ドントケアを用いたキャプチャセーフテスト集合の静的テスト圧縮法

越智小百合[†1]　山崎紘史[†2]　細川利典[†2]　吉村正義[†3]

近年，VLSI の微細化・大規模化に伴いテストパターン数が増加している．テスト品質を低下させずにテストパターン数を削減する手法にテスト圧縮技術がある．しかしながら，テスト圧縮後の各テストパターンの検出故障数が増加することにより，1 テストパターンあたりの遷移信号線数が増加する．それゆえ，スキャンテストにおける過度のキャプチャ消費電力は過度の IR ドロップを引き起こし，歩留り損失が発生する可能性がある．本論文では，圧縮可能なテストパターンの組み合せに対し，圧縮後のキャプチャ時消費電力の見積もり値を計算し，その見積もり値が閾値を越えた場合，そのテスト圧縮の組合せを禁止するキャプチャ時消費電力を考慮した静的テスト圧縮法を提案する．ISCAS’89 ベンチマーク回路のキャプチャセーフテスト集合に対する実験結果は，本提案手法が効果的であることを示す．

A Static Test Compaction Method of Capture Safe Tests Set Using Don't Cares

SAYURI OCHI[†1]　HIROSHI YAMAZAKI[†2]
TOSHINORI HOSOKAWA[†2]　MASAYOSHI YOSHIMURA[†3]

In recent years, the number of test patterns has increased due to VLSI circuit density and complexity. A test compaction technique can reduce the number of test patterns without losing test quality. However, the number of transitioned signal lines per one test pattern increases since each compacted test pattern detects more faults. Therefore, excessive capture power consumption at scan testing cause the excessive IR-drop and it might induce unnecessary yield loss. In this paper, we propose a static test compaction method taking capture power consumption into account. The method calculates the estimated values of the capture power consumptions after the compaction for the combinations of compactable test patterns and prohibits the test compaction for the combinations when the estimated values are more than the threshold value. Experimental results for capture-safe test sets of ISCAS’89 benchmark circuits show that our proposed method was effective.

1. はじめに

近年，半導体微細化技術の発達に伴い，超大規模集積回路（Very Large Scale Integrated Circuits : VLSI）の実速度スキャンテストは必要不可欠な技術となっている．スキャンテストでは，順序回路の全てのフリップフロップ（Flip-Flop:FF）がシフトとキャプチャの 2 つのモードで動作するスキャン FF に置き換えられる[1]．シフトモードでは，シフトインによるテストパターンの印加とシフトアウトによるテスト応答の観測を行い，キャプチャモードでは，組合せ回路部のテスト応答をスキャン FF に取り込む．フルスキャン設計された回路に対するテストでは，順序回路を擬似的に組合せ回路として取り扱うことが可能となり，テスト容易性が向上する[2].

一般的に実速度スキャンテストの消費電力は，テストコスト削減のため少数のテストパターンで多くの故障を検出しようとするため，VLSI が機能動作する際の消費電力と比較して 3 倍～5 倍になることが報告されている[3].

スキャンテスト時消費電力として，スキャンチェインへのテストデータの印加を行うシフト動作時に発生するシフト時消費電力[4]と，テスト応答 FF への格納を行うキャプチャ時に発生するキャプチャ時消費電力[4]が挙げられる．シフト動作時には最大スキャンパス長分のクロックサイクル数が必要となるため，シフト時消費電力の影響は温度上昇という形で現れ，VLSI を熱破壊する可能性がある．一方，キャプチャ動作時には FF の遷移が同じタイミングで発生し，回路内の多くの信号線が遷移するため，キャプチャ時消費電力の影響は IR ドロップ[3]による回路内の遅延増加という形で現れる．増加した遅延は遅延故障として検出される可能性がある．その結果，良品を不良品として判定する誤テストを行う可能性があり，歩留り損失の原因となる．したがって，歩留り損失や回路の熱破壊を抑制するため，テスト時の消費電力を削減することが重要な課題となっている．

キャプチャ時消費電力の削減のための手法は，テストパターンの変更による手法[5][6][7][8]とテスト生成による手法[9][10][11]が提案されている．テストパターン変更による手法は，ドントケア（X）判定による手法[5]，X 割当てによる手法[6][7][8]がある． X 判定による手法[5]は，低消費電力な X 割当てを効果的に行うために，1 つのテストパターンで検出可能な故障数を平均化する手法[5]が提案されている．また，X 割当てによる手法は，正当化や含意操作などの決定的アルゴリズムを用いてテストパターン中の X に適切な 0 や 1 を割当てる LCP-FILL[6]や確率計算を用いてテストパターン中の X に適切な 0 や 1 を割当てる Preferred-FILL[7], その両方の手法を組み合わせた JP-FILL[8]などが挙げられる．これらの手法は，キャプチャ時の FF の遷移数を削減す

