時間展開モデルを用いた無閉路順序回路のテスト系列圧縮方法

細川　利典[†]　井上　智生[††]　平岡　敏洋[†*]　藤原　秀雄[††]

Test Sequence Compaction Methods for Acyclic Sequential Circuits Using a Time Expansion Model

Toshinori HOSOKAWA[†], Tomoo INOUE[††], Toshihiro HIRAOKA[†*], and Hideo FUJIWARA[††]

あらまし　無閉路順序回路に対するテスト系列は，時間展開モデルを用いて生成することができる．本論文では，時間展開モデルを用いて生成されるテスト系列は（1）テスト系列長が一定である，(2）各外部入力に対する未定義値 (X) が存在する位置がテスト生成の対象故障とは無関係に決まる，という性質に着目し，静的圧縮，動的圧縮の二つのテスト系列圧縮方法を提案する．まず，テスト系列の値に依存しないテンプレートを用いた静的テスト系列圧縮方法を提案する．また圧縮後のテスト系列を逆変換したテストパターンで，時間展開モデルに対して故障シミュレーションを実行する逆変換故障シミュレーションによる動的テスト系列圧縮方法を提案する．いくつかの実際の回路にパーシャルスキャン設計を適用して作成した無閉路順序回路で本提案方法を評価した結果，テスト系列長を 19〜34%に削減することができた．

キーワード　時間展開モデル，無閉路順序回路，テスト系列圧縮，テンプレート，逆変換故障シミュレーション

1. まえがき

近年の LSI の大規模化，複雑化により，LSI のテスト設計の自動化が必要不可欠となっている．LSI のテスト設計のコストを考えた場合，

（1）　高い故障検出効率（例，95%以上）を達成するテスト系列を生成する時間

（2）　高い故障検出効率を達成するテスト系列の長さ，すなわち LSI テスタでのテスト時間

の 2 点を挙げることができる．

一般の順序回路に対して現実的な時間で高い故障検出効率を達成することは非常に困難なので，(1）に関するテスト設計コストを削減するためには，フルスキャン設計 [1], [2] やパーシャルスキャン設計 [3]〜[9] に代表されるテスト容易化設計が必要である．パーシャルスキャン設計では回路中のどのフリップフロップ（FF）をスキャン FF に置き換えるかということが重要な問題である．無閉路順序回路は組合せテスト生成アルゴリズムでテスト生成可能であることが知られている [6]〜[9]．したがって核回路（スキャン FF を取り除いた回路）が無閉路順序回路になるようにスキャン FF を選択する [6]〜[9] ことで，フルスキャン設計に比べて小さいハードウェアオーバヘッドで高い故障検出効率を得ることができる．特に文献 [9] では，時間展開モデルを用いた無閉路順序回路のテスト系列生成方法が提案されいる．

また，(2）に関するテスト設計コストを削減する解決策として，テストパターンの圧縮技術がある．

このテストパターン圧縮技術に関して近年数多くの論文 [10]〜[19] が発表されている．文献 [10]〜[13] は組合せ回路を対象にしたテストパターン圧縮の方法，文献 [14]〜[19] は順序回路を対象にしたテスト系列圧縮の方法を提案している．

本論文では，文献 [9] で提案された時間展開モデルを用いて得られたテスト系列を圧縮する方法を提案する．時間展開モデルを用いて得られるテスト系列には，生成されたテスト系列の値に依存しない規則があることに着目し，次の二つの圧縮方法を提案する．

（1）　静的圧縮方法

† 松下電器産業株式会社半導体開発本部，守口市
Corporate Semiconductor Development Division, Matsushita Electric Industrial Co., Ltd., 3–1–1 Yagumo-Nakamachi, Moriguchi-shi, 570–8501 Japan
†† 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, 8916–5 Takayama-cho, Ikoma-shi, 630–0101 Japan
* 現在，京都大学大学院情報学研究科

時間展開モデルに対する基本テンプレートを作成し，テスト系列長を最小にするテンプレートを求め，そのテンプレートを用いる静的圧縮方法

（2） 動的圧縮方法

静的圧縮後に得られるテスト系列から，まだ故障シミュレーションが行われていないテスト系列を時間展開モデルのテストパターンとして抽出する逆変換故障シミュレーションによる動的圧縮方法.

本論文は次のような構成になっている．まず，**2.** で時間展開モデルを用いた無閉路順序回路のテスト系列生成方法について述べる．**3.** では，テンプレートを用いた無閉路順序回路の静的なテスト系列圧縮方法について述べる．また **4.** では，逆変換故障シミュレーションによる動的圧縮方法について述べる．**5.** では，実際の回路を用いた実験結果を示し，その考察を行う．**6.** では，本論文のまとめと今後の課題について述べる．

2. 時間展開モデルを用いた無閉路順序回路のテスト系列生成方法

2.1 テスト生成

時間展開モデルを用いた無閉路順序回路のテスト系列生成方法 [9] について，例を用いて説明する．図 1 は無閉路順序回路 S の例を示す．図 1 において，FF1〜FF11 は FF，1〜9 は組合せ論理部，PI1〜PI3 は外部入力，PO は外部出力を示す.

図 2 は図 1 の無閉路順序回路 S の時間展開モデル $C(S)$ を示す．時間展開モデル $C(S)$ は S の各外部出力から外部入力まで，入力方向に組合せ論理部を展開した組合せ回路である．$C(S)$ の組合せ論理部，外部入力，外部出力はラベル t をもつ．時間展開モデル $C(S)$ において，ラベル t の組合せ論理部 a からラベル $t+1$ の組合せ論理部 b にデータが入力されているときには，対応する無閉路順序回路 S において組合せ論理部 a からのデータが一つの FF を通過して，1 時刻遅れて組合せ論理部 b に到着することを表している．図 2 において，時間展開モデル $C(S)$ の上部に書かれている数は，その列に存在する組合せ論理部，外部入力，外部出力のラベルを表す．また，組合せ論理部の黒塗りの部分は，外部出力又は他の組合せ論理部の入力のいずれにも到達不可能である信号線又は論理ゲートを表し，時間展開モデルから除去されている.

