2A-4

実数値論理シミュレーションによる VLSI 故障検出用テストパターン生成

鈴木　五郎　　中村　匡利

北九州市立大学情報メデイア工学科

1　あらまし

実数値論理シミュレーションによる VLSI 故障検出用テストパターン生成手法の検討を行った。Cheng and Agrawal [2] が提案した手法を基本にした新たな algorithm を提案し、並列計算機上への implement を行った結果、乱数パターンと故障 simulation を単純に組み合わせた方法と比べて 64 倍程度の高速化を図ることができた。

2　はじめに

VLSI 故障検出用テストパターン生成方法において、deterministic な処理と heuristic な処理が存在する。前者としては、backtrack 処理の軽減を図った PODEM[1]、後者としては乱数パターンと故障 simulation を組み合わせた方法[2]などがあるが、100(%)の故障検出率を実現するにはいずれの方法でも非常に時間のかかる処理となっている。我々は後者の方法に注目するが、乱数パターンがある仮定故障を検出するテストパターンとなる最悪の確率は $1/2^n$ n:primary input 数 となることから、乱数パターンと故障 simulation を単純に組み合わせた方法では処理効率が悪い。そこで、Cheng and Agrawal [2] が提案した手法を基本に高速化の検討を行うことにした。さらに、並列計算機を用いることによる処理効率向上を検討する。

3　Cheng and Agrawal 法

論理値”0”を実数値 0.1、論理値”1” を実数値 0.9 に対応させ、gate ごとに用意した出力計算 table を用いることによって、出力信号を決定している。table の横軸は入力信号の平均値である。Fig. 1 の例では、論理値では AND gate の出力信号はいずれも”0”であるが、実数値を使った信号の計算、つまり実数値シミュレーションでは出力値に違いが現れ、「どの入力が該当 gate 出力信号の論理値を”1”にし易い状態なのか」を表現することができる。

Fig. 2 のように、故障有無 2 種類の回路に関

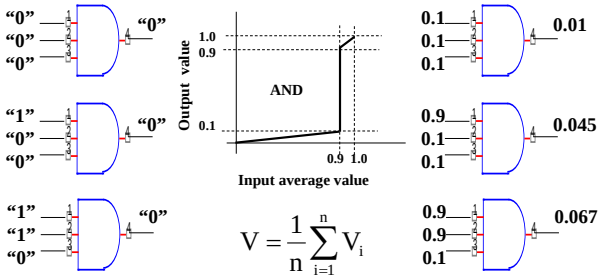


Fig.1 Cheng and Agrawal Method (1)

して primary output の実数値を計算し、その差を表現する cost があらかじめ設定したある threshold 値より小さくなった場合に primary input に与えた乱数が仮定故障を検出するテストパターンとなる。乱数の与え方としては、初期乱数から始めて、primary input の並び順に論理値を flip させる。ただし、cost が改善された場合であり、されない場合は flip させない状態に戻す。この処理を primary input 数繰り返す。

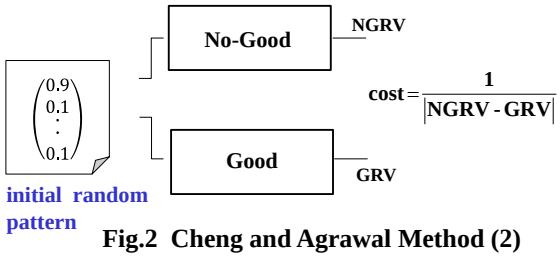
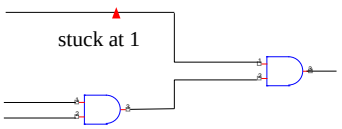


Fig.2 Cheng and Agrawal Method (2)

彼らの手順を適用させた結果、テストパターン生成に失敗する例を Fig.3 に示す。出力計算 table を用いる際、入力信号の平均値を取るところに失敗の理由がある。



initial random pattern	input	Good	No-Good	cost	
➡	“0,0,0”	0.0000	0.0560	17.86	
	“1,0,0”	0.0560	0.0560	∞	×
	“0,1,0”	0.0031	0.0590	17.89	×
	“0,0,1”	0.0031	0.0590	17.89	×

Fig.3 Failure Example by Cheng and Agrawal Method

4　提案する手法

Basic Study on Real Value Logic Simulation
Goro Suzuki　Masatoshi Nakamura
University of Kitakyushu