†1 日本大学大学院　生産工学研究科
Graduate School of Industrial Technology, Nihon University
†2 日本大学　生産工学部
College of Industrial Technology, Nihon University
†3 京都産業大学　コンピュータ理工学部
College of Faculty of Computer Science and Engineering, Nihon University

ることで，キャプチャ時消費電力を削減することを目的としている．また，テスト生成による手法[9][10][11]は，従来の故障検出重視のテスト生成手法により生成されたテスト集合内の高消費電力テストパターンでのみ検出されるアンセーフ故障[9]に対して，低消費電力テストパターンを模倣し，高速に低消費電力なテストパターンを合成する手法[10][11]などが挙げられる．上記で示したキャプチャ時消費電力削減のための手法を用いることにより，高消費電力テストパターンであるアンセーフパターン数[9]やアンセーフ故障数[9]を削減することは可能だが，テストパターン数の増加が課題となっている．

テストパターン数の増加は，VLSI のテストコストの増加につながるため，テストパターン数を削減することが重要である．テストパターン数を削減するために様々なテスト圧縮法[12][13][14]が提案されている．テストパターン数を削減する手法の 1 つとして，生成されたテスト集合に対し，X 判定，X に基づく静的圧縮，X 割当て，二重検出法[14]を繰返し適用することで，テストパターン数を削減する手法が提案されている[13]．この手法において，セーフテスト集合に対して，X 判定や X 割当てに低消費電力指向な手法を適用したとしても，アンセーフパターンを生成し，アンセーフ故障が存在する可能性がある．

本論文では，キャプチャ時消費電力を考慮した静的テスト圧縮法を提案する．提案手法では，テスト圧縮前に圧縮後のテストパターンの消費電力を見積もり，それを考慮して圧縮することでアンセーフパターンを生成しないで，できる限りテストパターン数を削減することを目指す．

2. 低消費電力テスト

相補型金属酸化膜半導体（Complementary Metal Oxide Semiconductor：CMOS）回路の消費電力は静的なリーク電流，及びスイッチング動作に起因する動的電流により構成される．

2.1 動的消費電力の推定

CMOS 回路の動的消費電力は(1)式のとおりである[15]．

$$P_d = \frac{1}{2} \times V_{DD}^2 \times f_p \times \sum_{i=1}^{G} E(t_i) \times C_i \quad \cdots(1)$$

（1）式において，V_{DD}は電源電圧，f_pはクロック周波数，Gは回路中の総ゲート数，$E(t_i)$はあるテストパターンt_i当たりの予想スイッチング数，C_iはゲートg_iの負荷容量を表す．本論文では，スイッチング動作に起因するキャプチャ時消費電力を評価する．キャプチャ時消費電力を見積もる手法として，多くの評価尺度では，FF や内部信号線における論理値の遷移数を評価している．そのため，消費電力の評価として計算式を簡略化した WSA[15]を用いる．あるテストパターンt_iの WSA は（2）で表される．

$$\mathrm{WSA}(t_i) = \sum_{j=1}^{G} tran(g_j) \times \left(1 + fanout(g_j)\right) \cdots(2)$$

（2）式において，$\mathrm{WSA}(t_i)$はテストパターンt_iの WSA 値を表す．G は回路中の総ゲート数を表す．$tran(g_j)$はゲートg_jの遷移関数であり，g_jに遷移が発生した場合は 1，それ以外は 0 を返す．$fanout(g_j)$はゲートg_jのファンアウト数を示す．

2.2 アンセーフパターンとアンセーフ故障

2.1 節で説明したように，多数のスイッチング動作は過度な消費電力を発生させる．キャプチャ時の過度な消費電力は過度な IR ドロップを発生させる．過度な IR ドロップは遅延を増加させ，増加した遅延が遅延故障として検知され可能性がある．その結果，良品を不良品として判定する誤テストを行う可能性がある．そのため，閾値を超えた高キャプチャ時消費電力なテストパターンは，テストに使用することができない．このようなテストパターンをアンセーフパターン[9]と呼ぶ．一方，テストに使用できるテストパターンのことをセーフパターン[9]と呼ぶ．その集合をセーフテスト集合と呼ぶ．また，アンセーフパターンはテストに使用できないため，アンセーフパターンでのみ検出可能なアンセーフ故障[9]は，テストされなくなる．