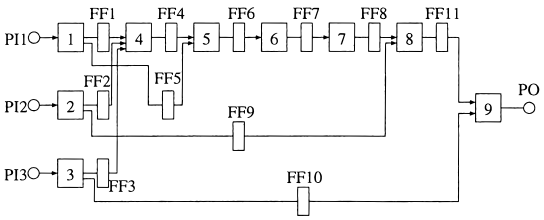

図 1 無閉路順序回路 S
Fig. 1 Acyclic sequential circuit: S.

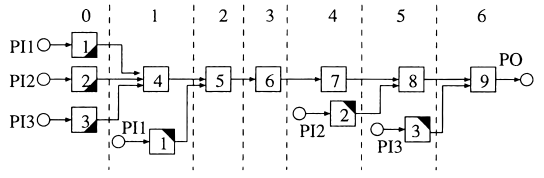

図 2 S の時間展開モデル：$C(S)$
Fig. 2 Time expansion model of S: $C(S)$.

無閉路順序回路 S の故障 f_a に対応する時間展開モデル $C(S)$ の故障 f_e は，故障 f_a の存在する組合せ論理部と対応する $C(S)$ の各組合せ論理部の同じ位置（信号線）に存在する一つの多重故障となる．例えば，図 1 の無閉路順序回路 S の組合せ論理部 3 中の故障は，図 2 の時間展開モデル $C(S)$ に示すように組合せ論理部 3 が二つ存在するので，時間展開モデル $C(S)$ では一つの 2 重故障として扱う．ここで，無閉路順序回路 S の故障集合を F_a, F_a に対応する時間展開モデル $C(S)$ の故障集合を F_e としたとき，以下のことが成り立つ [9].

（1） 無閉路順序回路 S の任意の故障 $f_a \in F_a$ について，f_a に対応する時間展開モデル $C(S)$ の故障 $f_e \in F_e$ がただ一つ存在する.

（2） 故障 $f_a \in F_a$ に対するテスト系列が存在するとき，かつそのときに限り，故障 f_a に対応する故障 $f_e \in F_e$ に対するテストパターンが存在する.

（3） 故障 $f_e \in F_e$ に対するテストパターンは，対応する故障 f_a に対するテスト系列に変換可能である.

すなわち，時間展開モデル $C(S)$ の故障に対して生成したテストパターンは無閉路順序回路 S のテスト系列に変換可能で，その変換したテスト系列で対応する無閉路順序回路の故障を検出することができる．よって，時間展開モデル $C(S)$ に対して組合せテスト生成（多重故障対応）を適用し，テストパターンを生成することができる．その後で，時間展開モデル $C(S)$ で生成したテストパターンを $C(S)$ の各外部入力の存在するラベルの値を参照して，無閉路順序回路 S のテスト系列に変換する.

図 2 の時間展開モデル $C(S)$ のある故障に対して生

表 1　S のテスト系列
Table 1　Test sequences for S.

(a) Test sequence T_1

Time（時刻）	$PI1$	$PI2$	$PI3$
0	0	1	0
1	1	X	X
2	X	X	X
3	X	X	X
4	X	1	X
5	X	X	0
6	X	X	X

(b) Test sequence T_2

Time（時刻）	$PI1$	$PI2$	$PI3$
0	0	0	1
1	1	X	X
2	X	X	X
3	X	X	X
4	X	0	X
5	X	X	1
6	X	X	X

成されたテストパターンを ($PI1(0)$, $PI2(0)$, $PI3(0)$, $PI1(1)$, $PI2(4)$, $PI3(5)$) = (0, 1, 0, 1, 1, 0) とすると（$PIi(t)$: ラベル t に存在する外部入力 PIi の値），回路 S における時刻 0 の PI1 の値は 0，時刻 0 の PI2 の値は 1，時刻 0 の PI3 の値は 0，時刻 1 の PI1 の値は 1，時刻 4 の PI2 の値は 1，時刻 5 の PI3 の値は 0 となり，無閉路順序回路 S のテスト系列は表 1 (a) に示されるテスト系列に変換される．

2.2 圧　　縮

時間展開モデル $C(S)$ で生成された二つのテストパターン t_1 = (0, 1, 0, 1, 1, 0), t_2 = (0, 0, 1, 1, 0, 1) を無閉路順序回路 S のテスト系列に変換した二つのテスト系列をそれぞれ T_1, T_2 とする．テスト系列 T_1 を表 1 (a) に，テスト系列 T_2 を表 1 (b) に示す．テスト系列 T_1 中に未定義 (X) の部分が存在するので，テスト系列 T_2 はテスト系列 T_1 の時刻 2 から入力できる．したがって，表 2 に示すようにテスト系列 T_1 と T_2 を圧縮したテスト系列 T を生成することができる．

また，テスト系列 T_1, T_2 に示すように，テスト系列において 0 又は 1 に値が決定している箇所と X である箇所は，すべてのテスト系列について一定である．この情報から複数のテスト系列が圧縮可能か否かは，テスト系列中の値に関係なく決定できる．よって，テスト系列が生成される前の段階で，静的に圧縮方法が決定でき，圧縮時に高速にテスト系列を圧縮することができる．この静的圧縮方法については次の **3.** で述べる．

表 2 に示すテスト系列 T に着目すると，まだ X の部分が残っているので，表 3 に示すように，X の部分に対してランダムに 0 又は 1 の値を設定したテスト系列 T' を生成する．この T' において，例えば，時刻 1 から時刻 7 のテスト系列に着目すると，テスト系列 T_1, T_2 とは別のテスト系列であることがわかる．このテスト系列を無閉路順序回路 S に対して故障シミュレーションを実行すれば，新たに未検出故障を検出できる可能性がある．また，そのテスト系列を時間展開モデル $C(S)$ のテストパターンに逆変換すると，($PI1(0)$, $PI2(0)$, $PI3(0)$, $PI1(1)$, $PI2(4)$, $PI3(5)$) = (1, 0, 1, 0, 0, 0) になり，このテストパターンで時間展開モデル $C(S)$ に対して，故障シミュレーションを実行することもできる．時間展開モデル $C(S)$ は組合せ回路なので，故障シミュレーションの実行速度は無閉路順序回路 S に比べて速くなる．この逆変換故障シミュレーションによる動的圧縮方法については **4.** において述べる．

表 2　圧縮されたテスト系列：T
Table 2　Compacted test sequence: T.

