設計自動化 68－14
(1993. 10. 29)

入力／状態・並列故障シミュレーションを用いた順序回路のテスト生成について

児玉 剛　東 功　高松 雄三

愛媛大学工学部情報工学科

〒790 松山市文京町3

あらまし　入力並列および状態並列を併用した入力／状態・並列故障シミュレーション法を提案し, これを用いた順序回路のテスト生成について考察する. 提案する入力／状態・並列故障シミュレーションは, 順序回路の一つの状態から到達可能な状態を一度に拡大する入力並列故障シミュレーション (Phase1) と, 生成された回路の複数の状態を並列に処理する状態並列故障シミュレーション (Phase2) からなる.Phase2 の状態並列故障シミュレーションは, これまでにない新しい概念である. 提案する手法はコスト関数を導入することなく, 少ない処理でより多くの状態における故障シミュレーションによってテスト系列を生成しようとするものである. 本手法ではテスト生成能力を向上させるため,Phase1 と Phase2 を動的に切り替える手法を導入している. 提案するテスト生成法を ISCAS'89 のベンチマーク回路によって評価した結果は, 実用的な時間で多くの回路に対して高い検出率をもつテスト系列が生成されていることを示している.

和文キーワード　順序回路, テスト系列, テスト生成, 並列故障シミュレーション

Test Generation for Sequential Circuits Using Input/State • Parallel Fault Simulation

Tsuyoshi Kodama,　Isao Higashi　and　Yuzo Takamatsu

Department of Computer Science, Faculty of Engineering, Ehime University,

3bankyo-cho, Matsuyama, 790

Abstract　In this paper, we propose a new parallel fault simulator for sequential circuits, in which Phase 1 simulates 32 patterns at a time in order to expand the transition states from one state by a pattern parallel scheme, and Phase 2 simulates 32 states at a time. The state parallel scheme of Phase 2 is a new concept for a sequential parallel fault simulator. The proposed method generates a test sequence by simulating with as many states as possible and with less processes of simulation, without using a cost function. In order to do this efficiently, we introduced a dynamic exchange between Phase1 and Phase 2 during test generation. Experimental results show that our method achieves high coverage tests by acceptable CPU times.

英文 key words　sequential circuits, test sequence, test generation,parallel fault simulation

1. まえがき

論理回路の大規模,高集積化が進むにつれて,そのテスト生成はますます時間と経費を要する問題となっている.一般に論理 LSI は順序回路であり,そのテスト生成は組合せ回路に比べてさらに困難である.そこで,スキャンパス方式などのテスト容易化設計により,順序回路のテスト生成を組合せ回路の問題に変換して行っている.しかしながら,テスト容易化設計が行われていない順序回路も多くあり,順序回路の効率のよいテスト生成法を開発することは重要である.

これまで,テスト容易化設計が行われていない順序回路(以下,単に順序回路という)のテスト生成法として,D アルゴリズム[1],PODEM[2]などを適用したいくつかの手法が提案されている[3]-[6].これらの手法は,故障励起と伝搬,一致操作,および初期化の基本操作を反復回路モデルで行うものである.

また,故障シミュレーションの結果を利用して外部入力の変更を行いながら,テスト系列を生成する故障シミュレーション援用型テスト生成法が提案されている[7]-[10].これらは外部入力の故障に対するコスト関数を定義し,収束計算処理を行うことによりテスト系列を生成しようとする手法である.これらの手法の性能は,故障選択順序,コスト関数と故障シミュレーションの精度,および収束条件の判定法などに影響される.

筆者らは目標の故障に対して状態初期化系列を生成することなく,現在の状態を利用して外部入力をアルゴリズム的に決定する前方テスト生成法(FORTE[11],[12])を提案した.FORTE はテストが生成できない時刻における外部入力の決定法に問題が残っている.

さて,順序回路の故障シミュレーションに関する研究は古くから行われているが,最近,並列法[13]に基づく手法がいくつか提案されている[14]-[16].これらの中で PROOFS[14]はこれまで得られている故障シミュレーションの戦略を取り入れた高速化および使用メモリの効率化を図った順序回路の故障シミュレーションである.また,PARIS[15]は順序回路固有の時間系列を並列で行う故障シミュレーションであり,テスト系列が長い場合には PROOFS より速い結果が示されている.また,32 個の故障並列処理と部分樹状回路構造を利用した HOPE[16]が提案され,PROOFS より約 2 倍高速である結果が報告されている.

一方,筆者らは先に,入力並列および状態並列を併用した順序回路の入力/状態・並列故障シミュレーションを提案した[17].この入力/状態・並列故障シミュレーションは,順序回路の一つの状態から到達可能な状態を一度に拡張する入力並列故障シミュレーション (Phase1) と,生成された回路の複数の状態を並列に処理する状態並列故障シミュレーション (Phase2) から構成されている.Phase2 の状態並列故障シミュレーションは,これまでにない新しい概念である.

