設計自動化 20－4
（1984. 2. 21）

高速コンカレントシミュレータ

広瀬文保
（ 株式会社富士通研究所 ）

概要 小文ではコンカレント故障シミュレータを高速化するための新しい故障伝播法を提案する。

シミュレーションに先立って，回路のゲートを，「単純ゲート」，「合流ゲート」と名付ける二つのクラスに分類することが本手法のキーポイントである。ファンアウトFにより分岐した信号の流れがゲートGにおいて再び合流する時，GをFの合流ゲートと呼び，そのようなFがない時Gを単純ゲートと呼ぶ。

シミュレーションに際しては，合流ゲートには従来のコンカレント法を適用するが，単純ゲートを新手法で迅速に処理することにより全体の高速化を図る。「他の故障リストへのポインタ」という新しいタイプの要素を持つ故障リストを導入することにより，故障リストが要する記憶容量を大幅に削減する。メモリのオーバフロー回避のための，故障の分割によるマルチパスシミュレーションの回数が減り，結果として速度が向上する。

①新しい手法の適用例，②単純ゲートの割合の調査，③〔従来／新手法〕を〔実行時間／メモリ容量〕について比較する。実験回路では，速度が1．8倍向上し，メモリが40%削減された。

1．序論

「テスト系列の検査率の解析」や「診断のための故障辞書作成」のために我々はしばしば故障シミュレータを用いる。シミュレータは与えられた様々の故障条件のもとで回路をシミュレーションし，出力端子の応答を正常回路と故障回路の出力の間で比較する。結果として，どの故障がどのテストパターンでどのように検査されるかを解析する。しかし，シミュレーションというのは実回路に比べて動作が遅く，かつ大規模な回路は極めて多くの独立な故障を持ちうるため，故障を非常に効率よく処理してゆくテクニックを開発する必要がある。

シミュレーション時間の短縮のために主に3つのテクニックが報告されている。それは，パラレル法〔SESHU 62〕，ディダクティブ法〔ARMSTRONG 72〕とコンカレント法〔ULRICH 73〕である。パラレル法の並列処理能力が使用計算機の語長に依存すること，ディダクティブ法において本質的である集合演算が多値論理や機能レベルシミュレーションに拡張する際非常にやっかいとなることを考えると，大規模回路にはコンカレント法が比較的むいていると思われる。〔BREUER 76〕〔樹下 81〕

しかしながら，コンカレント法にも，メモリ容量と計算効率の観点から，いくつかの問題点がある。コンカレント法のおもな弱点は次のようである。

(1)故障リストに必要なメモリ容量がシミュレーション中に動的に変化し，その最大値は極めて大きいがそれをあらかじめ予測することは難しい。

(2)特定の故障が評価中のゲートのどの入力端子に悪影響を及ぼすかを求めるため，故障リストの要素を順に検索する必要がある。

(3)結果として同じ入力パターンとなるゲートシミュレーションを何回も繰り返している。（評価中のゲートにおいて同等に悪影響を及ぼす故障をあらかじめ知ることが難しいため）

(4) 評価ゲートについての旧故障リストと新故障リストの内容が同じかどうかを比較する必要がある。（評価ゲートの出力は不変だが故障リストのみが変化する場合があるため）

小文の目的は，適用が簡単で計算時間とメモリ容量を大幅に削減する新しいテクニックを提案することにある。

2．基本概念

もし回路が木構造をしてれば，あるゲートの故障は他のゲートの高々一つの入力端子に悪影響を及ぼす。従って，評価ゲートの一つの入力端子に伝播してきている故障リストの要素の全てはゲート出力側のリストにそろって伝播さ