Time（時刻）	$PI1$	$PI2$	$PI3$
0	0	1	0
1	1	X	X
2	0	0	1
3	1	X	X
4	X	1	X
5	X	X	0
6	X	0	X
7	X	X	1
8	X	X	X

表 3　X に 0，1 を設定したテスト系列：T'
Table 3　Test sequence after setting 0 or 1 to Xs in T: T'.

Time（時刻）	$PI1$	$PI2$	$PI3$
0	0	1	0
1	1	0	1
2	0	0	1
3	1	1	0
4	0	1	0
5	1	0	0
6	0	0	0
7	1	1	1
8	0	0	1

3. テンプレートを用いた無閉路順序回路の静的テスト系列圧縮方法

本章では，テンプレートを用いた無閉路順序回路のテスト系列の圧縮方法について述べる．

3.1 諸　定　義

まず，無閉路順序回路のテスト系列圧縮方法を説明するために必要な用語を定義する．

［定義 1］（テンプレート） 外部入力 (P_0, P_1, ..., P_{w-1}) をもつ回路に対するテスト系列 T を考える（w は外部入力数）．T のテスト系列長を l とする．T の時刻 t における外部入力 P_i の値を $T(t,i)$ と表記する．テスト系列 T における $T(t,i) = 0$ 又は 1 を満たすすべての値 ($0 \leqq t < l, 0 \leqq i < w$) を b に置き換えてできるテスト系列 T' を，テスト系列 T のテンプレートという．また，X 又は b からなるテスト系列を単にテンプレートという．また，T' のテスト系列長をテンプレートの長さ又はテンプレート長といい，（テンプレート T' 中に b が存在する最大時刻）−（b が存在する最小時刻）+1 をテンプレート T' の有効長と

表 4　演　算 $\cap_c$
Table 4　Operation $\cap_c$.

$\cap_c$	b	X
b	ϕ	b
X	b	X

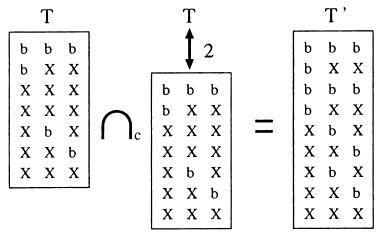

図 3　スキュー 2 で圧縮可能な例
Fig. 3　Example that T is compatible with skew 2.

いう．　□

［定義 2］（圧縮可能） 外部入力 $(P_0, P_1, \ldots, P_{w-1})$ をもつ回路に対する二つのテンプレート T_1, T_2 を考える（w は外部入力数）．T_1, T_2 のテンプレート長をそれぞれ l_1, l_2 とする．T_1, T_2 の時刻 t における外部入力 P_i の値をそれぞれ $T_1(t,i), T_2(t,i)$ と表記する．任意の $i(0 \leqq i < w)$ について，次の二つの条件のうちいずれかを満たす非負の整数 k が存在するとき，T_2 は T_1 にスキュー k で圧縮可能であるという．

（1）　$k \geqq l_1$

（2）　$k \leqq t < \min\{l_1, k+l_2\}$ となる任意の t について，$T_1(t,i) = X$，又は，$T_2(t-k,i) = X$ となる．

また，$T_1 = T_2$ のとき，$T_1(T_2)$ はスキュー k で圧縮可能であるという．　□

T_2 は T_1 にスキュー k で圧縮可能であるときに，実際に圧縮して生成されるテンプレート T は表 4 に示す演算 $\cap_c$ を用いると，任意の $i(0 \leqq i < w)$ について，以下の式で表すことができる．

$$T(t,i) = \begin{cases} T_1(t,i) & 0 \leqq t < \min\{l_1, k\}, \\ & k + l_2 \leqq t < l_1 \\ T_1(t,i) \cap_c T_2(t-k,i) & \\ & k \leqq t < \min\{l_1, k+l_2\} \\ X & l_1 \leqq t < k \\ T_2(t-k,i) & \max\{l_1, k\} \leqq t < k + l_2 \end{cases}$$

例 1：図 3 は T がスキュー 2 で圧縮可能であるときに，圧縮して生成したテンプレート T' を示している．

［定義 3］（基本テンプレート） 無閉路順序回路を S，S の時間展開モデルを $C(S)$，$C(S)$ に対して得られたテストパターンを t とする．ただし，テストパターン t には未定義を表す X は含まれないものとする．このとき，テストパターン t を S のテスト系列 T に変換し，テスト系列 T のテンプレートを T' とする．このテンプレート T' を時間展開モデル $C(S)$ に対する基本テンプレートという．　□

表 5　図 2 の基本テンプレート
Table 5　Primitive template for Fig. 2.