そこで,本稿では,この入力/状態・並列故障シミュレーションを用いた順序回路のテスト生成について考察する.一般に故障シミュレーションを利用する順序回路のテスト生成[7]-[10]は,コスト関数を用いて収束計算処理により回路の状態をテストが生成される方向へ導くことにより,テストを生成するという手法である.提案する手法はこのようなコスト関数を導入することなく,少ない処理でより多くの状態における故障シミュレーションによってテスト系列を効率よく生成しようとするものである.本手法ではそのテスト生成能力を向上させるために,Phase1 と Phase2 を動的に切り替える手法を導入している.

まず,**2.** で入力/状態・並列故障シミュレーションの概要について述べる.次に,**3.** で入力/状態・並列故障シミュレーションを用いたテスト生成の効率向上のために,Phase1 と Phase2 を動的に切り替える二,三の手法を考察し,これらを適用したテスト生成処理の流れを述べる.最後に **4.** で,提案したテスト生成法をベンチマーク回路に適用した実験結果を示す.実験では,未知の状態から初期状態を設定する場合,および初期状態が特定の状態に設定可能な場合について行った.その結果は,実用的な時間で多くの回路に対して高い検出率をもつテスト系列が生成されていることを示している.

2. 入力/状態・並列故障シミュレーション[17]

入力/状態・並列故障シミュレーションで扱う各信号線の信号値は 0,1,X の 3 値である.また,各信号線の信号値を 2 語 V0,V1 の各ビットで図 1 のように表す[14].このように表した信号値の論理演算は表 1 で与えられる[14].

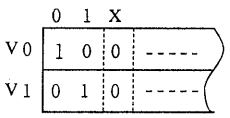


図 1. 信号値 0,1,X のビット表現

表 1. 論理演算表

	V0	V1
AND	A0 | B0	A1 & B1
OR	A0 & B0	A1 | B1
INV	A1	A0
XOR	(A0 & B0) | (A1 & B1)	(A0&B1) | (A1 & B0)

入力/状態・並列故障シミュレーションは,正常の順序回路の一つの状態から遷移する状態を一度に拡大する入力並列シミュレーション,生成された複数の回路の状態を並列に処理する状態並列シミュレーション,および入力並列単一故障伝搬法[18]から構成される.

[Phase1:入力並列シミュレーション]

Phase1 はある一つの状態 S_n に対して複数の入力パターンを印加することによって,S_n から遷移する状態を一度に拡大させる処理である.

従って,初期状態を未知の状態から設定する場合,1 時刻目は図 2 で示すように回路の状態を表すすべてのビットに未知の状態 S_x を設定し,ビットごとに異なった 32 個の

ランダム入力 T_1を入力並列として並列シミュレーションを行う.

また, ある時刻において Phase1 を実行する場合は, 一つのある状態 S_iをすべてのビットに設定し, ビットごとに異なった 32 個のランダム入力を入力並列として並列シミュレーションを行い, 状態 S_iから遷移する状態を一回の処理で拡大する.

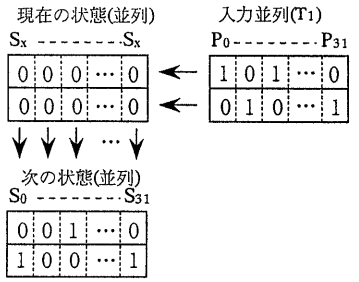


図 2. Phase1 における状態（並列）とそれに対する入力並列の例

[Phase2 : 状態並列シミュレーション]

Phase2 は Phase1 の入力並列によって遷移した複数の回路の状態を並列に処理する.Phase2 の概念を図 3 に示す. 図 3 では, ビット表現された 32 個の状態 S_0,..S_{31}にビット表現された 32 個のランダム入力 T_2を印加し, 状態並列シミュレーションを行ったとき, それぞれ S_0^1,..S_{31}^1に遷移した例を示している.

Phase2 では,32 個の状態を並列に処理するが,32 個の状態が必ずしも異なるとは限らない. そこで,Phase2 においても少ない処理でより多くの状態におけるシミュレーションを行うことができるように, 状態とともに入力における並列処理も行う. 従って厳密には,Phase2 は入力および状態に対する並列シミュレーションであるが, ここでは Phase2 を状態並列シミュレーションと呼んでいる.

