High-Effort Logic Optimization for IWLS 2025 Programming Contest

Yukio Miyasaka (UC Berkeley)
Jiun-Hao Chen (National Taiwan University)

Outline

- Problem Definition
- I/O Matching
- Reverse Engineering
- Circuit Construction
- Rewiring
- Resynthesis
- Summary

Problem Definition

- Synthesize as small AIGs as possible from truth tables
  - The number of inputs is at most 16
- 200 benchmarks in total
  - 100 benchmarks are the same as 2022-2024
  - 100 new benchmarks

I/O Matching

- For the continued 100 benchmarks, we reuse the best results over the previous 3 years
- 2023 and 2024 benchmarks are obfuscated within **NPN classes**, with extra constant/buffer outputs
- We figured out the permutations of inputs and inverter presences using **QBF**
- We used **ABC's QBF solver** that has been known as the state-of-the-art
- However, for some cases it did finish even after a day
- We modified ABC to use **CaDiCaL** as backend solver of QBF

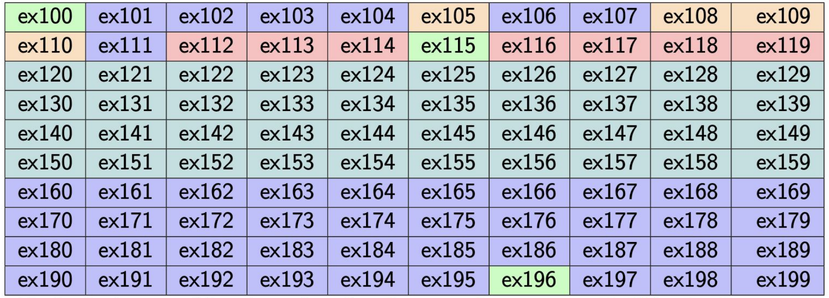
Reverse Engineering

- The 2025 benchmarks appear to be unobfuscated
- Manual Inspection
    - Story from ex120 (an unsigned 4x4 multiplier)
    - ex120~ex149 (common? arithmetic circuit)
    - Interesting case from ex150~ex159
        - 0 70 35 0 21 91 56 0 42 7 77 0 63 28 98 0 84 49 14 0 0 …
        - z = (a >= 7 || b >= 5 || c >= 3) ? 0 : (15*a+21*b+70*c) % 105;
- Acknowledgements
    - Chien-Kai, Ma
    - Shao-Jui, Wu
    - Che, Cheng

Circuit Construction

The initial circuit is generated from the following methods

- ABC commands:
  - strash
  - clp;reorder;muxes;strash (BDD-based)
  - &ttopt (BDD-based)
- Decision graph
  - Entropy+Fourier analysis+functional decomposition
- Reverse engineering
  - Generate from word-level expression with yosys



ex100	ex101	ex102	ex103	ex104	ex105	ex106	ex107	ex108	ex109
ex110	ex111	ex112	ex113	ex114	ex115	ex116	ex117	ex118	ex119
ex120	ex121	ex122	ex123	ex124	ex125	ex126	ex127	ex128	ex129
ex130	ex131	ex132	ex133	ex134	ex135	ex136	ex137	ex138	ex139
ex140	ex141	ex142	ex143	ex144	ex145	ex146	ex147	ex148	ex149
ex150	ex151	ex152	ex153	ex154	ex155	ex156	ex157	ex158	ex159
ex160	ex161	ex162	ex163	ex164	ex165	ex166	ex167	ex168	ex169
ex170	ex171	ex172	ex173	ex174	ex175	ex176	ex177	ex178	ex179
ex180	ex181	ex182	ex183	ex184	ex185	ex186	ex187	ex188	ex189
ex190	ex191	ex192	ex193	ex194	ex195	ex196	ex197	ex198	ex199

Methods denote with color: `strash`, `clp;reorder;muxes;st`, `&ttopt`, `dg`, `yosys`

Rewiring

Don’t care-based redundancy addition and removal method

- ABC: rewire
  - -E [10,150] (total wire can be added)
  - -D [5,20] (total common divisor can be extracted)
  - -L [2,5,unlimited] (distance between candidate wire and target node)
- ABC: stochmap
  - dch;amap (map an AIG into a library only with variant of AND2 and INV)
  - With or without -o (enable overlapping partitioning)
  - Optimize each subcircuits with rewire

Resynthesis

Implemented a heuristic flow in Yukio’s framework applying the following methods:

- ABC: balance, rewrite, refactor, resub, orchestrate
    - with or without -z, -l
    - resub/orchestrate: -K [5, 15], -N [0, 3]
    - resub/orchestrate ODC level: -F [1, 3], while it often crashes as there is a hard limit (2^16) for ODC circuit, which not easy to fix
- ABC: deepsyn sequence
    - Also tried LMS (&if -y) and other balancing methods (&if -g, &if -x)
- ABC: map/amap-based deepsyn
    - There is a bug that makes non-equivalent results for some libraries with MUX (fixed: 6/17/2025)
- Mockturtle: rewrite (cut, window), refactoring (dsd), resubstitution (aig, aig2, sim)
    - Similar options to ABC
    - Fixed a bug in combinational loop handling in window rewriting
    - Fixed a bug in window construction referencing resized vector element
    - use_dont_care was not stable
- Partial resynthesis to perturb
    - Used Yukio’s framework to extract a small partition (about 30 nodes)
    - &ttopt with SDC
    - &transduction -T 3
    - collapse; {sop; fx, dsd, bidec}, while CUDD was problematic when trying multithreading, ended up executing ABC through system call for these
    - bidec also has some hard coded number of nodes to alloc
- Yukio’s implementation of redundancy addition and removal

Summary

- List of methods
  - QBF-based matching
  - Manual reverse engineering to construct compact initial arithmetic circuit
  - Decision graph and BDD-based initial circuit construction
  - Don’t care-based rewiring
  - Resynthesis using various ABC/mockturtle algorithms
- We submitted what we had at deadline, but it is not saturated yet; further reduction is possible
  - For some cases, we spent weeks in rewiring to reduce just 1 node
  - Optimization progress was made by alternating different methods
    - Applying just one method, including rewiring, tends to get stuck
- We didn’t have time to try out circuit model learning