(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5066684号
(P5066684)
(45)発行日　平成24年11月7日(2012.11.7)　(24)登録日　平成24年8月24日(2012.8.24)

(51)Int.Cl.　F I
G01R 31/3183 (2006.01)　G01R 31/28　Q
G01R 31/28 (2006.01)　G01R 31/28　G

請求項の数 19　(全 23 頁)

(21)出願番号　特願2006-88695(P2006-88695)
(22)出願日　平成18年3月28日(2006.3.28)
(65)公開番号　特開2007-263724(P2007-263724A)
(43)公開日　平成19年10月11日(2007.10.11)
審査請求日　平成20年12月11日(2008.12.11)

(73)特許権者　504174135
国立大学法人九州工業大学
福岡県北九州市戸畑区仙水町１番１号
(74)代理人　100116573
弁理士　羽立　幸司
(72)発明者　温　暁青
福岡県飯塚市大字川津６８０－４　九州工業大学内
(72)発明者　梶原　誠司
福岡県飯塚市大字川津６８０－４　九州工業大学内
(72)発明者　宮瀬　紘平
福岡県福岡市早良区昭代１－６－２７　４０２号

最終頁に続く

(54)【発明の名称】生成装置、生成方法、生成方法をコンピュータに実行させることが可能なプログラム、及び、このプログラムを記録した記録媒体

(57)【特許請求の範囲】
【請求項１】
論理回路に対する初期テストベクトル集合を変換して新しいテストベクトル集合を生成する生成装置であって、
前記論理回路はフルスキャン設計された順序回路であって、
前記初期テストベクトル集合の各テストベクトルのうち、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数について所定の基準を超えている対象テストベクトルを特定する特定手段と、
前記特定手段が特定した対象テストベクトルに代えて、前記対象テストベクトルにおける前記論理値に相違が発生するビットの論理値を少なくとも１つ変更して、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる新しいテストベクトルを生成する生成手段とを備えた、生成装置。
【請求項２】
前記生成手段は、
前記特定手段が特定した対象テストベクトルに対し、対象故障を選択する選択手段を有し、
前記特定手段が特定した対象テストベクトルにおける前記論理値に相違が発生するビットの論理値を少なくとも１つ変更することにより、前記選択手段が選択した対象故障を全て検出でき、且つ、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプ

チャの前後において論理値に相違が発生するビットの数を減少させる対象テストベクトルに代わる新しいテストベクトルを生成する、請求項１記載の生成装置。

【請求項３】

前記選択手段は、前記特定手段が特定した対象テストベクトルのみで検出される各故障を、一つの対象テストベクトルのみで検出されるベクトル必須故障と複数の対象テストベクトルで検出されるセット必須故障に区別し、前記ベクトル必須故障についてそれを検出する対象テストベクトルの対象故障として選択するとともに前記セット必須故障についてそれを検出する複数の対象テストベクトルのうちのいずれかの対象故障として選択する、請求項２記載の生成装置。

【請求項４】

前記生成手段は、前記順序回路に対応する組合せ回路に対し、テスト生成過程において前記組合せ回路のある擬似外部入力の値と前記擬似外部入力に対応する擬似外部出力の値とが異なることを表すキャプチャ衝突を定義し、前記キャプチャ衝突が生じたときにバックトラックを行って、前記特定手段が特定した対象テストベクトルに代わる新しいテストベクトルを生成する、請求項１から３のいずれかに記載の生成装置。

【請求項５】

論理回路に対する初期テストベクトル集合を変換して新しいテストベクトル集合を生成する生成装置であって、

前記論理回路はフルスキャン設計された順序回路であって、

前記順序回路に対応する組合せ回路に対し、テスト生成過程において前記組合せ回路のある擬似外部入力の値と前記擬似外部入力に対応する擬似外部出力の値とが異なることを表すキャプチャ衝突を定義し、前記キャプチャ衝突が生じたときにバックトラックを行って、前記初期テストベクトル集合のうちの対象テストベクトルに代えて、前記対象テストベクトルにおける前記キャプチャ衝突を生じるビットの論理値を少なくとも１つ変更することにより、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる新しいテストベクトルを生成する生成手段を備えた、生成装置。