れるかされないかのいずれかである。故に，非常に簡単でかつ効率の良いテクニックがこのようなクラスの回路には適用できる。

>ゲート評価：ゲート評価の際に用いる真理値表の機に「故障伝播表」というものを用意し，入力パターンをアドレスとして両者を同時に参照する。故障伝播表とは，どの入力端子の故障リストが出力側に伝播するかを判断する情報を与えるもので，表の各行がそれぞれの入力ベクトルにアドレスという形で対応しており，行の各列はゲートの各入力ピンに対応している。（表１参照）もしｉ行ｊ列が「Ｐ」というエントリを持つならば，入力ベクトルｉが入力パターンのときｊ番目の入力端子の故障リストの要素が全てゲートの出力側のリストに伝播されることを表わす。もし，エントリが「－」ならば，対応する入力端子のリストの要素のどれもが出力側に伝播されないことを表わす。結果的に，従来のように故障リストの各々の要素について，同じものがないか他のリストを検索し，その後にゲートの評価を行なうという繰り返しが全く不要になる。

>故障リストの取り扱い：故障リストの要素として，「他の故障リストへのポインタ」も従来の要素である「故障ゲート」につけ加えて許すこととする。これにより，故障伝播表からアクセスした行のエントリのうちＰとなっている入力端子の故障リストのみをポインタとして評価ゲートの出力側の故障リストに登録すれば，伝播作業は終了する。入力側の故障リストを検索したり，要素を一つずつ出力側のリストに転送するといった手間はかからない。

>メモリ容量：我々の新しい故障リストでは，ある故障リストの要素の全てがたった一つのポインタという形で置き換わって伝播されるので，メモリ容量の最大値が大幅に削減される。

>故障イベントチェック：評価中のゲートの出力は不変だが，故障リストのみが変化する時，これを「故障イベント」が発生したと呼ぶことにする。従来手法においては，故障イベント発生の検査を行なうためには，評価ゲートの旧故障リストと新故障リストの要素を一々比較する必要があった。もし，故障イベントが発生しなかったとすると，〔新故障リストの要素数〕×〔旧故障リストの要素数〕分の要素の比較が必要ある。我々のプロセスではリスト比較は行わない。評価ゲートについては，前のネット値とともに前の「故障伝播マーク」（故障伝播表からアクセスした行のこと）も記憶されている。もし故障伝播マークが変化すれば，故障イベントが発生していることがわかる。もし，故障伝播マークが一緒ならば，イベントが入力した入力端子に対応する故障伝播マークのエントリがＰならば故障イベントが発生する。それ以外では故障イベントは発生しないことがわかる。従って，従来プロセスよりもかなり速く故障イベントのチェックができる。

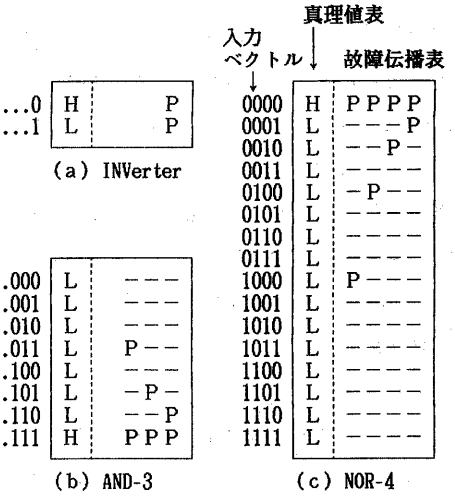

表１：故障伝播表

一般的には，故障の影響が同一ゲートの二つ以上の入力端子に悪影響を及ぼす場合もあるので，いままでに述べた方法をそのまま用いることはできない。しかし，かなりの割合のゲートが上記の処理を受け入れることができる。次章で詳述する。

３．単純ゲートと合流ゲート

ファンアウトＦで分岐した信号がゲートＧで再び合流す

る時，GをFの「合流ゲート」呼び，そのようなFがない時Gを「単純ゲート」呼ぶことにする。もしゲートGがファンアウトFに関する合流ゲートならば，Gの二つ以上の入力がFの故障の影響を受ける可能性がある。従って，前述の処理は適用せず，従来のコンカレント法をこのゲートには適用することにする。しかし，もしゲートGが単純ゲートならば，どのゲートの故障もGの高々一つの入力にしか影響を及ぼせない。即ち，これは木構造の回路にあるゲートと同様である。そこで，新手法を適用する。