Time（時刻）	PI1	PI2	PI3
0	b	b	b
1	b	X	X
2	X	X	X
3	X	X	X
4	X	b	X
5	X	X	b
6	X	X	X

例 2：図 2 の時間展開モデルの基本テンプレートを表 5 に示す．

3.2　問題の定式化

時間展開モデル $C(S)$ で生成されるすべてのテストパターン数を n とすると，n 個のテストパターンをそれぞれ無閉路順序回路 S のテスト系列に変換し，n 個のテスト系列を圧縮して，全体のテスト系列長を最小化することが，理想的な目標である．しかしながら，n 個のテスト系列を圧縮して，全体のテスト系列長を最小にすることは，計算量が膨大となるために困難であると考えられる．

そこで本節では，ある一定長のテンプレート中に，最大個の基本テンプレートを圧縮して，かつテンプレートの有効長を最小にする部分問題を解き，全体のテスト系列長を短縮する．この部分問題を解く圧縮方法は **4.** で述べる動的圧縮方法と組み合わせることができ，更に圧縮が可能となる．ある一定長のテンプレート中に，最大個の基本テンプレートを圧縮して，かつテンプレートの有効長を最小にする最適化問題の定式化を行う．この定式化を行う前に，基本テンプレートがスキュー k で圧縮可能であることを表現する圧縮可能グラフを以下に定義する．

［定義 4］（圧縮可能グラフ） 圧縮可能グラフは無向グラフ $G = (V, E, t)$ であり，頂点 $v \in V$ は，基本テンプレートを表し，各頂点にはラベル $t : V \to Z+$（$Z+$ は非負の整数を表す）が付けられている．また $\forall u, v \in V$ $(u \neq v)$ において $t(u) \neq t(v)$ が成り立つ．辺 $(u,v) \in E$ は，基本テンプレートはスキュー $|t(u) - t(v)|$ で圧縮可能であることを表す．　□

圧縮可能グラフからクリーク（部分完全グラフ）を

形成する頂点の集合を取り出し，頂点の個数の基本テンプレートを各頂点のラベル t の値をスキューとして圧縮を行い，圧縮に用いるテンプレートを生成する．このとき，できるだけ数多くの頂点を取り出せば，圧縮効率の高いテンプレートになる．このことは以下の最大クリーク（頂点数が最大であるクリーク）抽出問題 [20] としてとらえることができる．

［最大クリーク抽出問題］

入力：圧縮可能グラフ $G(V, E, t)$

出力：クリークサイズ $|C|$ が最大となる C のうち，$|\max t(v) - \min t(u)|$ が最小となるクリーク C．ただし，$v, u \in C$ かつ $v \neq u$ で，$\max t(v)$ は最大のラベル値，$\min t(u)$ は最小のラベル値である．

例 3：図 4 に示す圧縮可能グラフのクリークについて考察する．基本テンプレートは表 6 に示す．図 4 において，三つのクリーク $C1$, $C2$, $C3$ を考えてみる．

$C1 = \{0, 1, 7\}$, $C2 = \{0, 1, 2\}$, $C3 = \{0, 1\}$ である．$C1$ と $C2$ は最大クリークである．$C1$ からテンプレートを生成する．三つの基本テンプレートをスキュー 1,7 で圧縮することにより生成されるテンプレートの有効長は 13 となる．ここで，時間展開モデルで生成される総テストパターン数を n 個とするとき，$C1$ から生成したテンプレートを用いて圧縮を行うと，無閉路順序回路の全体のテスト系列長は $13n/3$ となる．同様にして，クリーク $C2$, $C3$ から得られるテンプレートの有効長はそれぞれ 8,7 となり，また，全体のテスト系列長はそれぞれ $8n/3$, $7n/2$ となる．

$C1$ と $C2$ のそれぞれの無閉路順序回路におけるテスト系列長を比較すると，$C2$ が $C1$ よりも短いことがわかる．このことは，| 最大ラベル値 − 最小ラベル値 | の比較において，$C1$ が $|7-0|$, $C2$ が $|2-0|$ となることから明らかである．また，$C2$ と $C3$ を比較した場合，$C2$ の方が無閉路順序回路の全体のテスト系列長が短いことがわかる．よって，| 最大ラベル値 − 最小ラベル値 | が最小となる最大クリークを求めると，圧縮効率の高いテンプレートが生成できることがわかる．

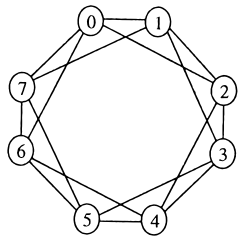

図 4　表 6 の圧縮可能グラフ
Fig. 4　Compatibility graph for Table 6.

表 6　基本テンプレート
Table 6　Primitive template.

Time（時刻）	PI1	PI2	PI3
0	b	b	b
1	X	X	X
2	X	X	X
3	X	b	X
4	b	X	X
5	X	X	b

3.3　ヒューリスティックアルゴリズム

本節では，圧縮に用いるためのテンプレートを生成するヒューリスティックアルゴリズムについて述べる．圧縮可能グラフから最大クリークを抽出するヒューリスティックとして，圧縮可能グラフの辺 (u, v) に対して以下のようにして求められる重み $w(u, v)$ を付けることにする．頂点 u, v の隣接頂点集合を $nbr(u)$, $nbr(v)$ とし，$nbr(u) - v$, $nbr(v) - u$ をそれぞれ A, B とする．そして，$|A \cup B| - |A \cap B|$ を重み $w(u, v)$ とする．この重みが表す数字は，頂点 u, v をクリークに加えたときに，頂点 u, v の隣接頂点の中で，クリークに加えることができなくなる隣接頂点の数になる．

図 5 に与えられた圧縮可能グラフから最大クリークを抽出するヒューリスティックアルゴリズムを示す．図 5 の関数 Extract_max_clique で説明すると，まず 0 のラベルが付いた頂点 v をクリークの頂点集合 C に初期値として加える．次に C に含まれる各頂点 u

```
Extract_max_clique(G)              /* G = (V, E, t) */
{
    C = { v s.t. t(v) == 0};
    while((S = Candidates(C)) ! = ϕ){
        v=Best_vertex(S,C) ;
        C = C ⋃ {v};
    }
    return C ;
}
Candidates (C)
{
    S = { v | v∈ V - C ∧ ∀ u ∈ C, ∃ (u, v)∈ E};
    /* S is a set of vertices s.t. any vertex in C is compatible with them */
    return S;
}
Best_vertex(S, C)                  /* heuristics */
{
    V1 = { v | Σ w(u, v) is minimum in S, u ∈ C};
    V2 = { v | |nbr(v)| is maximum in V1 };
    V3 = {v | t(v) is minimum in V2};
    return (one vertex selected from V3);
}
```

図 5　テンプレート生成ヒューリスティックアルゴリズム
Fig. 5　Heuristic algorithm of a template generation.

