Weekly Report

March 24, 2019

1 Targets

1.1 Urgent

- 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.

1.2 Important

- 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.
- How DC affects BDD
- Induce DC with PLA files

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 Unsuccessful Verification of Circuits Generated by Directly Inserting Approximate DCs to PIs

Last week, I made a mistake in my code. When I inserted the approximate DC patterns to the PIs of a window, I did not correctly match the order of PIs and the order of DC patterns. For example, by default, there are 3 PIs a, b and c. PIs[0] = a, PIs[1] = b, PIs[2] = c. I inserted a pattern p = 010, p[0] = 0, p[1] = 1, p[2] = 0. For a window, its supporting PIs are a and c. I failed to record the correct reflection of windows' supporting PIs with the original circuit's PIs. Then I probably treated a as PI[0], c as PI[1], so mistakes occur.
By fixing the bug mentioned above, I found that the effect of a small set of approximate DCs was not prominent. But it is impractical to create a SAT solver for a large DC set. New techniques are required.

2.2 New Ideas

Notes: Upperbase characters in blackboard bold ($\mathbb{X}$) denote sets. Lowerbase characters in boldface ($\mathbf{x}$) denote vectors. Upperbase characters in boldface ($\mathbf{X}$) denote matrixs.

2.2.1 Simulation Based Approximate Computing

Let the primary input vector of the circuit be $\mathbf{x}$. The number of primary inputs is I, namely, $|\mathbf{x}| = I$. The number of possible assignments for $\mathbf{x}$ is 2^I.
Let the primary output vector of the circuit be $\mathbf{y}$. The number of primary outputs is O, namely, $|\mathbf{y}| = O$. We have $\mathbf{y} = (f_1(\mathbf{x}), \ldots, f_O(\mathbf{x}))$.
If we simulate the circuit for R rounds with input assignments $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_R]$, let the approximate function be:

$$\hat{\mathbf{y}} = (\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_O(\mathbf{x})), \tag{1}$$

where

$$\hat{f}_i(\mathbf{x}) = f_i(\mathbf{x})|_{\mathbf{x}=\mathbf{a}_1} \vee \ldots \vee f_i(\mathbf{x})|_{\mathbf{x}=\mathbf{a}_R}$$

When $R \to \infty$, $\hat{\mathbf{y}} \to \mathbf{y}$.
The error rate E can be controlled by R,

$$E = \frac{e}{2^I} \leq 1 - \frac{R}{2^I},$$

where e is the errornous input number. Actually, E is monotonic to R.
In practice, we directly use Eq. (1) and build the approximate circuit increasementally. The number of nodes in such a circuit is less than IOR.

2.2.2 Combine Eq. (1) with Don't Cares

For a node V, we extract its window $\mathbb{W}$ by limiting the level of transitive fanin/fanouts.
The input vector of $\mathbb{W}$ is $\mathbf{s}$. The number of window inputs is K, namely, $|\mathbf{s}| = K$.
Assume all binary combinations of the assignments of $\mathbf{s}$ are $\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_{2^K}]$.
Assume all possible assignments of $\mathbf{s}$ are $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_T]$. $T \leq 2^K$ (there are SDCs).
The value of V is v.
Similarly, we have $v = h(\mathbf{s}) = h(\mathbf{s})|_{\mathbf{s}=\mathbf{b}_1} \vee \ldots \vee h(\mathbf{s})|_{\mathbf{s}=\mathbf{b}_T}$.
If we simulate the circuit for R rounds with primary input assignments $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_R]$, and let the corresponding window input assignments be $\hat{\mathbf{B}} = [\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_L](L \leq T)$, The approximate function is:

$$\hat{v} = \hat{h}(\mathbf{s}) = h(\mathbf{s})|_{\mathbf{s}=\hat{\mathbf{b}_1}} \vee \ldots \vee h(\mathbf{s})|_{\mathbf{s}=\hat{\mathbf{b}}_L} \tag{2}$$

In practice, Eq. (2) can be implemented by setting $\mathbf{G} \backslash \hat{\mathbf{B}}$ as don't cares.