Weekly Report

March 31, 2019

1 Targets

1.1 Urgent

- Change PI don't care set: simplify CNF by logic synthesis or special input pattern.
- Change PI care set: use different random seed.
- Change window don't care set.

1.2 Important

- Randomly selecting a node, compute its care set, change its care set, synthesize the local circuit, evaluate its area and error rate, accept it with a certain probability.
- How DC affects BDD
- Induce DC with PLA files
- Use model counting to compute error rate, i.e., how many assignments satisfying a SAT problem.
- Use approximate confidence interval / hypothesis testing of Bernoulli experiments to evaluate the accuracy of error rate.
- Trade off the accuracy of batch error estimation for speed (even directly use Su's equation to update Boolean difference), perhaps use hypothesis testing to evaluate the accuracy.
- Combine the simulation of circuits with the simulation of Monte Carlo Tree Search. In other words, in one loop of Monte Carlo Tree Search, merge logic simulation and playout (only simulate circuit once and playout once).
- Represent circuit with AIG because of more potential LAC candidates. For each round, select one or more input wires and replace them with constant 0 or 1. Consider how to combine Wu's method (choose a subset of input wires and substitute).
- Accelerate Approximate Logic Synthesis Ordered by Monte Carlo Tree Search: reuse the result of batch error estimation in playout.
- Use UCB1's bound to guide the simulation time. Find relationship of different bounds.

1.3 Worth Trying

- Enhance default policy with greedy approach or field domain knowledge.
- In expansion process of MCTS, expand more than one layers.
- Tune parameters in MCTS.
- Perform greedy flow on leaves of the final Monte Carlo Search Tree.
- Combine beam search and MCTS.

1.4 Potential Topics

- Relationship between power simulation and logic simulation.
- Combine Binarized Neural Network with approximate computing.
- Relationship between Boolean network and Bayesian Network.
- Approximate TMR.

2 Progress

2.1 Issues of Last Meeting

2.1.1 Equation Revision

Notes:

- Lowercase characters of normal style (x) denote scalars.
- Uppercase characters in blackboard bold ($\mathbb{X}$) denote sets.
- Lowercase characters in boldface ($\mathbf{x}$) denote vectors.
- Uppercase characters in boldface ($\mathbf{X}$) denote matrices.

Let the vector of PI variables be $\mathbf{x} = (x_1, \ldots, x_I)$. The length of $\mathbf{x}$ is I.
Let the vector of PO variables be $\mathbf{y} = (y_1, \ldots, y_O) = (f_1(\mathbf{x}), \ldots, f_O(\mathbf{x}))$. The length of $\mathbf{y}$ is O.
If we simulate the circuit for R rounds with PI assignments $\mathbf{A} = [\mathbf{a_1}, \ldots, \mathbf{a_R}]$, where $\mathbf{a_i} = (a_1, \ldots, a_I)(1 \leq i \leq R)$.
Let the approximate output vector be:

$$\hat{\mathbf{y}} = (\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_O(\mathbf{x}))$$

For the i-th output,

$$\hat{f}_i(\mathbf{x}) = g_i(\mathbf{x}, \mathbf{a_1}) \vee \ldots \vee g_i(\mathbf{x}, \mathbf{a_R})(1 \leq i \leq O)$$

where

$$g_i(\mathbf{x}, \mathbf{a_j}) = (g_i(x_1, a_{j1}), \ldots, g_i(x_1, a_{jI}))(1 \leq j \leq R)$$

$$g_i(x_k, a_{jk}) = \begin{cases} x_k, & a_{jk} = 1 \\ \neg x_k, & a_{jk} = 0 \end{cases}$$

2.1.2 Explanation of the Bug on Directly Inserting DCs of PIs

Appearance:
After extract the window for a node, I built a miter for it. Then I used two similar methods to insert DCs of PIs:

- all PIs are used;
- only the PIs supporting the window are used.

The derived circuits are quite different. The first way naturally generated what I wanted, but the second way generate circuits different from those in the first.
Reason:
I give a simple counterexample to explain the error of the second method.
Assume there are 5 PIs, a, b, c, d and e. Let $abcde = 01010$ be the only don't care.
For a node n, its supporting PIs are a, c and e. Then $ace = 000$ is used to simplify node n.
For node n's fanout node m, all PIs support node m. In other words, $m = f(n, b, d)$. Then m is in trouble, since n is generated by setting $abcde$ = 0-0-0 as don't care.

2.2 Test

2.2.1 Change PI don't care set with special input pattern

For c880, I tried one input pattern with only one determined bit, and set it as don't care, For example, a circuit has 5 PIs, I set $--0--$ as its external don't care. It generated circuit with error rate which is near 50%. And it is feasible.
However, when I tried several thousands of input patterns with 2 determined bits, it does not further simplify c880 compared to the original circuit. It's strange.

2.2.2 Change window don't care set

Under development. Program is still somewhere wrong.