# Weekly Report

April 23, 2019

## 1 Targets

### 1.1 Urgent

- 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. Change simplification method (mfs$\rightarrow$bdd).
- Introduce 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.
- Influence of network representation on synthesis. Why does mfs use local AIG function to represent the circuit, is it more fittable to LUT mapping?

### 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

**Pseudo code**

```
Algorithm 1: DCALS
  Input: $C_{ori}, originalcircuit$
  $E$, error threshold
  $nWinTfoLevs$, maximum level of TFO cone of windows
  $nWinTfiLevs$, maximum level of TFI cone of windows
  $nFanoutMax$, maximum branch factor of TFO cone of windows
  Output: approximate circuit $C_{ax}$

1 $C_{ax} \leftarrow C_{ori}$

2 // Directly given so far
3 Determine a proper simulation number $N$ according to $E$

4 forall node $V$ in $C_{ori}$ do
5     // not the topological order, just select nodes one by one
6     ApproxWinResub ($C_{ax}$, $V$, $N$, $nWinTfoLevs$, $nWinTfiLevs$)

7 return $C_{ax}$;
```

**Algorithm 2:** ApproxWinResub

```
Input: $C_{ax}$, current approximate circuit
$V$, the pivot node
$N$, the frame number of simulation $nWinTfoLevs$, maximum level of TFO cone of windows
$nWinTfiLevs$, maximum level of TFI cone of windows
Output: None

1  Update levels of each node in $C_{ax}$

2  // Logic simulation, generate approximate care set
3  Simulate ($C_{ax}$, $N$)

4  // From $V$, traverse its TFO cone, collect nodes satisfying one of conditions:
5  // (i) POs or nodes with many fanouts less than $nWinTfiLevs$ levels away from $V$
6  // (ii) $nWinTfoLevs$ levels away from $V$
7  $vRoots \leftarrow$ FindRoots ($C_{ax}$, $V$, $nWinTfoLevs$, $nFanoutMax$)

8  // from each node in $vRoots$, traverse their TFI cone, collect all nodes
9  $vTFICone \leftarrow$ FindTFICone ($C_{ax}$, $vRoots$)

10 // from each node in $vRoots$, traverse their TFI cones, collect all nodes that are:
11 // (i) PIs less than $nWinTfiLevs$ levels from $V$ (ii) $nWinTfiLevs$ levels from $V$
12 $vWinInputs \leftarrow$ FindWinInputs ($C_{ax}$, $vRoots$, $V.level - nWinTfiLevs$)

13 // PIs are divided into: (a) those in the TFI cone of the pivot node, and (b) the remainder.
14 // All nodes on paths between $V$ and the PIs of type (a) are added to the set of candidates,
     excluding the node itself and any node in the fanout free cone of $V$.
15 // Other nodes of in $vTFICone$ are added if they have no type (b) structural support.
     $vDivs \leftarrow$ FindDivisors ($C_{ax}$, $V$)

16 // build aig with approximate care set and divisors
17 $aig \leftarrow$ BuildAppAig ($vTFICone$, $vWinInputs$, $vDivs$, $N$)
18 // convert to cnf
19 $cnf \leftarrow$ AigToCnf ($aig$)
20 // create sat solver by adding condition of the existence of resubstitution
21 $sat \leftarrow$ CreateSolver ($cnf$, $vWinInputs$)

22 // simplify the circuit according to sat solver, I cannot understand proof of unsatisfiability
23 ResubNode ($C_{ax}$, $V$, $sat$)
```

**Algorithm 3:** BuildAppAig

```
Input: vTFICone, the TFI cone of vRoots
vWinInputs, window inputs
vDivs, divisor candidates
N, simulation frame number
Output: aig, an aig containing vTFICone and expressions of approximate care set

1  aig ← ∅
2  // add original function
3  forall node X in vTFICone do
4      transform X and add it into aig

5  // add approximate care set
6  for i = 1 to N do
7      pCare ← AigConstOne ()
8      forall node X in vWinInputs do
9          if value of X is 1 in the i-th frame then
10             pCare ← AigAnd (pCare, X)
11         else
12             pCare ← AigAnd (pCare, AigNot (X))
13     aig ← AigOr (aig, pCare)

14 // add divisors
15 Set nodes in vDivs as outputs of aig

16 return aig
```

**Choices in design**

Note: current parameters are presented in brackets.

1. Simulation
    - Number of simulation frame $N$ controls the error rate of circuit.
    - Various input combinations generate distinct resubstitution.
2. Window construction
    - $nWinTfoLevs$, maximum level of TFO cone. (2)
    - $nWinTfiLevs$, maximum level of TFI cone. (1-5)
    - $nFanoutMax$, maximum branch factor of TFO cone of windows. (30)
    - How to select divisors. (non-FFC nodes in $V$'s TFI cone, part of nodes in window but not in $V$'s TFI cone)
3. Don't-cares
    - Manipulate care set or don't care set. (care set)
    - Manipulate global set on PIs or local set on window inputs. (local set on window inputs)

- Select all cares or part of cares according to appearance probability. (all)

4. Simplification order of each node. (one by one)
5. Approach that coping with EXDCs. (mfs)

### Questions

1. Is mfs a state-of-the-art method for logic synthesis with EXDC?
2. Is it reasonable to insert care set on window inputs?
3. Is there any toy example for proof of unsatisfiability?
4. An interesting observation: I use the simulation results as approximate care set, when I increase simulation number, the circuit might be simplified significantly due to the addition of one simulation frame. Is it caused by the property of mfs?

### Explaination of that error rate is not monotonous to simulation number

When I increase simulation number, the size of approximate care set increases, while the size of approximate don't-care set decreases. Mfs only utilizes part of don't-cares to synthesize the circuit. It is possible that mfs utilizes more don't-cares after increasing simulation number.