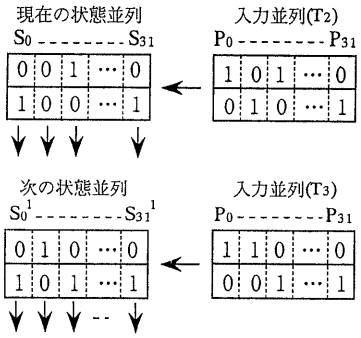


図 3.Phase2 における状態並列とそれに対する入力並列の例

[入力／状態・並列故障シミュレーションの概要]

入力／状態・並列故障シミュレーションは, 図 4(a) で示すように Phase1 の入力並列故障シミュレーションから始め,Phase1 で拡大された状態に対して, 以下 Phase2 を繰り返し実行する.

図 4(b) に Phase1 および Phase2 における回路の状態遷移の様子を示している. ここで,Phase1 の 1 時刻目は回路の状態を未知の初期状態 S_xに設定し, ビットごとに異なった 32 個のランダム入力の入力並列シミュレーションを示している. また,2 時刻目の Phase2 は,Phase1 で拡大された 32 個の状態 S_0,..S_{31}にビット表現された 32 個の入力 T_2を印加したとき, それぞれ S_0^1,..S_{31}^1に遷移した状態並列シミュレーションの例を示している.

なお, 本稿の入力／状態・並列故障シミュレーションにおける故障の挿入・伝搬は, 入力並列単一故障伝搬（PPSFP[18]）の手法と同様の単一故障伝搬法を用いている.

入力／状態・並列故障シミュレーションにおける Phase1 と Phase2 は, 処理する回路の状態を異にするのみで基本的には同様の処理であり, その実現は容易である.

以下,Phase1 の処理する回路の状態を Phase2 のそれと特に区別するとき,Phase1 の回路状態という.

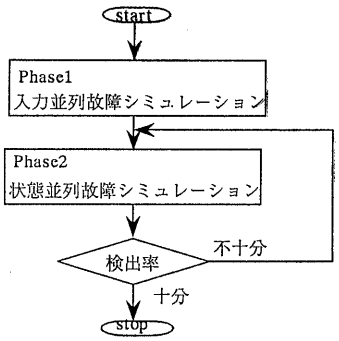


図 4.(a) 入力／状態・並列故障シミュレーションの流れ

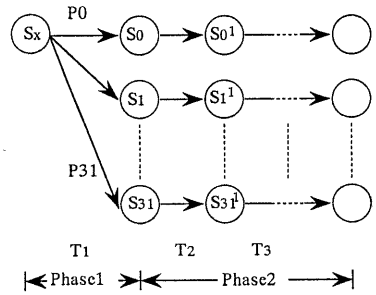


図 4.(b)Phase1 および Phase2 における状態遷移の概念図

図 4. 入力／状態・並列故障シミュレーション

3. 入力／状態・並列故障シミュレーションを用いたテスト生成法

2. で述べた入力／状態・並列故障シミュレーションを用いたテスト生成について考察する. この故障シミュレーションは図 4(b) に示したように, 少ない処理でより多くの状態における故障シミュレーションを行うことに特徴

がある. しかしながら,Phase1 および Phase2 で遷移する状態は, 必ずしもすべて異なる状態には遷移しない. 従って,Phase2 の処理を繰り返し行うと, ある特定の状態の間でループを構成する場合が生じる. このような状態に陥いると, 故障シミュレーションによるテスト系列の生成は効率よく行うことができない. このような例は処理回数と検出率の関係を表す図（例えば後述の図 8(a),(b) の実線）から見ることができる. すなわち, 図 8(a)s208 および (b)s526n について, それぞれ検出率が 43.26%および 9.40%で飽和している.

以下本稿では, このような状態を「不適切状態」という. この不適切状態を検出, 回避するため,Phase2 の処理において検出される故障が 0 である状態が, あらかじめ決めた回数発生したとき, 不適切状態であると判定し Phase2 から Phase1 へ切り替える. これはある一つの状態をすべてのビットに設定する Phase1 の回路状態で,Phase1 の処理を行うことにより, 一回の処理で新しい複数の状態を生成して, その状態から抜け出すことを試みるためである. この Phase2 から Phase1 への切り替え手法について次に述べる.

3.1 Phase の動的切り替え手法

不適切状態に陥ったとき, この状態から抜け出す Phase1 の処理を実行するため,Phase2 で遷移した正常回路の状態に対して, ある規則に従って一つの状態を選択する. このとき, 次の規則 1 あるいは規則 2 に従ってこれまで遷移した状態の中で, 最も少ない状態と考えられる状態を選択する. これは不適切状態から抜け出すためには, これまで遷移した状態の中で最も遷移しづらいと思われる状態を選択するすることにより, その状態から Phase1 を実行すれば新しい状態が得られる可能性が高いと考えられるからである.

こうして, 選択した状態を Phase1 の回路状態を表す語の 0~31 ビットすべてにセットし,Phase2 から Phase1 へ切り替えることによって, 少ない処理でより多くの状態における故障シミュレーションでテスト系列を生成することを試みている. 図 5 に Phase2 から Phase1 への切り替えの様子を示す.

