A Stochastic Computational Approach for Accurate and Efficient Reliability Evaluation

A Python Implementation

Jake Humphrey

Department of Electronic and Electrical Engineering
Imperial College London
jbh111@ic.ac.uk

November 20, 2014

Reliability of Circuits

Gates in a logic circuit are susceptible to errors:

- **Stuck-At-One Error:** Output goes high.
- **Stuck-At-Zero Error:** Output goes low.
- **Von Neumann Error:** Output is inverted.

Masking Effects

However, the error may not affect the output due to one of the following *masking effect*s

- **Electrical Masking:** Error signal too weak to be detected.
- **Temporal Masking:** Error misses the detection window of a latch.
- **Logical Masking:** The error does not change the output of a logic gate.

Reliability Analysis Principles

Logical Masking is the most common, and so we try to analyse circuits on their ability to logically mask errors.

Probability of a signal = chance that it is logically True.
Reliability of a signal = chance that it takes the correct value

- Construct an error-prone representation of the circuit.
- Derive the *probabilities* of the outputs from the inputs.
- Find the output *reliabilities*.

Reliability Analysis Probabilistic Gate Models

However, existing algorithms are inefficient!

For example, *Probabilistic Gate Models* (PGMs) attempt to derive the output probabilities deterministically and analytically.

Reliability Analysis Probabilistic Gate Models

The problem occurs when the inputs to a gate are statistically dependent.

The PGM equations do not account for statistically dependent signals, and the solution involves splitting the circuit into two sub-circuits, doubling the cost of the algorithm.

Reliability Analysis Stochastic Computing

The use of *Stochastic Computing* can avoid these issues.

Generate input bitstreams and propagate them through the circuit.

The output probabilities can then be accurately calculated from the output bitstreams.

Reliability Analysis Stochastic Logic with Bernoulli Sequences

Existing Stochastic Logic algorithms use the input probabilities to generate *Bernoulli Sequences* of the form:

$$[X_0, X_1 \ldots X_{n-1}]; X_i \sim B(p)$$

for each input, where p is the input probability.

Requires n random numbers must be generated for each input!

Reliability Analysis Stochastic Logic with Non-Bernoulli Sequences

Non-Bernoulli Sequences reduce the random number generation overhead.

They are generated deterministically with the expected number of 1s, and then randomly permuted.

Only one random number generation is required per input bitstream!

Reliability Analysis Algorithm

Using Stochastic Computation with Non-Bernoulli Sequences, we arrive at an algorithm for Reliability Analysis, which I will describe over the following slides.

Reliability Analysis Algorithm Pseudocode

```
Data: Logic circuit to be tested
Result: Reliabilities for each output
for each gate in the circuit do
    represent gate in faulty circuit;
end
for each input to the faulty circuit do
    generate a non-Bernoulli sequence;
end
for each output in the circuit do
    for each input vector do
        if outputs are the same for each circuit then
            add 1/n to reliability of that output;
        end
    end
end
```

Reliability Analysis Algorithm Complexity Analysis

The most costly section is the final double loop!

We have to propagate a signal through each circuit once for each output, and n times.

Could be at worst $O(ogn)$, where:
o = number of outputs
n = length of Non-Bernoulli Sequences
g = number of gates

Reliability Analysis Algorithm Further Work

Still has room for improvement: inputs propagates through circuit once per output

It has to redundantly recalculate many intermediate values!

Calculating all the output values in one go would reduce the double loop to a single one over the length of the input sequences.

This reduces the algorithm complexity to $O(gn)$

Reliability Analysis Algorithm Runtime Analysis

Input Files	c17.v	c432.v
Inputs	5	36
Outputs	2	7
Gates	6	160
Input Sequence Length	runtime /s	runtime /s
1	0.00023	1.34
10	0.00067	12.2
100	0.00485	122
1 000	0.0473	1250
10 000	0.458	No Data
100 000	4.66	No Data

Reliability Analysis Algorithm Runtime Analysis