我々は Fig.4 に示した gate 種類毎に用意した式を用いて gate 出力信号の計算を行うことにした。故障有無 2 種類の回路に関して、その差 difference がある threshold 値よりも大きくなった場合に primary input に与えた乱数をテストパターンとして採用する。第 3 章で説明した同じ回路に関して、提案手法を適用した結果を Fig.5 に示すが、今度は成功していることが分かる。しかし実際の回路では、提案する手法でも初期乱数の与え方によりテストパターン生成に失敗する場合がある。その場合は初期乱数を変えて処理を繰り返すことにしている。提案する手法の処理手順を Fig.6 まとめる。

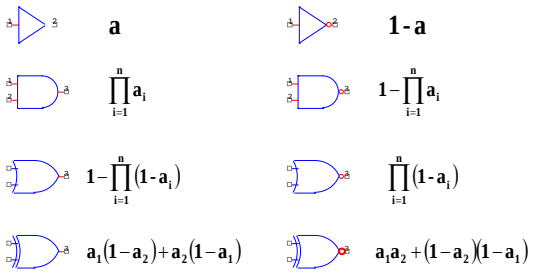


Fig. 4 Proposed Method (1)

	input	Good	No-Good	distance	
initial random pattern ➡	"0,0,0"	0.001	0.009	0.008	
	"1,0,0"	0.009	0.009	0.000	×
	"0,1,0"	0.009	0.081	0.072	○
	"0,1,1"	0.081	0.729	0.648	○

Fig. 5 Success Example by Proposed Method

(S1) Set up initial random pattern at primary input and calculate primary output difference
(S2) Select next downward input and flip the value and calculate primary output difference
(S3) If the difference becomes larger than the threshold then stop process
else if the difference becomes small comparing with the previous one,
then cancel the flip and go to (S2)
else if the input is the last one
then set up another initial random pattern and go to (S1)
else cancel the flip and go to (S2)

Fig. 6 Proposed Method Procedure

5 並列処理化

提案する手法を並列計算機 NVIDEA 社 GeForce GTX 650 1.1(GHz)上に implement し、Fig.7 のように、回路毎に故障有無回路 2 種類の実数値シミュレーションを 1 thread [3]に割り当て、複数の回路の複数の仮定故障に関するテストパターン生成を並列処理させることにした。

6 評価結果

乱数パターンと故障 simulation を単純に組み合わせた方法と提案手法の性能比較および提案手法を並列化した場合の性能比較を行った。Benchmark として用いた回路例を Fig.8 に示す。

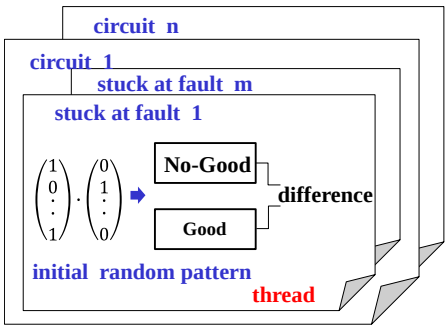


Fig. 7 Proposed Method (2)

これを含め、720 種類の回路に関して約 30k 種類の仮定故障検出用テストパターンの生成を行った。故障検出率 100(%)で、Table 1 のようにそれぞれ 64 倍と 5.3 倍の高速化が図れた。Host 計算機は Windows 7m, 2.2 (GHz)であり、表中 serial は host 計算機のみでの処理を行っている。

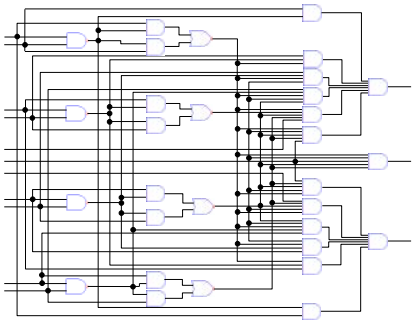


Fig. 8 Benchmark Circuit Example

Table 1 Acceleration

	serial	parallel
random pattern	377.9(s)	-
proposed method	31.0(s)	5.9(s)

7 まとめ

実数値シミュレーションを用いた VLSI 故障検出用テストパターン生成 algorithm を提案し、それを並列計算機上で動かすことで高速処理が可能になることを確認した。

参考文献

[1]藤原秀雄,"デジタルシステムの設計とテスト",工業図書,2004
[2]K.T. Cheng and V.D.Agrawal ,"A Simulation-Based Directed-Search Method for Test Generation",Proc. Of ICCD pp.48-51, 1987
[3]伊藤智義,"GPU プログラミング入門",講談社,2013