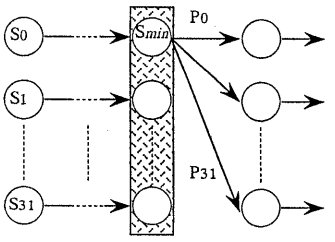


図 5. Phase2 から Phase1 への切り替え

[規則 1:出現数最少の状態を選択]

Phase2 で遷移した最後の正常回路における 32 個の状態の中で, 最も少ない状態を一つ選択する.

[規則 2:記憶素子の状態優先による状態選択]

Phase2 で遷移した数時刻間における各記憶素子の状態から計算される評価値 (ZO) と, その最後に遷移している正常回路における 32 個の状態に対して, それぞれ計算される値 (W_j) の中で, 最大値を有する一つの状態を以下の手順で選択する.

[1] Phase2 で遷移した状態に対して, さらに数時刻間 Phase2 の処理を繰り返し, 各記憶素子における状態 0 および 1 の値に対して, 状態 0 には負の値（例えば,−1）を, 状態 1 には正の値（例えば,+1）を割り当て, その数時刻間における和を各記憶素子の評価値 (ZO) とする. このようにして求めた ZO は, それが負（正）であれば, その記憶素子は 0(1) に遷移する可能性が 1(0) に遷移する可能性よりも高く, またその絶対値が大きいほど 0(1) に遷移する可能性が高いことを示している.

[2] m 個の記憶素子 FF_i,i=1~m, の評価値を ZO_iで表す. また, 最後に遷移している正常回路における状態 S_j,j=1~32, の m 個の記憶素子 FF_iを FF_{ij}で表す. このとき, 状態 S_jに対して次の W_jを定義する.

$$W_j = \sum_{i=1}^{m} W_{ij}$$

ただし, W_{ij}= ZO_i(FF_{ij}=0)
W_{ij}= −ZO_i(FF_{ij}=1)

図 6 に W_j,W_{ij}の計算例を示している. この例では,FF_1(FF_m) は 0(1) に遷移する可能性が高いことを示しており, また,FF_2は 1 よりは 0 に遷移するする可能性が高いが,FF_1よりは低いことを表している.

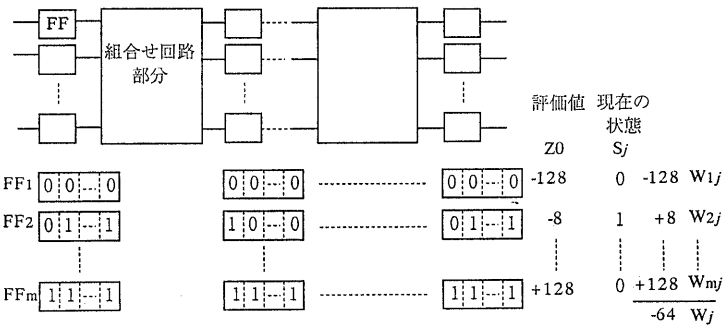


図 6. 記憶素子部分の状態優先による状態選択の例

[3] 最後の32個の状態S_jに対するW_jを計算し，その最大の値を持つ状態S_jを選択する．

以上の手順で選択される一つの状態S_jは,Phase2で遷移した数時刻間における状態と最も距離の大きい（これまで遷移したことが少ない）と考えられる状態を表す一つの尺度である．

上記の規則1および2は，正常回路の状態に基づくものであるが，次の規則3は故障回路の状態に基づくものである．

[規則3:故障が伝搬している記憶素子の数が最多である状態を選択]

Phase2で遷移した最後の故障回路における32個の状態の中で，故障が伝搬している記憶素子の数が最多である状態を一つ選択する．

この規則3は多重時刻における故障伝搬を優先する状態選択の一つである．

本稿のテスト生成では，上記のいずれか一つの規則を用いて,Phase2の最後の32個の状態の中からある一つの状態を選択した．

3.2 テスト生成法

本稿で提案する入力／状態・並列故障シミュレーションを用いたテスト生成は，まずPhase1を未知（あるいは，設定可能）の初期状態に適用して遷移する状態を一度に拡大し，次にその状態に対してPhase2を数時刻間実行する．さらにPhase2で不適切状態が検出されたとき,**3.1**で述べた手法に従って選択した一つの状態に対してPhase1を行い，その状態に対してPhase2を実行する．以下同様の処理を繰り返す．提案するテスト生成の手順を述べると次のように要約できる．

[入力／状態・並列故障シミュレーションによるテスト生成]