の隣接頂点集合の積集合を求め，それを新たに S とする（関数 Candidates）．S が ϕ（空集合）になるまで，S の探索と S の中から一つの頂点 v を選択し，C に挿入する処理を繰り返す．この「S の中から一つの頂点 v を選択する」ときには，候補となる頂点の集合 S の各頂点 v について，以下の（1）から（3）の三つの評価尺度を求め，ただ一つの頂点を選択できるまで（1）から優先的に用いる．すなわち,（1）を満たす頂点がただ一つ存在するならばその頂点を選択し，複数あるならばその中で（2）を満たす頂点を選択する．更に（2）を満たす頂点が複数あるならば，その中で（3）を満たす頂点を選択する（関数 Best_vertex）．

（1） 頂点 v とクリークの頂点集合 C にあるすべての頂点 u との間にある辺 (v,u) に付けられた重み $w(v,u)$ の総和が最も小さい頂点を選択する（頂点集合 $V1$）．重みの総和が最小となる頂点を選択することで，C に加えられる可能性がある頂点数が増加し，結果として最終的に得られる C の頂点数が多くなる．すなわちテンプレートに圧縮される基本テンプレート数が多くなる可能性がある．

（2） 頂点集合 $V1$ において，隣接頂点数が最も多い頂点を選択する（頂点集合 $V2$）．

隣接頂点数が最大の頂点を C に加えることで,（1）と同様にテンプレートに圧縮される基本テンプレート数が多くなる可能性がある．

（3） 頂点集合 $V2$ において，ラベル t が最小となる頂点を選択する（頂点集合 V3）．

ラベル t が最小となる頂点を選択することで，生成されるテンプレートの有効長を短くできる可能性がある．

以上の手順に従ってクリーク C を抽出した後，C に含まれる頂点数の基本テンプレートを，C に含まれる頂点の各ラベル値をスキューとして圧縮することで，無閉路順序回路の圧縮に用いるテンプレートを生成する．ATPG 時には，生成したテンプレートを繰り返し使用する．また，テンプレートは一つ前に使用したテンプレートの有効長の時刻から重ねて置くことが可能である．

例 4：表 5 の基本テンプレートをスキュー 9 まで考慮に入れた圧縮可能グラフ（頂点数 10）を図 6 に示す．また，図 6 の圧縮可能グラフについて，図 5 に示したヒューリスティックアルゴリズムを用いて求めたクリークは $\{0,3,6,9\}$ となり，図 7 に示すテンプレートが圧縮に用いるテンプレートとして生成される．

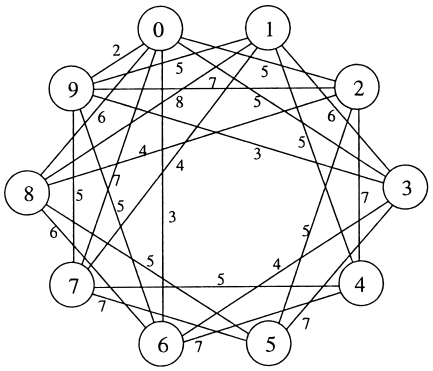

図 6　表 5 の圧縮可能グラフ
Fig. 6　Compatibility graph for Table 5.

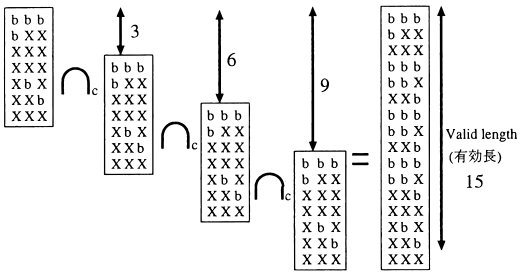

図 7　図 6 から求めたテンプレート
Fig. 7　Template generated from Fig. 6.

4. 逆変換故障シミュレーションによる無閉路順序回路の動的圧縮方法

図 8 に，テンプレートを用いた静的テスト系列圧縮方法と逆変換故障シミュレーションによる動的テスト系列圧縮方法を含んだ無閉路順序回路のテスト系列生成アルゴリズムを示す．

まず，与えられた無閉路順序回路 S の時間展開モデルを生成する．続いて，圧縮可能グラフからテンプレートを生成する．テンプレートは N 個の基本テンプレートを圧縮して生成されたとする．つまり，時間展開モデル上で生成された N 個のテストパターンは，テンプレートを参照してテスト系列の定められた位置に無条件に変換（圧縮）できることを意味する．しかし，テンプレートにおいて，その N 個のテストパターンが変換される部分以外は故障シミュレーションを実行していないことになる．そこで,「時間展開モデルの未検出故障 f に対してテストパターン生成 → 時間展開モデル上で故障シミュレーション → 未検出故障集合 F から検出故障を削除 → 無閉路順序回路のテスト系列に変換」という操作を N 回繰り返した

後 ($n == N$) に，圧縮後のテスト系列において，まだ故障シミュレーションを実行していない基本テンプレート長分のテスト系列を逆変換して時間展開モデルのテストパターンを抽出する．具体的には，まだ故障シミュレーションを実行していないテスト系列の先頭に基本テンプレートの先頭を重ね合わせて，基本テンプレートの b の位置に重ね合わさったテスト系列の値を取り出す．続いて，その抽出したテストパターンを用いて時間展開モデル上で故障シミュレーションを実行し，検出故障を未検出故障集合 F から削除するというものである（逆変換故障シミュレーション）．ただし，逆変換故障シミュレーションを行う前に，圧縮後のテスト系列に存在する X にランダムに 0 か 1 の値を設定しておくものとする．最終的に得られるテスト系列は，テンプレートを用いて圧縮したテスト系列を一つ前のテンプレートを用いて圧縮したテスト系列の有効長の時刻から重ねていく操作を繰り返すことにより生成できるので，以上で述べた「N 回のテストパターン生成 → 逆変換故障シミュレーション」という処理を未検出故障集合 F が空集合になるまで繰り返すことになる．

```
Test_generation_acyclic(S)
{
    Generate a time expansion model corresponding to S.
    Generate a template for a test sequence compaction.
      (compacted N primitive templates)
    n = 0.
    for (undetedcted fault f ∈ F)
    {
        Select a fault f from undetected fault set F.
        Generate a test pattern for f in the time expansion model.
        Perform fault simulation for the time expansion model
        with the test pattern.
        Remove detected faults from F.
        Transform the test pattern into a test sequence with a test
        sequence compaction using the template.
        n++;
        if (n= = N)
        {
            Extract test patterns by a reverse transformation.
            Perform fault simulation for the time expansion
            model with the test patterns.
            Remove detected faults from F.
            n = 0;
        }
    }
}
```

図 8　静的，動的圧縮を含んだ無閉路順序回路のテスト系列生成アルゴリズム
Fig. 8　Test sequence generation algorithm including static and dynamic compaction.