3. テストパターン中のドントケア判定手法に基づく静的圧縮法

本章では，従来手法[13]の説明をする．従来手法では，テスト集合に対して X 判定を適用したあと，静的テスト圧縮を適用し圧縮されたテスト集合を求める．また，圧縮されたテスト集合に対して，X 割当てと二重検出法[14]を適用し，さらにテストパターン数を小さくする．

図 1 に従来手法であるテストパターン中のドントケア判定手法による静的テスト圧縮法のアルゴリズムを示す．入力として，テスト生成で生成した 2 値のテスト集合Tと回路Cを与える．まず，入力として与えられたTに対して，二重検出法を適用し，極小なテスト集合を生成する（行 5）．行 5 で生成したT_{mini}に対して，行 7 から行 15 の処理を適用する．行 5 で生成したT_{mini}に対して，X 判定を適用し，X を含むテスト集合T_{xid}を生成する（行 7）．行 7 を適用したT_{xid}をもとに頂点をテストパターン，頂点間の辺を圧縮不可能性とした圧縮不可能グラフを生成する（行 8）．行 7 で生成したT_{xid}と行 8 で生成したGに対して，頂点彩色問題[13]を解き，ドントケアに基づくテスト圧縮を適用し，圧縮されたテスト集合T_{comp}を生成する（行 9）．行 9 で生成したT_{comp}中にあるドントケアに対して 0 や 1 を割当て，T_{f_comp}を生成する（行 10）．行 10 で生成したT_{f_comp}に対して，再度，二重検出法を適用し，極小テスト集合$T_{minimal}$を生成する（行 11）．もし，行 9 で生成したT_{comp}中のテストパターン数がT_{xid}中のテストパターン数と比較して削減されていなければ，行 11 で生成した$T_{minimal}$を返し，処理を終了する（行 13）．そうでなければ，行 7 から行 11 までの処理を再度繰り返し適用する．

```
1. input C: Circuit, T: Test Set
2. output T_minimal : MinimalTest Set
3. Static Test Compaction Based on Don't Care Identification (C, T)
4. {
5.     T_mini = double_detection_fault_simulation(C, T);
6.     while(1){
7.         T_xid = x_identification(C, T_mini);
8.         G = uncompactability_graph_generation(T_xid );
9.         T_comp = graph_coloring_compaction (G, T_xid );
10.        T_f_comp = x_Filling(T_comp, C);
11.        T_minimal = double_detection_fault_simulation(C, T_f_comp);
12.        if(|T_comp| == |T_xid|){
13.            return( T_minimal);
14.        }
15.    }
16.}
```

図 1. テスト集合中のドントケア判定に基づく静的テスト圧縮法のアルゴリズ

4. ドントケアを用いたキャプチャセーフテスト集合の静的テスト圧縮法

本章では，提案手法であるドントケアを用いたキャプチャセーフテスト集合の静的テスト圧縮法について説明する．提案手法では，キャプチャセーフになるテストパターンのみで構成され，できるだけサイズが小さいテスト集合を求める．そのために，まず，圧縮後もキャプチャセーフになるテストパターンの組合せを列挙する．次に，列挙された圧縮後のテスト集合と圧縮されてないテスト集合の和集合を得る．最後に，和集合から故障検出率を維持するために必要な極小のセーフテスト集合を最小被覆問題とみなして解く．