図1は単純ゲートと合流ゲートの分類の例を示すための回路で，SN74LS280 パリティジェネレータである。全てのゲートには番号がふってある。ゲート11に着目すると，少なくとも 1→ 8→11と 1→ 9→11の二つのパスがあるため，ゲート11はファンアウト 1に関する合流ゲートである。同様にゲート41もファンアウト11，22，33，34，35と36に関する合流ゲートである。結果的にこの回路では，ゲート11，22，33，41と46の5個のゲートが合流ゲートで，他の41個のゲートは単純ゲートであることがわかる。つまり，89%のゲートが単純ゲートであるということになる。

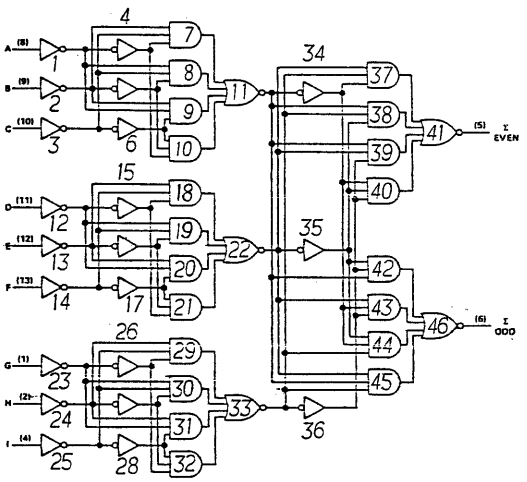

図1：例を説明する回路

分類のための一般的な手続きを表2に示す。手続きは，高々〔回路の入力端子数〕×〔ゲート数〕分のラベリングを必要とする。

```
PROCEDURE  CLASS

    PI≡set of primary inputs  ;
    PO≡set of primary outputs ;
    L  ≡set of labeled gates  ;
    S  ≡set of seed gates     ;
  REC≡set of reconvegent gates ;
    begin
    REC⇐φ ;
    for each input of PI do ;
     begin
      L⇐φ ; S⇐φ ; S←input ;
      while S≠φ do
      begin
        g←S ; S⇐S-g ;
        if g ¬in PO
           then if g in L then  REC←g
                else begin  L←g ;
                  S←fanouts of g end ;
      end
     end
    print REC
    end ;
```

表2：一般的分類手続き

表3は，ＴＩ社のデータブック〔TI 82〕よりいくつかの回路を摘出し，単純ゲートの割合を調査したものである。一般的な感覚での回路の複雑さが単純ゲートの割合を減少させているように思われる。しかしなお，かなりの数のゲートが単純ゲートとして期待できることがわかる。

回路名	機能	総ゲート数	単純ゲート数	割合(%)
'LS154	Decoder	25	25	100
'LS148	Encoder	30	27	90
'LS280	Parity Generator	46	41	89
'LS 85	Comparator	33	27	82
'LS283	Full Adder	35	27	77
'LS261	Multiplier	47	35	74
'LS157	Selecter	15	11	73
'LS670	Register File	66	46	70
'LS181	ALU	63	39	62
'LS 94	Shift Register	20	12	60
'LS169	Updown Counter	46	15	33

表3：単純ゲートの割合

4．新手法と従来手法の比較

図1の例を用いて，新手法と従来手法の比較を行なう。今，入力ベクトル（A,B,C,D,E,F,G,H,I）が（1,0,0,1,0,0,0,0,0）から（0,0,0,0,0,0,0,0,0）に変化したとする。シミュレータはちょうどゲート34とゲート35を前のタイミングで処理して今ゲート40を処理しようとしているとこ

ろであるとする。（ゲート 1～ゲート36が処理されたところとする。）図2（a）に示すように，ゲート34～ゲート36の新しい故障リスト FL34,FL35,FL36 がセットアップされたところである。一方，ゲート40はまだ古い故障リスト FL40 を持っている。今，我々は，ゲート40の新しい故障リスト FL40'を生成しようとしている。リストの要素を表わす記号について少々説明すると，ハイフンの前の番号はゲート番号を表わす。ハイフンの後の記号ＬとＨは，そのゲートの縮退故障の種類を表わす。例えば，1-L はゲート 1 の 0 縮退故障を表わす。