例 5：図 7 のテンプレートを圧縮に用いるテンプレートとする．図 7 のテンプレートは表 5 の 4 個の基本テンプレートを圧縮したものである．またここでは，時間展開モデル上で生成した四つのテストパターン $(PI1(0), PI2(0), PI3(0), PI1(1), PI2(4), PI3(5)) = (0,1,0,0,0,1), (1,1,0,0,1,0), (1,0,1,0,0,0), (0,0,0,0,0,0)$ があるとする．この四つのテストパターンを変換した四つのテスト系列をテンプレートを用いて圧縮すると，図 9 に示すようなテスト系列になる．圧縮に用いたテンプレートの有効長は 15 であるので，次のテンプレートを用いて圧縮したテスト系列と重なる時刻は時刻 15 である．よって，時刻 0〜時刻 14 までのテスト系列中に存在する X にランダムに 0 又は 1 を設定する．図 10 に示す

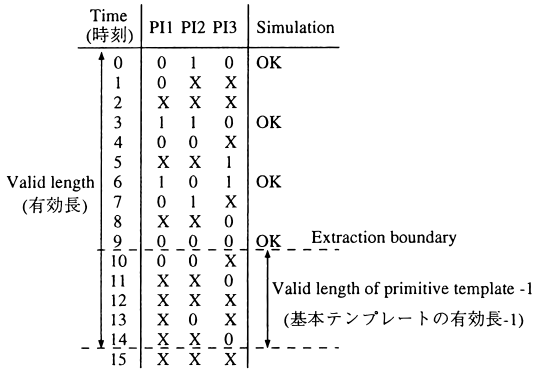

図 9　図 7 のテンプレートを用いた圧縮の例
Fig. 9　Example of compaction using the template of Fig. 7.

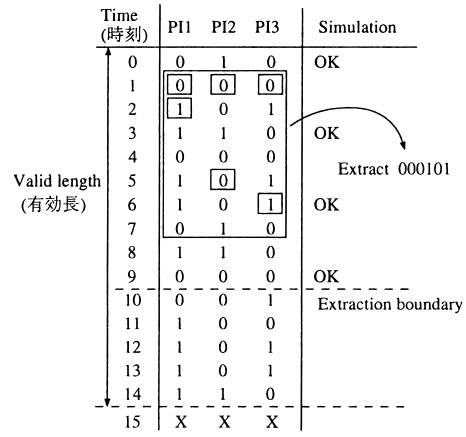

図 10　時間展開モデルのテストパターン抽出の例
Fig. 10　Example of extraction of a test pattern for a time expansion model.

ように，時刻 1〜時刻 7 までのテスト系列を逆変換して，テストパターン $(0,0,0,1,0,1)$ を抽出した．また，図 10 において，値を抽出した部分を四角で囲んだ．同様に，時刻 2〜時刻 8 までのテスト系列を逆変換して，テストパターン $(1,0,1,1,0,0)$，時刻 4〜時刻 10 までのテスト系列を逆変換して，テストパターン $(0,0,0,1,1,0)$，時刻 5〜時刻 11 までのテスト系列を逆変換して，テストパターン $(1,0,1,1,0,1)$，時刻 7〜時刻 13 までのテスト系列を逆変換して，テストパターン $(0,1,0,1,0,1)$，時刻 8〜時刻 14 までのテスト系列を逆変換して，テストパターン $(1,1,0,0,0,1)$ をそれぞれ抽出できる．

5. 実験結果

表 7 に実験に用いる実際の回路（#1〜#4）の特性を示す．表 7 において，#PI は外部入力数，#PO は外部出力数，#FF は FF 数，SR は回路をテスト時に無閉路順序回路にするために必要なスキャン FF の割合（スキャン化率），#GATE は 2 入力 NAND ゲート換算でのゲート数をそれぞれ表す．表 8 に圧縮可能グラフの頂点数を変化させて，提案したヒューリスティックアルゴリズムを用いてテンプレートを生成する実験結果を示す．表 8 において，NV は圧縮可能グラフの頂点数，CPU（単位：秒）は圧縮に費やした CPU 時間すなわち圧縮可能グラフの作成とクリーク抽出に要した CPU 時間（Ultra2 使用，動作速度：296 MHz，SPECint95：12.1，SPECfp95：18.3），VL は生成したテンプレートの有効長，NP は圧縮した基本テンプレート数，TL は全体のテストパターン長，CR は VL/NP で圧縮効率をそれぞれ表す．CR の値が小さいほど，圧縮効率が高くなる．実験結果は#1，#2 のみ示す．#3 と#4 は結果として一定のスキュー値の整数倍で基本テンプレートの圧縮を繰り返したテンプレートが生成される．つまり二つの基本テンプレートをテンプレートの有効長が最短になるように圧縮し，更に圧縮したテンプレートと基本テンプレートを同様に圧縮して新たなテンプレートを生成する方法（貪欲なアルゴリズム）を使用した場合と同一の結果になった．表 8 に示すように圧縮可能グラフの頂点数が多くなるほどテンプレートの生成に時間がかかるが，圧縮可能グラフの頂点数が 200 個程度であれば，ほとんど無視できる時間内で生成できることがわかる．圧縮効率と頂点数についての考察は今後の課題として残る．

表 7　実験回路の特性
Table 7　Properties of experimental circuits.