4.1 問題定式化

“ドントケアを用いたキャプチャセーフテスト集合の静的テスト圧縮法”による問題定式化を示す．

（問題定式化）

入力：フルスキャン設計回路，セーフテスト集合

出力：圧縮済みセーフテスト集合

制約：圧縮済みセーフテスト集合中のテストパターンの WSA 値が WSA 閾値以下

最適化：圧縮後のセーフパターン数

4.2 提案手法の概要

図 2 に，テスト集合の変化例を示す．提案手法では，文献[10]で生成した 2 値のキャプチャセーフテスト集合（図 2(a)）

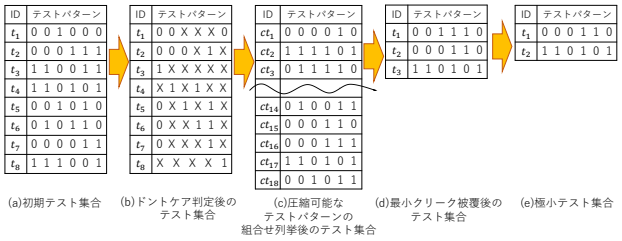

図 2. テスト集合の変化例

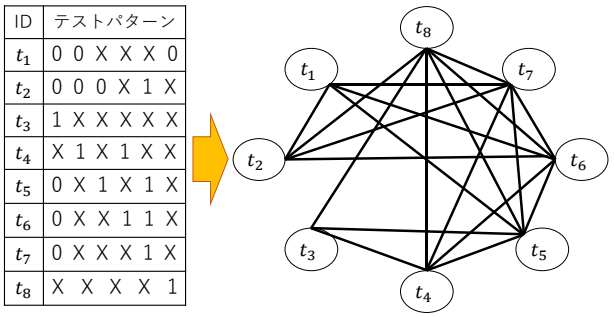

図 3. 圧縮可能グラフの例

に対して，低消費電力を考慮したドントケア判定を適用する（図 2(b)）．次に，ドントケアを含んだ 3 値のテスト集合に対して，圧縮可能なテストパターンの組合せを列挙する（図 2(c)）．次に，図 2（c）で列挙したテスト圧縮の組合せに対して最小被覆問題を解き， 2 値の圧縮したテスト集合を生成する（図 2(d)）．最後に，二重検出法を適用することで極小テスト集合を生成する（図 2(e)）．

4.3 圧縮可能グラフ

図 3 に，圧縮可能グラフの例を示す．提案手法では，圧縮可能グラフを作成する．圧縮可能グラフは，各頂点をドントケア判定後の 3 値のテストパターンt_iで表現する．また，各頂点間の辺は，その隣接頂点がテスト圧縮可能であることを表現する．図 3（a）のテストパターンt_2とt_3は，左から 1 ビット目で値の衝突が発生する．そのため，図 3（b）のように頂点t_2とt_3の間に辺は存在しない．一方，図 3（a）のテストパターンt_1とt_2は，全ビットにおいて値の衝突が発生しないため，圧縮可能とみなされ，図 3（b）のようにt_1とt_2の間に辺を引く．

4.4 アルゴリズム

提案手法では，キャプチャセーフテスト集合に対して，低消費電力を考慮したドントケアに基づくテスト圧縮を行うことで，アンセーフパターンを生成しないテスト集合の生

```
1. input C: Circuit, T_safe : Capture Safe Test Set, th : wsa_threshold
2. output T_min_comp_safe : Minimum Compacted Capture Safe Test Set
3. Procedure_low_power_test_compaction (C, T_safe )
4. {
5.     T_xid = x_identification(C, T_safe);
6.     G_us = compactability_graph_generation(T_xid );
7.     T_comp_safe = compaction_clique_enumeration(G_us , T_xid , th, C);
8.     T_sm_comp_safe = minimum_clique_cover(T_comp_safe, T_safe);
9.     T_min_comp_safe = double_detection_fault_simulation(C, T_m_comp_safe);
10.    return(T_min_comp_safe);
11.}
```

図 4. 提案手法のアルゴリズム

```
1. input C: Circuit, T_xid:X Identification Test Set
       th: WSA_threshold, G_us : Compactability Graph
2. output T_comp_safe: Compacted Capture Safe Test Set
3. compaction_clique_enumeration(T_xid, th, G_us, C)
4. {
5.     T_two_unsafe = ∅;
6.     T_two_safe = ∅;
7.     T_n_safe = ∅;
8.     T_two = two_size_compaction_enumeration(G_us, T_xid);
9.     T_two_fill = low_power_x_filling(C, T_two);
10.    for each test pattern t_twoi in T_two_fill {
11.        wsa_i = calc_wsa(C, t_twoi);
12.        if(wsa_i > th){
13.            T_two_unsafe = T_two_unsafe ∪ t_two_i;
14.        }else{
15.            T_two_safe = T_two_safe ∪ t_two_i;
16.        }
17.    }
18.    G_s = addition_edge(T_two_unsafe, G_us);
19.    T_n = n_size_compaction_enumeration(G_s, T_xid);
20.    T_n_fill = low_power_x_filling(C, T_n);
21.    for each test pattern t_ni in T_n_fill {
22.        wsa_i = calc_wsa(C, t_ni);
23.        if(wsa_i < th){
24.            T_n_safe = T_n_safe ∪ t_n_i;
25.        }
26.    }
27.    T_comp_safe = T_two_safe ∪ T_n_safe;
28.    return(T_comp_safe);
29.}
```