(1) 時刻iを1として,Phase1の回路状態を設定する．

(2) ランダムに生成した32個の入力並列パターンT_iを作る．

(3) 32個の入力並列パターンで正常回路の並列シミュレーションを行う．

(4) 未検出の故障を一つ選び単一故障伝搬法による入力並列故障シミュレーションを行う．検出された故障を除き，以下この処理(4)を挿入可能なすべての未検出故障について繰り返す．

(5) 不適切状態ならば(6)へ．そうでなければ，時刻iをi+1と更新して，状態選択回数$\leq N_{min}$のとき，入力並列パターンは$T_i+1 \leftarrow T_i$として(3)へ行き，そうでなければ(2)へ行く．

(6) 不適切状態であることが検出され，かつ状態選択回数$\leq N_{max}$ならば,**3.1**で述べた規則によりPhase1のための状態を一つ選択する．時刻を更新してPhase1の回路状態を設定し(2)へ行く．そうでなければ，処理を終了する．

上記の手順(5)において，不適切状態でない場合，入力並列パターンを変更しないで数時刻間故障シミュレーションを行う戦略を導入している．これは回路構造に依存するが，ある特定の外部入力に0または1の値を連続して印加すること（連続した同一の入力）によって，回路の状態が不適切状態に陥ることが避けられる場合があり，テスト生成に有用であると考えたからである．

以上の手順の詳細をフローチャートとして示したものが，図7である．

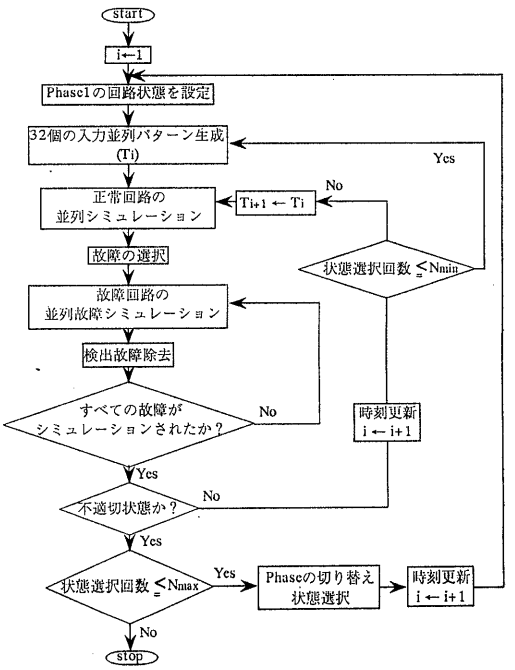


図7. 入力／状態・並列故障シミュレーションによるテスト生成手順

4. 実験結果

3.2の入力／状態・並列故障シミュレーションを用いたテスト生成法の評価を行うため，その手法をUNIX上のC言語を用いてプログラム化し,ISCAS89ベンチマーク回路[19]に適用して実験を行った．使用計算機はSUN SPARC station IPXである．なお,**3.2**の手順における不適切状態の検出は，検出される故障が連続して10時刻得られない場合とした．また，入力並列パターンの更新のためのパラメータN_{min}，およびPhase2からPhase1への切り替え回数N_{max}は，それぞれ,1および5としている．

[初期状態を未知状態とするテスト生成実験]

まず，回路の初期状態が設定できない場合について評価する．すなわち，回路の初期状態を未知状態としてテスト

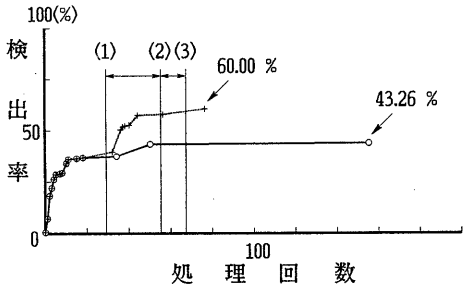
生成を行う実験である.

表 2, 表 4 は, それぞれ規則 1, 規則 3 を適用した場合の結果である. また, 表 3 は 1 回目の状態選択にのみ, 規則 1 を適用した後で規則 2 を適用し, その後は規則 2 を適用した場合の結果である. ここで, 表では左から回路名, 代表故障数, 検出率 (%), テスト系列数および生成時間 (sec) を表している. なお, テスト系列数はその検出率が得られたときのそれであり, 生成時間はそのときまでの入力並列パターンを生成する時間も含めた合計のテスト生成時間である.