回路名	#PI	#PO	#FF	SR(%)	#GATE
#1	97	16	176	0	4687
#2	113	16	272	0	4706
#3	342	11	1550	61.9	11682
#4	673	487	3744	72.5	51135

表 8　実験結果：テンプレート生成
Table 8　Experimental results: Template generation.

回路	NV	CPU	VL	NP	TL	CR
#1	8	0.01	12	3	984	4.00
	15	0.02	18	5	864	3.60
	114	1.16	119	37	738	3.22
	171	6.54	176	56	720	3.14
	200	8.35	205	66	708	3.11
#2	11	0.01	16	3	1616	5.33
	21	0.02	29	5	1762	5.80
	153	1.50	162	30	1635	5.40
	200	4.66	209	39	1622	5.36
	304	25.70	314	58	1638	5.41

表 7 にヒューリスティックアルゴリズムの有効性を評価するために，圧縮可能グラフから圧縮効率が最も高くなる最大クリークを抽出して生成したテンプレートとの比較を行った実験結果を示す．

表 9 において HVL，HNP，HCR はヒューリスティックアルゴリズムを用いて生成したテンプレートの有効長，圧縮する基本テンプレート数，圧縮効率（HVL/HNP）をそれぞれ示し，OVL，ONP，OCR は最大クリークから生成したテンプレートの有効長，圧縮する基本テンプレート数，圧縮効率（OVL/ONP）をそれぞれ表す．表 9 の結果から提案したヒューリスティックアルゴリズムはほぼ最高の圧縮効率を達成していることがわかる．また，圧縮可能グラフの頂点数は 200 とした．以後テンプレートを用いた圧縮の実験は頂点数 200 で行う．

表 10 にテンプレートを用いた圧縮方法を評価した実験結果を示す．表 10 において，NC は圧縮を行わない実験結果，TC はテンプレートを用いて圧縮を行った実験結果，3TC はテスト系列の先頭から $X,0,1$ の 3 値の圧縮演算で圧縮可能 [8], [17] である部分を探索し，圧縮可能である部分を探索した時点で圧縮を行った実験結果を示す．CPU（単位：秒）は圧縮時間（Ultra2 使用，動作速度：296 MHz，SPECint95：

表 9　実験結果：ヒューリスティックアルゴリズム
Table 9　Experimental results: Heuristic algorithm.

回路	HVL	HNP	HCR	OVL	ONP	OCR
#1	206	66	3.12	205	67	3.06
#2	209	39	5.35	208	39	5.33

表 10　実験結果：テンプレートを用いた圧縮方法
Table 10　Experimental results: Compaction method using a template.

回路	NC	TC			3TC		
	TL	CPU	TL	TR	CPU	TL	TR
#1	1589	8.85	708	44.56	6.08	910	57.27
#2	3030	4.58	1622	53.53	5.86	1821	60.10
#3	1140	13.17	584	51.23	9.98	572	50.18
#4	13740	8.86	7573	55.12	5137.75	6791	49.43

表 11　逆変換故障シミュレーションを用いた圧縮方法の実験
Table 11　Experimental results: Compaction method using reverse transformation fault simulation.

回路	TC		TCRF			
	CPU	TL	CPU	TL	TR1	TR2
#1	215.88	708	136.92	287	40.53	18.06
#2	138.24	1622	74.57	803	49.51	26.50
#3	44.32	584	51.54	351	60.10	30.79
#4	1150.23	7573	1525.21	4792	63.28	34.88

12.1，SPECfp95：18.3），TL は全体のテスト系列長，TR は NC のテスト系列長を基準とした TC 又は 3TC のテスト系列長の割合を示す．TC は NC に比べて，テスト系列長を 45〜55%に削減することができた．また 3TC と比較すると，テスト系列長は複数の基本テンプレートの圧縮を同時に考えることによって効果のあった#1，#2 の回路に対しては TC のテスト系列長のほうが短く，#3，#4 の回路に対しては，3TC のほうが短くなる．圧縮時間に関しては，#1〜#3 の回路については TC，3TC ともほぼ同程度であるが，大規模回路である#4 については TC は 3TC の約 57 倍の速度でテスト系列の圧縮を行うことができた．表 11 に逆変換故障シミュレーションを用いた圧縮方法を評価する実験結果を示す．表 11 において，TC はテンプレートを用いた圧縮のみを行った実験結果，TCRF はテンプレートを用いた圧縮と逆変換故障シミュレーションを用いた圧縮を組み合わせて行った実験結果，TR1 は TC のテスト系列長を基準とした TCRF のテスト系列長の割合，TR2 は表 10 の NC のテスト系列長を基準とした TCRF のテスト系列長の割合を示す．CPU は ATPG 時間，TL 全体のテスト系列長である．TCRF は TC に比べて全体のテスト系列長を 41〜63%に，NC に比べて全体のテスト系列長を 19〜34%に削減することができ，静的圧縮と動的圧縮を組み合わせて更にテスト系列を圧縮することができた．

6. むすび

本論文では，テンプレートを用いた無閉路順序回路の静的テスト系列圧縮方法と逆変換故障シミュレーションによる無閉路順序回路の動的テスト系列圧縮方法を提案した．実際の回路で提案方法を評価した結果，以下の結論を得ることができた．

（1） 提案したヒューリスティックアルゴリズムは効果的である．

（2） テンプレートを用いた圧縮を用いることで，テスト系列長を 45〜55%に短縮することができた．

（3） 逆変換故障シミュレーションによる圧縮を用いることで，テスト系列長を更に 41〜63%に短縮することができた．

今後の課題として，以下のことが挙げられる．

（1） 圧縮効率と圧縮可能グラフの頂点数の関係を考察する．

（2） 逆変換故障シミュレーションによる圧縮方法をより圧縮効率が高くなるように改善する．

謝辞　本論文に関し，貴重な御意見をいただいた松下電器産業株式会社（株）半導体開発本部の辻史郎氏，太田光保氏，川口謙一氏，並びに，奈良先端科学技術大学院大学の増澤利光助教授，井上美智子助手に感謝致します．

文　献

[1] H. Fujiwara, Logic Testing and Design for Testability, The MIT Press, 1985.