図 5. 圧縮可能なテストパターンの
組合せ列挙アルゴリズム

成を実現する．図 4 は，提案手法の全体アルゴリズムである．提案手法では，入力として文献[10]で生成した 2 値のキャプチャセーフテスト集合T_{safe}と回路 C と WSA 閾値th を与える．はじめに，T_{safe}に対して X 判定を適用し，2 値のテスト集合から 3 値のテスト集合T_{xid}を生成する（行 5）．次に，T_{xid}に対して，圧縮可能グラフG_{us}を作成する（行 6）．次に，入力である C とth，行 5 で生成したT_{xid}，行 6 で生成したG_{us}を用いて，圧縮後のテストパターンがセーフパターンであることを制約として，圧縮可能なテストパターンの組合せを列挙した集合T_{comp_safe}を生成する（行 7）．入力であるT_{safe}と行 7 で生成したT_{comp_safe}の和集合に対して，テストパターンを頂点とし，頂点間の圧縮可能性を辺とする圧縮可能グラフG_sを作成する．このG_sに対して，最小クリーク被覆問題[16]を解く．得られたクリーク集合に対して，クリークごとにクリークに含まれる頂点のテストパターンを圧縮し，テスト集合$T_{sm_comp_safe}$を得る．$T_{sm_comp_safe}$は，セーフパターンのみで構成され、全ての故障を検出するテスト集合である（行 8）．最後に，二重検出法適用し，行 8 で生成したテスト集合より極小セーフテスト集合$T_{min_comp_safe}$を生成する（行 9）．テスト集合$T_{min_comp_safe}$を返す（行 10）．

図 5 に圧縮可能なテストパターンの組合せ列挙アルゴリズムを示す．また，図 6 に圧縮可能なテストパターンの組合せ列挙の例を示す．入力として，回路Cとドントケア判定によって生成された 3 値のテスト集合T_{xid} と WSA 閾値th，T_{xid}をもとに生成した図 6（a）のような圧縮可能グラフG_{us}を与える．まず，*2* サイズアンセーフテスト圧縮集合T_{two_unsafe}と *2* サイズセーフテスト集合T_{two_safe}と n サイズセーフテスト集合T_{n_safe}を∅に初期化する（行 5~行 7）．入力であるG_{us}とT_{xid}をもとに，*2* サイズテスト圧縮を列挙したテスト集合T_{two}を生成する（行 8）．また，本論文では，2 個のテストパターンを圧縮することを *2* サイズテスト圧縮と呼ぶ．行 5 で生成したT_{two}の全てのテストパターンに対して，キャプチャ時低消費電力指 X 割当て[7]を行い，2 値のテスト集合T_{two_fill}を生成する（行 9）．T_{two_fill}に含まれる各テストパターンt_{twoi}に対して行 10 から行 17 の処理を適用する．t_{twoi}に対して，WSA を算出しwsa_iを求める（行 11）．wsa_iがthより大きいか否かを判定する（行 12）．もし，wsa_iがthより大きければt_{twoi}をT_{two_unsafe}に加える（行 13）．それ以外の場合は，t_{twoi}をT_{two_safe}に加える（行 14）．行 10 から行 17 の処理の例が図 6（b）になる．図 6（b）は，*2* サイズテスト圧縮を行った後にキャプチャ時低消費電力指向 X 割当て[7]を行い，WSA 計算した状態である．例では，WSA 閾値を 150 に設定したため，図 6（b）のct_3，ct_{14}，ct_{18}のテストパターンがアンセーフパターンとなった．次に，行 13 で生成したテスト集合をもとに入力であるG_{us}を更新し，G_sを生成する（行 18）．図 6（b）でアンセーフパターンをもとに，圧縮不可能グラフを更新したものが図 6（c）となる．図 6（c）において，実線が圧縮可能を示す線，点線が圧縮により消費電力が閾値を超えるため削除した辺を示す線となる．次に，行 18 で更新したG_sと入力であるT_{xid}を用いて n サイズテスト圧縮を列挙したテスト集合T_nを生成する（行

