://fmv.jku.at/aiger) for the traditional AIGs can only represent and-gates. The work-around below allows for representing xor-gates in this format.

The idea is to represent each two-input xor-gate as an equivalent combination of three and-gates with complemented attributes as needed. The resulting traditional AIG can be written in the AIGER format and verified against the original truth table, as described in the last year's competition announcement. To determine the cost of the XAIG, according to this year's competition rules, it is possible to use ABC to read the AIG, with xor-gate replaced by and-gates, using command "&read file.aig", convert it into an XAIG using command "&st –m –L 1", and list the node statistics using command "&ps –m".

For example, the benchmark "i10.aig" contains 2675 two-input and-nodes, which is the cost of this circuit as the traditional AIG, according to the competition rules:

```
abc 01> &r i10.aig; &ps; &st -m -L 1; &ps -m
i10      : i/o =    257/    224  and =    2675  lev =   50 (  15.54)  mem = 0.04 MB
```

When this circuit is converted into an XAIG, as discussed above, it has 151 two-input xor-gates and 2235 and-gates. The total two-input node count in this case is 2386, which is the cost of "i10.aig" as an XAIG, according to the competition rules:

```
abc 01> &r i10.aig; &st -m -L 1; &ps -m
Generated AND/XOR/MUX graph.
i10      : i/o =    257/    224  nod =    2386  lev =   50 (  15.54)  mem = 0.04 MB
XOR/MUX stats:  xor = 151  16.85 %   mux =  0   0.00 %   and = 2235  83.15 %  obj = 2386
```