従来手法においては，まず第一にゲート40の正常時出力が入力ベクトル（0,0,0）で評価される。そして値 0 が変化しないことを確認する。第二に，故障伝播処理を開始する。一つの要素が FL34 から取り出される。その要素が他のリストに存在するか否かを確かめるため，FL35 と FL36 が検索される。この要素を含んでいるリストに対応する入力の値が反転された入力ベクトルでゲート40が評価される。出力が正常出力と比較され，もし異なっていればその

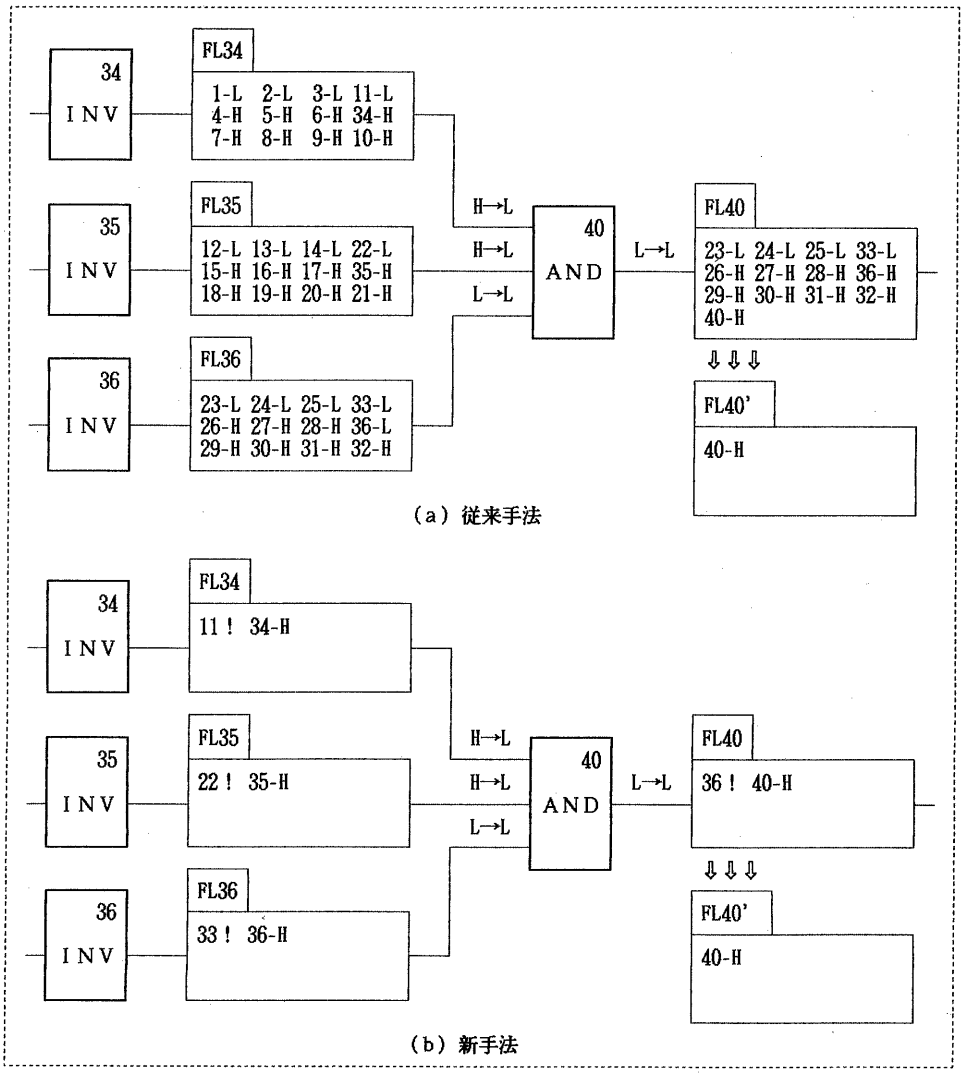

図2：手法の比較

要素はゲート40の新しい故障リスト FL40'に登録される。その要素を含む全リストよりその要素を取り除く。この処理を FL34, FL35, FL36 の要素が空になるまで繰り返す。この例の場合３６回のゲート評価とリスト検索が遂行されなければならない。とにかく，最後に我々は FL40'を生成する。その要素は 40-H のみである。