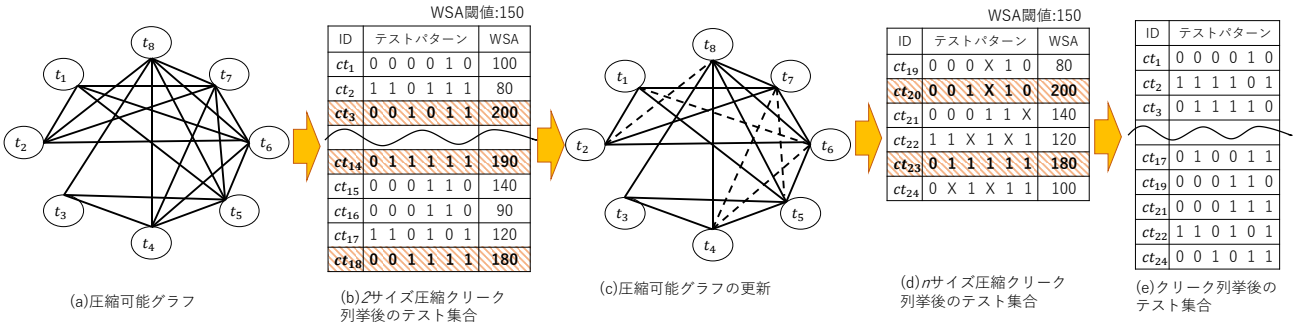

図 6. 圧縮可能なテストパターンの組合せ列挙の例

```
1. input T_comp_safe : Compacted Capture Safe Test Set, T_safe : Capture Safe Test Set
       T_xid : X Identification Test Set
2. output T_sm_comp_safe : Small Compacted Capture Safe Test Set
3. Minimum_Clique_cover(T_comp_safe, T_safe, T_xid)
4. {
5.     M = matrix_generation(T_comp_safe, T_safe, T_xid);
6.     while(M){
7.         if(L_ess = essential_row(M)){
8.             M= essential_row_deletion(M);
9.             M = singular_column(M);
10.        }else{
11.            L_max = max_compaction_row(M);
12.            M= max_compaction_row _deletion(M);
13.            M = singular_column(M);
14.        }
15.        M = dominated_row_deletion(M);
16.        M = dominating_column_deletion(M);
17.    }
18.    T_sm_comp_safe = L_ess ∪ L_max;
19.    return(T_sm_comp_safe);
20.}
```

図 7. クリーク被覆アルゴリズム

19)．また，本論文では，3 個以上のテストパターンを圧縮することを n サイズテスト圧縮と呼び，n の数は，テスト圧縮をするテストパターンの個数で決まる．行 19 で生成した T_nに対して，低キャプチャ時消費電力指向ドントケア割当てを行い 2 値のテスト集合T_{n_fill}を生成する（行 20）．T_{n_fill}に含まれる各テストパターンt_{ni}に対して行 21 から行 26 の処理を適用する．t_{ni}に対して，WSA を算出しwsa_iを求める（行 22）．wsa_iがthより小さいか否かを判定する（行 23）．もしwsa_iがthより小さければt_{ni}をT_{n_safe}に加える（行 24）．行 16 から行 23 までの処理の例が図 6 (d) となる．図 6 (d) は，n サイズテスト圧縮を行った後に低消費電力指向 X 割当てを行い，WSA 計算した状態である．例では，WSA 閾値を 150 に設定したため，図 6 (d) のct_{20}，ct_{23}のテストパターンがアンセーフパターンとなった．最後に，*2* サイズテスト圧縮と n サイズテスト圧縮の和集合であるテスト集合T_{comp_safe}に行 15 で生成したT_{two_safe}と行 24 で生成したT_{n_safe}を加え図 6(e)のようにT_{comp_safe}を更新する(行 27)．T_{comp_safe}を返す（行 28)．