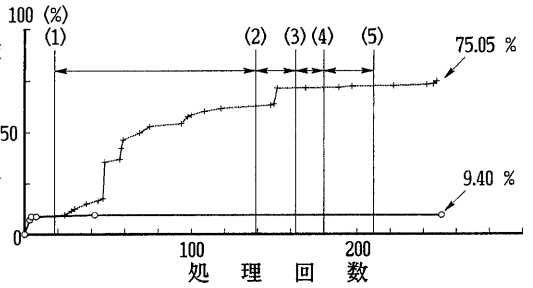
また, 表 2 の最右列の Step1 および Step2 は, それぞれ, 図 7 の 1 回目の不適切状態の検出 (N_{min}=1) まで, およびそれ以降のテスト生成処理で得られた故障検出率 (%) である. この 1 回目の不適切状態の検出 (N_{min}=1) までの結果は, 通常の故障シミュレーションによるテスト生成で得られる故障検出率に相当する. 表 2 の実験結果によると,s444,s820,s832,s1423 を除くほとんどの回路に対して, 文献 (9) とほぼ同等の検出率をもつパターンが高速に生成されていることを示している. また,s382,s400 などでは, より高い検出率およびより短いテスト系列数が得られている.

また, 直接比較することはできないが,PARIS[15] の 4096 個のランダム入力に対する故障検出率, 生成時間は, 回路 s208,s382,s526 について, それぞれ,43.26%(2.50sec),13.28% (15.15sec),9.37%(28.50sec) であるが, 本手法では表 2 から, 60.00%[36.28%](9.32sec), 88.72%[12.28%](56.63sec), 75.05%[8.68%](106.52sec) の結果が得られている. ここで, [%] は図 7 の 1 回目の不適切状態の検出 (N_{min}=1) までの結果であり, これらが PARIS の結果にほぼ同等となっている.

我々の手法のねらいの一つは, 前述したように入力並列および状態並列を動的に切り替えることにより, 少ない処理回数で多くの状態における故障シミュレーションを行うことでテストを効率よく生成しようとするものである. そこで, この効果について Phase を切り替えない通常の故障シミュレーションと比較するため, その処理回数と故障検出率について s208 および s526n に対して調べた. その結果を図 8 に示している.



(a) s208



(b) s526n

図 8. 処理回数と検出率

ここで, 図の (1),(2),... は,Phase2 から Phase1 への切り替えを表している. この例のように通常の処理ではテスト系列が得られないときでも, 本手法ではテスト系列が生成されている.

また, 表 3 の結果と表 2 の結果を比べると, ほとんどの回路に対して同等であるが,s444 および s1423 に対する検出率が, それぞれ 86.08%および 68.84%となり, これらの回路に対しては規則 2 が有用であることを示している. 表 3 と CRIS[10] の結果を比較すると, s1423 を除いた多くの回路で, ほぼ同等またはそれ以上の検出率が得られている.

また, 表 4 の結果では,s1423 に対して検出率の改善が得られている.

一般に, 用いた規則の有効性は回路に対して異なるため, 最適の規則を定めることは困難である.

表 2. 規則 1 による実験結果

回路名	代表故障数	検出率 (%)	テスト系列長	CPU Time (Sec)	Step1/Step2
s208	215	60.00	319	9.32	36.28/23.72
s298	308	84.42	871	15.88	54.87/29.55
s349	350	95.71	255	7.50	95.14/0.57
s382	399	88.72	996	56.63	12.28/76.44
s386n	384	77.60	1255	13.57	54.69/22.92
s400	424	87.50	997	65.67	12.03/75.47
s420	430	40.70	579	35.93	26.05/14.65
s444	474	59.07	160	63.03	11.18/47.89
s526n	553	75.05	1402	106.52	8.68/66.37
s641n	467	85.44	834	23.77	84.80/0.64
s713n	581	81.07	853	31.33	80.55/0.52
s820	850	55.65	1906	131.28	42.71/12.94
s832	870	54.48	1908	146.10	41.84/12.64
s838	857	28.70	291	124.48	23.69/5.02
s953	1079	8.25	13	244.00	8.25/0.00
s1196	1242	94.44	2600	91.85	94.12/0.32
s1238	1355	89.37	3617	140.58	89.00/0.37
s1423	1515	53.40	2450	833.07	43.83/9.57
s5378	4603	68.04	2769	3934.37	65.13/2.91

表 3. 規則 2 による実験結果

回路名	本手法			文献 (10)		
	検出率 (%)	テスト系列長	CPU Time (sec)	検出率 (%)	テスト系列長	CPU Time* (sec)
s208	60.00	280	9.82	60.9	715	22.8
s298	85.06	478	11.28	82.1	476	16.2
s349	95.71	259	8.47			
s382	88.72	1277	58.27	68.6	1230	16.7
s386n	74.74	870	14.15			
s400	87.50	1296	61.67	84.7	758	35.3
s420	39.77	354	35.58	33.7	857	65.4
s444	86.08	1077	66.58	83.7	519	34.2
s526n	76.85	1116	96.07			
s641n	85.44	798	27.00			
s713n	81.07	817	36.02			
s820	56.24	1933	144.80	53.1	1381	135.8
s832	55.17	2079	148.18	42.5	1328	105.1
s838	29.05	740	189.63	28.2	1078	216.8
s953	8.34	31	341.77			
s1196	95.25	3137	94.15	95.0	2744	97.0
s1238	89.96	3053	147.32	90.7	4313	131.1
s1423	68.84	5909	982.58	77.0	2696	580.5
s5378	65.76	3553	6333.92	65.8	1255	948.6