[2] M. Abramovici, M.A. Breuer, and A.D. Friedman, Digital Systems Testing and Testable Design, Computer Science Press, 1990.

[3] K.-T. Cheng and V.D. Agrawal, “A partial scan method for sequential circuits with feedback,” IEEE Trans. Comput., vol.39, no4, pp.544–548, April 1990.

[4] D.H. Lee and S.M. Reddy, “On determining scan flip-flops in partial scan design approach,” Proc. IEEE Proc. Int. Conf. Computer-Aided Design, pp.322–325, Nov. 1990.

[5] S.T. Chakradhar, A. Balakrishnan, and V.D. Agrawal, “An exact algorithm for selecting partial scan flip-flops,” Proc. ACM/IEEE Design Automation Conf., pp.81–86, June 1994.

[6] R. Gupta, R. Gupta, and M.A. Breuer, “The BALLAST methodology for structured partial scan design,” IEEE Trans. Comput., vol.39, no.4, pp.538–544, April 1990

[7] T. Takasaki, T. Inoue, and H. Fujiwara, “Partial scan design methods based on internally balanced structure,” IEEE Proc. Asia and South Pacific Design Automation Conf., pp.211–216, Feb. 1998.

[8] T. Hosokawa, T. Hiraoka, M. Ohta, M. Muraoka, and S.Kuninobu, "A partial scan design method based on n-fold line-up structures," IEEE Proc. Asian Test Symp., pp.306–311, Nov. 1997.
[9] T. Inoue, T. Hosokawa, T. Mihara, and H. Fujiwara, "An optimal time expansion model based on combinational ATPG for RT level circuits," IEEE Proc. Asian Test Symp., pp.190–197, Dec. 1998.
[10] P .Goel and B.C. Rosales, "Test generation and dynamic compaction of tests," IEEE Proc. Int. Test Conf., pp.189–192, Oct. 1979.
[11] G. Tromp, "Minimal test sets for combinational circuits," IEEE Proc. Int. Test Conf., pp.204–209, Oct. 1991.
[12] I. Pomeranz, L.N. Reddy, and S.M. Reddy, "COMPACTEST: A method to generate compact test sets for combinational circuits," IEEE Proc. Int. Test Conf., pp.194–203, Oct. 1991.
[13] S. Kajihara, I. Pomeranz, K. Kinoshita, and S.M. Reddy, "Costeffective generation of minimal test sets for stuck-at faults in combinational logic circuits," IEEE Trans. on Computer-Aided Design, pp.1496–1504, Dec. 1995.
[14] I. Pomeranz and S.M. Reddy, "Dynamic test compaction for synchronous sequential circuits using static compaction techniques," IEEE Proc. Fault-Tolerant Computing Symp., pp.53–61, June 1996.
[15] I. Pomeranz and S.M. Reddy, "On static compaction of test sequences for synchronous sequential circuits," ACM/IEEE Proc. Design Automation Conf., pp.215–220, June 1996.
[16] I. Pomeranz and S.M. Reddy, "Vector restoration based static compaction of test sequences for synchronous sequential circuits," IEEE Proc. Int. Conf. on Computer Design, pp.360–365, Oct. 1997.
[17] T.M. Niermann, R.K. Roy, J.H. Patel, and J.A. Abraham, "Test compaction for sequential circuits," IEEE Trans. on Computer-Aided Design, vol.11, no.2, pp.260–267, Feb. 1992.
[18] M.S. Hsiao, E.M. Rudnick, and J.H. Patel, "Fast algorithm for static compaction of sequential circuit test vectors," IEEE Proc. VLSI Test Symp., pp.188–195, April 1997.
[19] S.T. Chakradhar and A. Raghunathan, "Bottleneck removal algorithm for dynamic compaction in sequential circuits," IEEE Trans. on Computer-Aided Design, vol.16, no.10, pp.1157–1172, Oct. 1997.
[20] Giovanni De Micheli, Synthesis and optimization of digital circuits, McGraw-Hill, Inc., 1994.

（平成 10 年 10 月 23 日受付，11 年 2 月 10 日再受付）

細川　利典

昭 62 明大・工・電子通信卒．同年松下電器産業（株）入社．論理シミュレーションエンジン，テストパターン生成，故障シミュレーション，テスト容易化設計，上流テストの研究開発に従事．情報処理学会，IEEE 各会員．

井上　智生　（正員）

昭 63 明大・工・電子通信卒．平 2 同大大学院博士前期課程了．同年松下電器産業（株）に入社，明治大大学院博士後期課程を経て，現在奈良先端科学技術大学院大学助手．松下電器産業（株）においてマイクロプロセッサの研究開発に従事．明大，奈良先端大において，テスト生成，並列処理，テスト容易化設計の研究に従事．工博．情報処理学会，IEEE 各会員．

平岡　敏洋

平 6 京大・工・精密卒．平 8 同大大学院修士課程了．同年松下電器産業（株）入社．平 10 年 10 月より京都大学大学院情報科学研究科システム科学専攻助手．松下電器産業（株）において，テスト容易化設計，テストパターン生成の研究開発に従事．

藤原　秀雄　（正員）

昭 44 阪大・工・電子卒．昭 49 同大大学院博士課程了．同大・工・電子助手，明治大・工・電子通信助教授，理工・情報科学教授を経て，現在奈良先端科学技術大学院大学教授，昭 56 ウォールターループー大客員助教授．昭 59 マッギル大客員準教授．論理設計論，高信頼性設計，設計自動化，テスト容易化設計，テスト生成，並列処理，計算複雑度に関する研究に従事．著書「Logic Testing and Design for Testability」（MIT Press）など．工博．情報処理学会会員．IEEE Fellow.