【請求項６】

前記生成手段は、テスト生成過程における主含意スタックとは別に、前記キャプチャ衝突が生じたときに更新されて前記主含意スタックとしての回復に用いられうるスタックである回復含意スタックを用いる、請求項４又は５記載の生成装置。

【請求項７】

論理回路に対する初期テストベクトル集合を変換して新しいテストベクトル集合を生成する生成方法であって、

前記論理回路はフルスキャン設計された順序回路であって、

特定手段が、前記初期テストベクトル集合の各テストベクトルのうち、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数について所定の基準を超えている対象テストベクトルを特定する特定ステップと、

生成手段が、前記特定手段が特定した対象テストベクトルに代えて、前記対象テストベクトルにおける前記論理値に相違が発生するビットの論理値を少なくとも１つ変更することにより、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる新しいテストベクトルを生成する生成ステップとを含む、生成方法。

【請求項８】

前記生成ステップは、

前記特定された対象テストベクトルに対し、対象故障を選択する第１のステップと、

前記特定手段が特定した対象テストベクトルに代えて、前記対象テストベクトル前記論理値に相違が発生するビットの論理値を少なくとも１つ変更することにより、前記選択された対象故障を全て検出でき、且つ、前記順序回路に含まれるスキャンセルの出力につ

いてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させるように対象テストベクトルに代わる新しいテストベクトルを生成する第２のステップとを含む、請求項７記載の生成方法。

【請求項９】

前記第１のステップは、前記特定された対象テストベクトルのみで検出される各故障を、一つの対象テストベクトルのみで検出されるベクトル必須故障と複数の対象テストベクトルで検出されるセット必須故障に区別し、前記ベクトル必須故障についてそれを検出する対象テストベクトルの対象故障として選択するとともに前記セット必須故障についてそれを検出する複数の対象テストベクトルのうちのいずれかの対象故障として選択する、請求項８記載の生成方法。

【請求項１０】

前記セット必須故障がいずれの対象テストベクトルについての対象故障として選択されるかについての判断に、複数の前記対象故障から到達可能な擬似外部入力の集合の重なりの度合い及び複数の前記対象故障から到達可能な擬似外部出力の集合の重なりの度合いを示す演算式に基づく結果が用いられる、請求項９記載の生成方法。

【請求項１１】

前記生成ステップでは、前記順序回路に対応する組合せ回路に対し、テスト生成過程において前記組合せ回路のある擬似外部入力の値と前記擬似外部入力に対応する擬似外部出力の値とが異なることを表すキャプチャ衝突を定義し、前記キャプチャ衝突が生じたときにバックトラックを行って、前記特定手段が特定した対象テストベクトルに代わる新しいテストベクトルを生成する、請求項７から１０のいずれかに記載の生成方法。

【請求項１２】

論理回路に対する初期テストベクトル集合を変換して新しいテストベクトル集合を生成する生成方法であって、

前記論理回路はフルスキャン設計された順序回路であって、

生成手段が、前記順序回路に対応する組合せ回路に対し、テスト生成過程において前記組合せ回路のある擬似外部入力の値と前記擬似外部入力に応答する擬似外部出力の値とが異なることを表すキャプチャ衝突を定義し、前記キャプチャ衝突が生じたときにバックトラックを行って、前記初期テストベクトル集合のうちの対象テストベクトルに代えて、前記対象テストベクトルにおける前記キャプチャ衝突を生じるビットの論理値を少なくとも１つ変更することにより、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる新しいテストベクトルを生成する生成ステップを含む、生成方法。

【請求項１３】

前記生成ステップでは、テスト生成過程における主含意スタックとは別に、前記キャプチャ衝突が生じたときに更新されて前記主含意スタックとしての回復に用いられうるスタックである回復含意スタックのリストである回復含意スタックリストを用いる、請求項１１又は１２記載の生成方法。