ゲート40の正常出力が変化しなかったので，故障イベントの発生の有無を確かめるため，新故障リスト FL40'と旧故障リスト FL40 の内容の同一性をチェックする必要がある。この例の場合，FL40' の要素数が１なので検査は容易である。しかし一般には，もし二つのリストが同一であるとすると〔新リストの要素数〕×〔旧リストの要素数〕分のマッチングが必要となる。

さて，新手法がどうなるか確かめよう。図２（ｂ）に（ａ）に対応する新手法の故障リストを載せる。図２（ｂ）のリスト FL34, FL35, FL36 の要素の表記法で，記号！はその要素がポインタであることを表わす。例えば， FL34 の要素11！は FL11へのポインタであって，FL11のどの要素も FL34 の要素であることを表わす。シミュレータはまず第一に入力ベクトル（0,0,0 ）に対応する真理値表と故障伝播表の行を同時にアクセスする。（表３参照）そして，正常回路の出力が０のままであることを確認する。故障伝播マークが（-,-,- ）であるのでどの故障リストも FL40' に伝播されない。従って，ゲート40の故障 40-H のみが FL40'の要素として生成される。故障イベントについては，旧故障伝播マーク（-,-,P ）と新マーク（-,-,- ）が異なるので故障イベントが発生したことがわかる。結局，たった一回のゲート評価と故障伝播マークの比較で故障伝播処理が終了したことになる。

さて，故障リストのためのメモリ消費量に注目してみよう。この例では，例の入力の場合，従来手法では２７９個の故障要素を記憶する必要があるが，新手法では１７５個の領域で済んでいる。言い換えれば，故障リストが６３％に削減されたということがわかる。

５．近似シミュレーションへの応用

回路が大規模すぎるため，従来の故障シミュレーションでは時間的に処理不能な場合でも，我々の提案する手法は以下に掲げる三つの理由により依然適用可能性があることを示してきた。

(1) 単純ゲートが極めて高速に処理される。

(2) 故障リストに必要なメモリ容量の最大値が大幅に削減される。従って，マルチパスシミュレーションの回数が減る。

(3) 合流ゲートの故障イベントチェックは必ずしも必要でない。

しかしながら回路が超大規模になると，我々の方法も現実的な時間内で処理しうるかどうかはわからなくなる。このようなとき，我々は近似シミュレーションを提案する。

近似の方法は，一部の合流ゲートを単純ゲートとみなすことによる。最も粗い近似は全てのゲートが単純ゲートであるとみなすことである。また他の近似では，マクロファンクション内における分類を回路全体における分類に適用するという方法も考えられる。他にも色々な近似法が，単純ゲートの数を増やすという観点から考えられる。実験結果では，近似シミュレーションの効果についても述べられる。

６．実験結果

新手法は，４ＭＩＰＳコンピュータ，ＦＡＣＯＭ Ｍ－１９０上にインプリメントされている。実験の対象回路として，構造が明らかな LS280パリティジェネレータと LS181ＡＬＵを取り上げた。ＣＰＵ時間と故障リストに必要な最大メモリ容量を以下の四つのシミュレーションモードで調査した。

(1)従来のコンカレント法　　(2)新しいコンカレント法

(3)近似シミュレーション　　(4)論理シミュレーション

実験回路としては，LS280 の３０個のコピーと，LS181 の２０個のコピーを使って，回路規模を１４００ゲート程度とした。これは，小規模回路ではサブルーチンコールのためのオーバヘッドが表面化するため必要であった。

表４が実験結果を表わす。表中，処理速度は正常回路に

おける１信号線の変化あたりに要する処理ＣＰＵ時間として計測してある。即ち，全ＣＰＵ時間を正常回路の信号変化数で割ることによって求めた。最大メモリ容量は，故障リストに必要なワード数（４バイト／ワード）の最大値として計測した。また近似シミュレーションでは，全てのゲートが単純ゲートであるとした。