* SUN SLC workstation with 16Meg

表 4. 規則 3 による実験結果

回路名	検出率 (%)	テスト系列長	CPU Time (Sec)
s208	57.21	264	9.86
s298	85.06	696	14.17
s349	95.71	266	7.57
s382	59.15	191	50.10
s386n	60.16	651	15.28
s400	58.49	192	53.57
s420	39.77	302	38.70
s444	50.00	177	63.25
s526n	49.01	132	76.48
s641n	85.65	935	24.07
s713n	81.24	954	32.02
s820	45.18	921	139.27
s832	44.25	923	147.28
s838	28.59	240	132.90
s953	8.25	13	240.07
s1196	94.44	2600	90.73
s1238	89.37	2617	141.78
s1423	72.41	3126	792.67
s5378	67.85	2581	3872.07

[初期状態が設定可能な場合のテスト生成実験]

ここでは, 回路の状態がテスト生成時に, ある特定の初期状態に設定できると仮定した場合のテスト生成に対する評価を行う. すなわち, 回路はリセット, あるいはセットが可能であり, テスト生成時にすべての記憶素子が, それぞれ 0 あるいは 1 に設定できる場合である. 表 5(a),(b) に, それぞれ初期状態を 0 および 1 とした場合の実験結果を示している. なお, この実験における Phase2 から Phase1 への切り替えには規則 1 を用いた. また, 表 5(a) には, 文献 (20) で得られている結果と併せて示している.

文献 (20) のリセット可能な場合におけるテスト生成の結果と表 5(a) を比較すると,s820 および s832 を除いて, テスト系列は長くなるがほぼ同等の検出率が得られており, 本手法の生成時間は s953 を除いて文献 (20) のそれに比べてかなり高速である.

文献 (20) の検出率は冗長故障を除いた結果であり, 本手法で得られた検出率がそれらと同等であることは, 残された未検出故障の多くが冗長であると思われる.

また, 表 5(a),(b) を比べると, 初期状態が 1 にセットできる場合の方が,s420,s444 を除いて検出率がおおむね高く, 特に s208,s526n, および s838 では検出率が高いのが顕著である. このことからも, 順序回路のテスト生成における初期状態は, そのテスト容易化に大きな影響を与えることが分かる.

表 5. 初期状態が設定可能な場合における テスト生成結果

表 5(a) 初期状態 0

回路名	本手法			文献 (20)		
	検出率 (%)	テスト系列長	CPU Time (sec)	検出率 (%)	テスト系列長	CPU Time (sec)
s208	66.78	330	5.73	70.51	195	12.5
s298	87.01	682	8.67	87.66	280	17.7
s349	98.00	210	3.13	97.90	120	16.9
s382	89.22	988	35.55	90.73	1633	66(m)
s400	87.97	986	38.48	89.86	409	60(m)
s420	46.05	363	20.27	47.44	808	22(m)
s444	88.40	1392	50.03	88.39	994	1.66(h)
s526n	68.54	399	82.27	76.85	399	52(m)
s820	55.76	1875	133.12	95.83	1304	6(m)
s832	54.60	1869	140.53	93.85	1344	6(m)
s838	35.12	517	96.22	36.63	290	75(m)
s953	98.33	1050	249.98	99.10	1050	86
s1196	94.44	2600	89.98	98.71	545	3.4(h)
s1238	89.37	2617	139.32	93.96	576	3(h)
s1423	55.31	2629	753.00	56.43	4026	9(h)
s5378	71.39	2091	2876.20	69.0	1037	10(h)

m : minute　　h : hour

表 5(b) 初期状態 1

回路名	検出率 (%)	テスト系列長	CPU Time (Sec)
s208	75.81	605	5.28
s298	87.34	768	8.22
s349	98.57	247	3.25
s382	89.97	660	38.23
s400	87.97	662	41.23
s420	39.77	302	38.70
s444	87.55	1010	41.78
s526n	79.20	851	54.88
s820	56.82	1465	123.38
s832	56.21	1595	129.58
s838	50.53	511	78.53
s953	98.33	1652	50.48
s1196	94.44	2617	86.48
s1238	89.37	2259	136.20
s1423	55.31	2629	753.00
s5378	73.82	2163	2837.62

5. むすび

本稿では,入力並列および状態並列を併用した順序回路の並列故障シミュレーションを用いたテスト生成について考察した.利用した故障シミュレーションは,順序回路の遷移する状態を一度に拡張する Phase1 の入力並列故障シミュレーションと,生成された回路の状態を並列に処理する Phase2 の状態並列故障シミュレーションを行う新しい並列故障シミュレーションである.また,このテスト生成能力を向上させるために Phase1 と Phase2 を動的に切り替える手法を述べた.最後に,提案した手法をベンチマーク回路に適用した実験結果を示し,実用的な時間で多くの回路に対して高い検出率をもつテスト系列が生成されることを示した.