【請求項１４】

前記生成ステップでは、前記キャプチャ衝突が生じれば、現在の主含意スタックの複製が前記キャプチャに対応する回復含意スタックとして前記回復含意スタックリストの先頭に追加される、請求項１３記載の生成方法。

【請求項１５】

前記生成ステップでは、前記主含意スタックが空であって前記回復含意スタックリストが空ではないときには、前記回復含意スタックリストの先頭にある回復含意スタックが前記回復含意スタックリストから削除され、新たな主含意スタックとされ、前記回復含意スタックに対応するキャプチャ衝突は以降のテスト生成過程において無視されてテストベクトルの生成が再開されるリストアステップを含む、請求項１３又は１４記載の生成方法。

【請求項１６】

前記リストアステップは、初期テストベクトル集合で得られる故障検出率が低下しない

ために、対象故障を検出するまで繰り返される、請求項１５記載の生成方法。
【請求項１７】
前記生成ステップにおいて、前記生成手段は、
ビットの論理値を少なくとも１つ変更して未定値とするものであり、
前記新しいテストベクトルにおける未定値のうち少なくとも１つに元の論理値とは異なる論理値を割り当てることにより、スキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させるように前記未定値に論理値が割り当てられる、請求項７から１６のいずれかに記載の生成方法。
【請求項１８】
請求項７から１７のいずれかに記載の生成方法をコンピュータに実行させることが可能なプログラム。
【請求項１９】
請求項１８記載のプログラムをコンピュータが実行することが可能にて記録した記録媒体。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、生成装置、生成方法、生成方法をコンピュータに実行させることが可能なプログラム、及び、このプログラムを記録した記録媒体に関し、特に、論理回路に対するテストベクトル集合を生成する生成装置等に関する。
【背景技術】
【０００２】
図１２に示すように、半導体論理回路は、設計、製造、テストの三段階を経て出荷される。ここで、テストとは、製造された半導体論理回路に対してテストベクトルを印加し、半導体論理回路からテスト応答を観測し、それを期待テスト応答と比較して良品、不良品の判別を行う。その良品率を歩留りと呼び、歩留りは半導体論理回路の品質、信頼性及び製造コストを大きく左右する。
【０００３】
一般に、半導体論理回路は主に順序回路である。順序回路は、アンド（ＡＮＤ）ゲート、ナンド（ＮＡＮＤ）ゲート、オア（ＯＲ）ゲート、ノア（ＮＯＲ）ゲート等の論理素子からなる組合せ回路部と、回路の内部状態を記憶するフリップフロップとよりなる。この場合、組合せ回路部は、外部入力線（ＰＩ）、フリップフロップの出力線である擬似外部入力線（ＰＰＩ）、外部出力線（ＰＯ）、フリップフロップの入力線である擬似外部出力線（ＰＰＯ）を有する。組合せ回路部への入力は、外部入力線より直接与えられるものと、擬似外部入力線を介して与えられるものからなる。また、組合せ回路部からの出力は、外部出力線に直接現れるものと、擬似外部出力線に現れるものからなる。
【０００４】
順序回路の組合せ回路部をテストするために、組合せ回路部の外部入力線と擬似外部入力線から所要のテストベクトルを印加し、組合せ回路部の外部出力線と擬似外部出力線からテスト応答を観測する必要がある。１つのテストベクトルは、外部入力線と擬似外部入力線に対応するビットからなる。また、１つのテスト応答は、外部出力線と擬似外部出力線に対応するビットからなる。
【０００５】
しかし、順序回路のフリップフロップの出力線（擬似外部入力線）と入力線（擬似外部出力線）は一般に外部より直接アクセスできない。従って、組合せ回路部をテストするためには、擬似外部入力線の可制御性及び擬似外部出力線の可観測性に問題がある。
【０００６】
上述の組合せ回路部のテストにおける可制御性及び可観測性の問題を解決する主な手法として、フルスキャン設計がある。フルスキャン設計とは、フリップフロップをスキャン