図 7 にクリーク被覆アルゴリズムを示す．図 7 のアルゴリズムは，クワインマクラスキー法[16]に基づいている．図 5 のアルゴリズムの出力である *2* サイズテスト圧縮したテスト集合と n サイズテスト圧縮したテスト集合の和集合であるテスト集合T_{comp_safe}と文献[10]で生成した 2 値のキャプチャセーフテスト集合T_{safe}を入力として与える．まず，入力のテスト集合を用いてクリーク被覆テーブルMを作成する（行 5)．作成されるクリーク被覆テーブルの例を図 8 に示す．各行は，*2* サイズテスト圧縮したテスト集合と n サイズテスト圧縮したテスト集合の和集合であるテスト集合T_{comp_safe}と初期テスト集合T_{safe}に含まれるテストパターンを表している．そのため，総行数は，*2* サイズテスト圧縮と n サイズテスト圧縮の組合せを含んだテスト集合のテストパターン数と初期テストパターン数の和になる．また，各列は，初期テスト集合に対して X 判定を適用したテスト集合T_{xid}に含まれるテストパターンを表しているため，列の総数は，初期テストパターン数になる．さらに，テーブル内に描かれている丸印は，各行のテストパターンct_iに列のtx_jが包含されていることを示す．例えば，1 行目のct_1は，tx_1とtx_2の圧縮結果であるため，1 列目，2 列目のtx_1とtx_2に丸印が付く．次に，行 5 で生成したクリーク被覆テーブルMの行が存在する限り，行 7 から行 16 の処理を適用する．行 5 で生成したMに対して，必須行[16]が含まれているか判定し，含まれている場合，必須行を採用し，必須クリーク集合L_{ess}を更新する（行 7)．必須行は，ある列の要素（図 8 の場合，丸印）が一つしかない場合，その列を被覆するためには要素が 1 つしかない列を採用しなければならない．そのような行を必須行と呼ぶ. その後，採用した必須行を削除し，M を更新する（行 8)．また，必須行の採用により，元の列と同様に必須行によって被覆されているほかの列も削除し，M を更新する（行 9)．行 5 で生成したMに対して，必須行が含まれているか判定し，必須行がない場合，行 11 と行 12 の処理を適用する．まず，圧縮サイズが最大な行を採用し，最大圧縮サイズクリーク集合L_{max}を更新する（行 11)．図 8 の場合，圧縮サイズが 1 番最大ものはct_{21}，ct_{22}，ct_{23}なため，この例ではct_{21}を採用する．その後，行 11 で採用した行を削除し，M を更新する（行 12)．また圧縮サイズが最大な行の採用により，その行に被覆されている列を削除し，M を更新する（行 13)．次に，被支配行[16]の削除をおこない，M を更新する（行 15)．図 8 の場合，二つの行i_1とi_2に対して，i_1上で丸印がついているすべての列においてi_2上の要素も丸印がついている場合，行i_1は行i_2を支配している．また，行i_2を被支配行という。行i_1を被覆するためにいかなる列を採用しても同時に行i_2 が被覆されることは明らかなので，行i_2は削除できる．さらに，支配列[16]の削除をおこない，M を更新する（行 16)．図 8 の場合，二つの列j_1 とj_2に対してj_2 上で丸印がついているすべての行においてj_1上の要素も丸印がついている場合，列j_1は列j_2を支配している．また，列j_1を支配列という．列j_1を採用するときに被覆されるすべての行は列j_2を採用しても被覆されるので，列j_1は削除できる．最後に，行 7 で生成したL_{ess}とL_{max}を$T_{sm_comp_safe}$に加え更新する（行 18)．$T_{sm_comp_safe}$を返す（行 19)．

	tx_1	tx_2	tx_3	tx_4	tx_5	tx_6	tx_7	tx_8
ct_1	○	○						
ct_2	○				○			
ct_3	○					○		
ct_{21}	○	○					○	
ct_{22}			○	○				○
ct_{24}					○		○	○
t_1	○							
t_2		○						
t_6						○		
t_7							○	
t_8								○

図 8. クリーク被覆テーブルの例

表 1. 初期テスト集合情報

circuit	#initial_tps	fault coverage(%)	#detected faults	WSA_th
s5378	228	61.29	4315	1112
s15850	500	65.03	12386	2253
s35932	105	69.41	44076	8741
s38584	1857	71.56	43834	4500

5. 実験結果

提案手法を C 言語で実装し，ISCAS'89 ベンチマーク回路の一部（キャプチャセーフテスト集合のみで遷移故障検出効率 100%を達成できた回路）を対象として実験を行い，テストパターン数，アンセーフパターン数，アンセーフ故障数を評価した．初期テスト集合には，文献[10]の手法を適用して生成したキャプチャセーフテスト集合を用いる．表 1 に初期テスト集合情報を示す．「circuit」は，実験対象回路を示す．「#initial_tps」は，初期テスト集合であるキャプチャセーフテスト集合に含まれているテストパターン数を示す．「fault coverage(%)」は，初期テスト集合に対する故障検出率示し，その時の検出故障数を「#detected faults」で示す．また，「WSA_th」は，WSA の閾値を示す．また，初期テスト集合の最大 WSA の 70%を閾値としている．

表 2 に実験結果を示す．「circuit」は実験対象回路を示し，「initial_TP」は初期テスト集合の情報を示し，「previous」は初期テスト集合に対し，従来手法を施した結果を示し，「proposed」は初期集合に対し，提案手法を施した結果を示す．そのうち，「#initial_tps」は初期テスト集合のテストパターン数を示し，「#tps_after Vertex Coloring」は頂点彩色問題[13]を解いた後のテストパターン数を示し，「#tps_after prposed」は提案手法適用後のテストパターン数を示し，「#tps_after DD」は二重検出法適用後のテストパターン数数を示し，「#unsafe_tps」は各回路の WSA 閾値を超えたテストパターン数を示し，「#unsafe_faluts」は，アンセーフパターン数でしか検出できない故障数を示し，「reduction rate(%)」は提案手法適用後の集合と初期テスト集合のテストパターン数を比較したときの削減率を示す．また，「NA」は，現在実験中であることを示す．