今後,Phase の切り替え方法および状態の選択規則の組合せなどを考察して,提案した順序回路の故障シミュレーションによるテスト生成法の性能を改良する予定である.

参考文献

(1) J.P.Roth:"Diagnosis of automata failures: A calculus and a method", IBM J. Res. Develop., 10, pp.278-291(July 1966).

(2) P.Goel:"An implicit enumeration algorithm to generate tests for combinational logic circuits ", IEEE Trans. Comput., C-30, 3,pp.215-222 (March 1981).

(3) H-K.T.Ma, S.Devadas, A.R.Newton, and A.Sangiovanni-Vincentelli:"Test generation for sequential circuits," IEEE Trans. on Comput. -Aided Design, CAD-7, 10, pp.1081-1093(Oct. 1988).

(4) T.P.Kelsey and K.K.Saluja:"Fast test generation for sequential circuits," Proc. Int. Conf. CAD, ICCAD-89, pp.354-357(Nov. 1989).

(5) E.Auth and M.H.Schulz:"A Test-Pattern-Generation Algorithm for Sequential Circuits", IEEE Design & Test of Computers, Vol.8, No.2, pp.72-86(June 1991).

(6) D.H. Lee and S.M. Reddy:"A new test generation method for sequential circuits", Proc. Int. Conf. CAD, ICCAD-91, pp.446-449(Nov. 1991).

(7) K-T.Cheng, V.D.Agrawal, and E.S.Kuh:"A simulation-based method for generating tests for sequential circuits," IEEE Trans. on Comput., C-39, 12, pp.1456-1463(Dec. 1990).

(8) V.D.Agrawal, K-T.Cheng, and P.Agrawal:"A directed search method for test generation using a concurrent simulator," IEEE Trans. on Comput. -Aided Design, 8, 2, pp.131-138(Feb. 1989).

(9) 彦根, 池田, 畠山, 林:"実数値シミュレーションを利用した順序回路テスト生成", 信学論 (D-I), Vol. J75-D-I, No. 11, pp.1089-1098(1992-11).

(10) D.G. Saab, Y.G. Saab, and J.A. Abraham:" CRIS: A test cultivation program for sequential VLSI circuits", Proc. Int. Conf. CAD, ICCAD-92, pp.216-219(Nov. 1992).

(11) 高松, 小川, 高橋:"順序回路の前方テスト生成に対する一手法", 信学論 (D-I), Vol. J75-D-I, No. 9, pp.864-873(1992-09).

(12) Y.Takamatsu, T. Ogawa, and H. Takahashi:" Improved forward test generation of sequential circuits using variable-length time frames", IEICE Trans. INF. & SYST., Vol. E76-D, No. 7, pp.832-836(July 1993).

(13) E.W. Thompson and S.A.Szygenda:"Digital logic simulation in a time-based, table-driven environment, Part 2. Parallel fault simulation", IEEE Computer, 8, 3, pp.38-49, March 1975.

(14) T.M. Niermann, W.-T. Cheng, and J.H. Patel: "PROOFS:A fast, memory efficient sequential circuit fault simulator", Proc. 27th DAC, pp. 535-540, June 1990.

(15) N. Gouders and R. Kaibel:"PARIS:A parallel pattern fault simulator for synchronous sequential circuits", Proc. Int. Conf. CAD, ICCAD -91, pp.542-545, Nov. 1991.

(16) H.K.Lee and D.S.Ha:"HOPE: An efficient parallel fault simulator for synchronous sequential circuits", Proc. 29th DAC, pp.336-340, June 1992.

(17) 児玉, 東, 高松:"順序回路の並列故障シミュレーションに関する一考察", 第 29 回 FTC 研究会資料,(1993-07).

(18) J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy: "Fault simulation for structured VLSI", VLSI System Design, pp.20-32, Dec. 1985.

(19) F. Brglrz, D. Bryan, and K. Kozminski:"Combinational profiles of sequential benchmark circuit 1989 Int. Sympo. Circuits & Systems, pp.1927-1934(May 1989).

(20) A. Ghosh, S. Devadas, and A.R. Newton: "Sequential logic testing and verification", Kluwer Academic Publishers, pp.11-55, 1992.