Weekly Report

April 29, 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 Explain the influence of unbalanced number of window inputs

For different pivot nodes, if the local input number their windows differs greatly, we say that the window inputs are unbalanced. For unbalanced window inputs, we use same number of simulation patterns to approximate the local approximate local cares on window inputs. Since $S_d = 2^n - M$ (S_d is the size of approximate don't care set, n is the number of window inputs, M is the simulation number), it assigns much more don't cares for windows with smaller n, but less don't cares for windows with larger n. Thus, it induces smaller errors for windows with less inputs, but larger errors for windows with more inputs. However, the final error rate is dominated by larger errors induced by larger n. Consequently, we need to make the input number of windows closer to each other, in order to guarantee different windows induce closer errors.

A casual thought: why not select part of nodes instead of all nodes to approximate?

2.2 Balance the number of window inputs

For all windows, the input number of the window is less than n. To find less than n inputs, we keep a deque. Initially, the pivot node is in the double end queue. For each round, we pop the front node from the queue, and push the fanins of popped node to the back of queue. When the queue contains n elements, the pop and push process terminates. The final queue is the inputs of the window.

Here is the result of balancing the input number of windows. It seems that we eliminate the abrupt change of error rate. However, the area is still worse than Su's approach.

In addition, we do not list the result for simulation number less than 64, because there is something wrong in the program for that case.

Table 1: Area and error versus simulation number and window input number

		Approximate windows care set			reference (Su TCAD)	
circuit	n	#simulation	area	error	area	error upper bound
c880	30	128	527	0.0171875	519	0.03
		256	525	0.0149414	519	0.03
		512	562	0.00390625	551	0.005
		1024	577	0.00244141	554	0.003
	20	128	527	0.0171875	519	0.03
		256	525	0.0149414	519	0.03
		512	562	0.00390625	519	0.03
		1024	577	0.00244141	554	0.003
	10	128	533	0.0232422	519	0.03
		256	525	0.0149414	519	0.03
		512	562	0.00390625	519	0.03
		1024	577	0.00244141	554	0.003