実験結果によると，論理シミュレーションは１秒間に１万６千の信号変化を処理できる。故障シミュレーションでは，我々は故障ドロップなしで計測したが，従来手法で論理シミュレーションの５倍，近似手法で２．５倍の時間を要している。新手法の速度は，単純ゲートの割合によって変化する。８９％の単純ゲートをもつ LS280では，従来手法の４２％の時間を削減している。６２％の単純ゲートを持つ LS181では，２９％の時間を削減している。メモリ容量に着目すると，近似手法では従来手法の５０％の容量を削減している。新手法については，二つの回路とも約４０％の容量を削減している。プログラムは８Ｍバイトの０Ｓの制約の中で動き，また故障リスト以外に必要なメモリ量も少ないため，我々は８千ゲートの回路が故障ドロップなしにワンパスシミュレーション可能であると見積もっている。最近の並列処理化の傾向により我々は１メインクロック当り５０％のゲートを評価する必要があると推定する。この仮定によると，８０００ゲートのＣＭＯＳマスタスライスに対し１万パターンのテスト系列をシミュレーションするのに新手法で２時間，近似手法で１．５時間のＣＰＵ時間でよいと考えられる。一方，従来手法では，遅い処理速度とマルチパスシミュレーションのため８時間程度かかってしまうと考えられる。

７．まとめと結論

コンカレント法における高速な故障伝播テクニックについて報告した。我々の提案する新手法は故障リストに必要なメモリ容量を大幅に削減することを示した。従って，大規模な回路において，故障の分割やマルチパスシミュレーションの回数を減らすことができる。

単純ゲートが極めて高速に処理できることを示した。単純ゲートの故障伝播は，ゲートを論理シミュレーションすると同時にできる。従って，故障リストの検索や個々の故障のための評価の繰り返しが不要になる。

新手法の効果は，単純ゲートの割合に依存するが，かなりの割合の単純ゲートが一般回路に期待できることを調査によって明らかにした。

近似シミュレーションについても報告した。我々は，超大規模な回路においても，近似解が満足のゆく時間内で求まるものと期待している。

回路	比較項目	故障シミュレーション			論理シミュレーション
		従来手法	新手法	近似手法	
LS280 x 30 1499 gates	(1) 速度	303 μ	175 μ	130 μ	60 μ
		100 %	58 %	43 %	20 %
	(2) メモリ量	35,196 w	19,236 w	14,816 w	0 w
		100 %	55 %	42 %	0 %
LS181 x 20 1434 gates	(1) 速度	313 μ	223 μ	162 μ	62 μ
		100 %	71 %	52 %	20 %
	(2) メモリ量	36,376 w	20,936 w	17,592 w	0 w
		100 %	58 %	48 %	0 %

(1)正常回路の１信号線の変化を処理するのにようするＣＰＵ時間（μsec）

(2)故障リストのための最大所要メモリ量（words）

表４：実行速度・所要メモリ量の比較

参考文献

〔ARMSTRONG 72〕 ARMSTRONG,D.B., "A Deductive Method for Simulating Faults in Logic Circuits", IEEE Trans. on Computers, C-21, No.5, pp.464-471, 1972.

〔BREUER 76〕 BREUER,M.A. and A.D.FRIEDMAN, "DIAGNOSIS & RELIABLE DESIGN OF DIGITAL SYSTEMS", COMPUTER SCIENCE PRESS, INC., PP.224-248, 1976.

〔SESHU 62〕 SESHU,S. and D.N.FREEMAN, "The Diagnosis of Asynchronous Sequential Switcing Systems", IRE Trans. on EC, EC-11, pp.459-465, 1962.

〔TI 82〕 Texas Instruments, "The Bipolar DIgital Integrated Circuits Data Book for design engineers PART1", 1982.

〔ULRICH 73〕 ULRICH, E.G. and T.BAKER, "The Concurrent Simulation of Nearly Identical Digital Netwrks", Proc. D.A.Workshop, pp.145-150, June 1973.

〔樹下 81〕 樹下行三編, 「論理装置のＣＡＤ」, 情報処理学会叢書5, オーム社, pp.69-76, 1981年.