本論文の目的である圧縮後のテストパターン数は，初期テスト集合と比較して最大 40.8%，平均 25.6%削減することができた．

6. むすび

本論文では，キャプチャ時消費電力を考慮したドントケアを用いたキャプチャセーフテスト集合の静的テスト圧縮法を提案した．評価実験では，実験結果が出ている回路において，初期テスト集合と比較して最大 40.8%，平均 25.6%削減することができた．今後の課題として，アルゴリズムの高速化とさらなるテストパターン数の削減することが挙げられる．

参考文献

1) H. Fujiwara, "Logic Testing and Design for Testbility，" The MIT Press, pp298, 1985.
2) J.Savir and S.Patil, "Scan-based transition test," IEEE Trans. Comput. Aided Design Int. Circuit & Syst., vol.13, no.8, pp.1057-1064, 1994.
3) J.Saxena, K.M.Butler, V.B.Jayaram, S.Kundu, N.V.Arvind, P. Sreeprakash and M. Hachinger, "A case study of IR-drop in structured at-speed testing, "Proc.ITC, pp.1098-1104, 2003.
4) A.Krstic, and K.T.Cheng, "Delay Falut Testing for VLSI Circuits," Kluwer Academic Publishers, 1998.
5) K.Miyase, K.Noda, H.Ito, K.Hatayama, T.Aikyo, Y.Yamato, H.Furukawa, X.Wen, and S.Kajihara, "Effective IR-Drop Reduction in At-Speed Scan Testing Using Distribution-Controlling X-Identification," IEEE/ACM International Conference on Computer-Aided Design, pp.52-58, 2008.
6) X.Wen, Y.yamashita, S.kajihara, L-T.Wang, K. K.SALUJA, K.Kinoshita " A New Method for Low-Capture-Power Test Generation for Scan Testing", IEICE Trans. Inf. & Syst.., vol.E89-D, No.5, 2006, pp1679-1686.
7) S.Remersaro, X.Lin, Z.Zhang, S.M.Reddy, I.Pomeranz and J.Rajski, "Preferred Fill: A Scalable Method to Reduce Capture Power for Scan Based Designs," Proc. ITC, paper 32.2, 2006.
8) X.Wen, K.Miyase, S.Kajihara, T.Suzuki, Y.Yamato, P.Girard, Y.Ohsumi and L.-T.Wang, "A Novel Scheme to Reduce Power Supply Noise for High-Quality At-Speed Scan Testing," Proc. ITC, paper 25.1, 2007.
9) X.Wen, K.Miyase, S.Kajihara, H.Furukawa, Y.Yamato, A.Takashima, K.Noda, H.Ito, K.Hatayama, T.Aikyo and K.K.Saluja, "A Capture-Safe Test Generation Scheme for At-Speed Scan Testing," Proc. ETS, pp. 55-60, 2008.
10) T.Hosokawa, A.Hrai, Y.Yamauchi, M.Arai, "A Low Capture Power Test Generation Method Based on Capture Safe Test Vector Manipulation," IEICE Trans. Inf. & Syst., vol.E100-D, No.9, 2017(to be appeared.)
11) 細川利典, 平井敦士, 山崎紘史, 新井雅之, "キャプチャセーフベクトルを利用した低消費電力指向テスト生成における動的テスト圧縮法", 信学技法, vol.116, no.466, pp.1-6, 2017.
12) P.Goel, and B.C.Rosales, "Test Generation and Dynamic Compaction of Tests, "Digest of Papers 1979 Test Conf., pp.189-192, Oct.1995
13) K.Miyase, S.Kajihara, S.M.Reddy Electronic Design, "A Method Of Static Test Compaction Based on Don't Care Identification," Test and Applications, IPSJ Journal vol.43 No.5, pp .1290-1293, May, 2002.
14) S.Kajihara, I.Pomeranz, K.Kinoshita and S.M.Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," IEEE Trans. on CAD, pp.1496-1504, Dec.1995.
15) S-J.Wang, K-L.Fu, K.S-M.Li, "Low Peak Power ATPG for n-Detection Test", IEEE Trans., pp.1993-1996, 2009.
16) 笹尾勤, 論理設計-スイッチング回路理論, 近氏科学社, 1995

表 2. 実験結果

	initial_TP	previous				proposed				
circuit	#initial_tps	#tps_after Vertex Coloring	#tps_after DD	#unsafe_tps	#unsafe_faluts	#tps_after prposed	#tps_after DD	#unsafe_tps	#unsafe_faluts	reduction rate(%)
s5378	228	204	201	0	0	204	202	0	0	11.4
s15850	500	282	281	14	138	296	296	0	0	40.8
s35932	105	77	77	3	339	79	79	0	0	24.7
s38584	1857	527	527	46	934	NA	NA	NA	NA	NA