Weekly Report

March 8, 2019

1 Targets

1.1 Urgent

- Make out how *mfs* works in detail and implement its approximate computing version.

1.2 Important

- 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.

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 Detailed MFS Flow

From the paper and ABC's mfs code, the detailed process to resubstitute a node f is explained below.

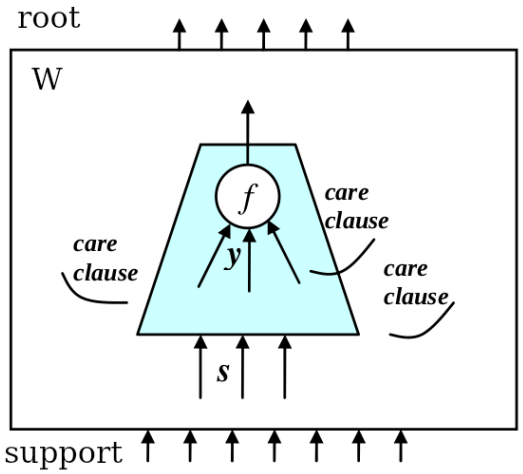
2.1.1 Construct Window

1. Perfrom DFS from node f to POs, record those *root* nodes. A *root* satisfies one of the conditions:

- the node has more than fanouts than the limit;
- the node has CO fanouts;
- the node has fanouts above the cutoff level.

2. Find the *support* nodes of node f, which means the PIs in the fanin cone of node f.
3. Record all nodes on paths from *root* nodes to PIs, and those nodes forms a window W.

The window is shown below:



2.1.2 Find Divisor Candidates

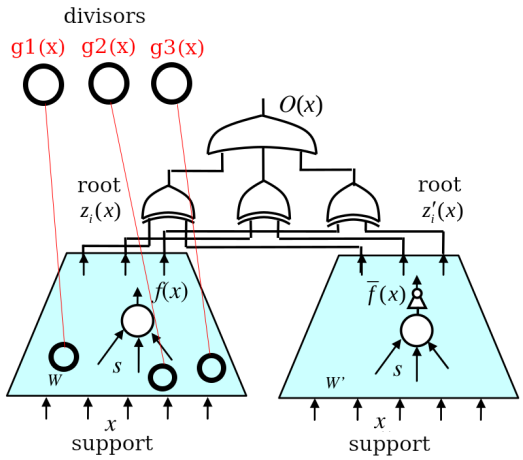
Collect *divisors* from node f to PIs, avoid these nodes:

- the node f itself;
- the nodes in the MFFC of f.

Explore the fanouts of already collected *divisors*, and add the fanouts into *divisors* excepted for:

- a divisor has too many fanouts;
- nodes in the TFO or in the MFFC of node f;
- PO nodes;
- nodes with large level;
- nodes whose fanins are not divisors (I am not clear here);

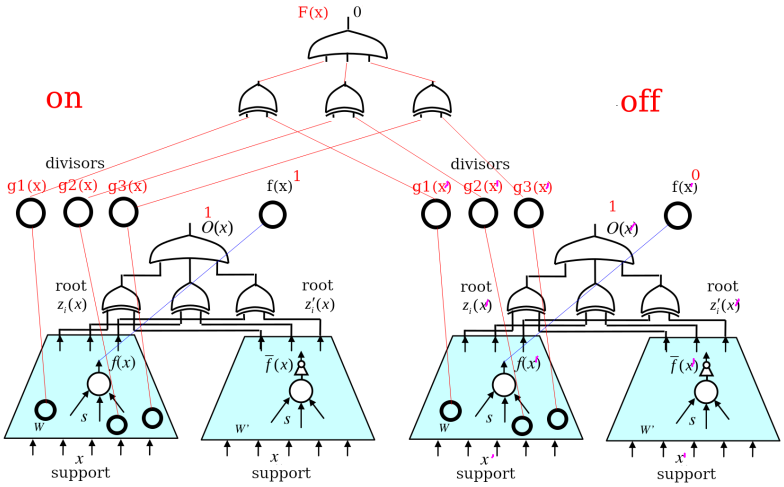
2.1.3 Construct AIG for Window W



If $O(x)$ is asserted to 1, then x is in the care set of f.

2.1.4 Translate AIG into CNF

2.1.5 Create the SAT Problem



Here, x and x' are different input patterns. $O(x)$ and $O(x')$ is asserted to 1, then x and x' are in f's care set.
We want to substitute $f = f(s)$ by $f = h(g_1, g_2, \ldots, g_n)$. The dependency function h exists if and only if there is no minterm pair (x, x'), where $f(x) \neq f(x')$ and $g_j(x) = g_j(x')$ for all j. So we assert $f(x) = 1$, $f(x') = 0$, all $g_j(x) \oplus g_j(x') = 0$.
The CNF expression of the SAT problem is:

$$G = C^{on} f^{on} O^{on} C^{off} \bar{f}^{off} O^{off} (g_1^{on} \equiv g_1^{off}) \ldots (g_n^{on} \equiv g_n^{off})$$

2.1.6 Interpolation

To compute a function $h(g)$, G divided into subsets A and B.

$$A = C^{on} f^{on} O^{on}, B = C^{off} \bar{f}^{off} O^{off} (g_1^{on} \equiv g_1^{off}) \ldots (g_n^{on} \equiv g_n^{off})$$

For unsatisfiable G, the resultant interpolant $A^{\#}$ derived from a refutation proof yields a desired dependency function h (I am not clear here).

Definition of interpolation:

2.4 Interpolation

Given Boolean functions, $(A(x,y), B(y,z))$, such that $A(x,y) \wedge B(y,z) = 0$, and (x,y,z) is a partition of the variables, an *interpolant* is a Boolean function, $I(y)$, such that $A(x,y) \subseteq I(y) \subseteq \overline{B}(y,z)$. If $A(x,y)$ and $B(y,z)$ are sets of clauses, then their conjunction is an unsatisfiable SAT instance, and an interpolant can be computed from a proof of unsatisfiability using the algorithm in [13] (Definition 2).

$A(x,y)$ can be interpreted as the onset of a function, $B(y,z)$ as the offset, and $\overline{A(x,y)} \wedge \overline{B(y,z)}$ as the don't-care set. Thus $I(y)$ can be seen as an optimized version of $A(x,y)$ where the don't cares are used somehow, e.g. I is only a function of the variables y.

2.2 Approximate Version of MFS

I have no mature idea up to now.