フリップフロップに置き換えた上で、それらを用いて１本または複数本のスキャンチェーンを形成することである。スキャンフリップフロップの動作はスキャンインネーブル（ＳＥ）信号線で制御される。例えば、ＳＥ＝０のとき、従来のフリップフロップと同じ動作をし、クロックパルスが与えられると、組合せ回路部からの値でスキャンフリップフロップの出力値が更新され、また、ＳＥ＝１のとき、同じスキャンチェーンにある他のスキャンフリップフロップと１つのシフトレジスタを形成し、クロックパルスが与えられると、外部から新しい値がスキャンフリップフロップにシフトインされると同時に、スキャンフリップフロップに現存の値が外部へシフトアウトされる。一般に、同じスキャンチェーンにあるスキャンフリップフロップは同じスキャンインネーブル（ＳＥ）信号線を共有するが、異なるスキャンチェーンのスキャンインネーブル（ＳＥ）信号線は同一の場合もあれば異なる場合もある。

【０００７】

フルスキャン順序回路の組合せ回路部のテストはスキャンシフトとスキャンキャプチャを繰り返すことによって行われる。スキャンシフトは、スキャンインネーブル（ＳＥ）信号が論理値１にされているシフトモードで行われる。シフトモードにおいては、１つまたは複数のクロックパルスが与えられ、外部から１つまたは複数の新しい値がスキャンチェーン内のスキャンフリップフロップにシフトインされる。また、それと同時に、そのスキャンチェーン内のスキャンフリップフロップに現存の１つまたは複数の値が外部へシフトアウトされる。スキャンキャプチャは、スキャンインネーブル（ＳＥ）信号が論理値０にされているキャプチャモードで行われる。キャプチャモードにおいては、１つのスキャンチェーンにあるすべてのスキャンフリップフロップに同時に１つのクロックパルスが与えられ、組合せ回路部の擬似外部出力線の値がすべてのスキャンフリップフロップに取り込まれる。

【０００８】

スキャンシフトは、擬似外部入力線を介して組合せ回路部へテストベクトルを印加するためと、擬似外部出力線を介して組合せ回路部からテスト応答を観測するために用いられる。また、スキャンキャプチャは、組合せ回路部のテスト応答をスキャンフリップフロップに取り込むために用いられる。すべてのテストベクトルに対して、スキャンシフトとスキャンキャプチャを繰り返すことによって、組合せ回路部をテストすることができる。このようなテスト方式はスキャンテスト方式という。

【０００９】

スキャンテスト方式では、組合せ回路部へのテストベクトルの印加は、外部入力から直接行われる部分と、スキャンシフトによって行われる部分とがある。スキャンシフトによって、任意の論理値を任意のスキャンフリップフロップに設定することができるので、擬似外部入力線の可制御性の問題が解決される。組合せ回路部からのテスト応答の観測は、外部出力から直接行われる部分と、スキャンシフトによって行われる部分とがある。スキャンシフトによって、任意のスキャンフリップフロップの出力値を観測することができるため、擬似外部出力線の可観測性の問題が解決される。このように、スキャンテスト方式においては、自動テストパターン生成（ＡＴＰＧ）プログラムを用いてテストベクトル及び期待テスト応答を求めるだけで十分である。

【００１０】

上述のスキャンテスト方式が有効性を有しているにもかかわらず、通常動作時よりテスト時の消費電力が非常に大きいという問題点が存在する。半導体論理回路がＣＭＯＳ回路で構成されていれば、消費電力としては、漏れ電流による静的消費電力と、論理ゲートやフリップフロップのスイッチング動作による動的消費電力とがある。さらに、後者の動的消費電力は、シフト操作時におけるシフト消費電力と、キャプチャ操作時におけるキャプチャ消費電力とがある。

【００１１】

１つのテストベクトルに対して、スキャンシフト時に与えられるクロックパルスの数は一般に多い。例えば、あるスキャンチェーン内のすべてのスキャンフリップフロップに新

しい値を設定するために、最大の場合にスキャンフリップフロップ個数分のクロックパルスを与える必要がある。このため、シフト消費電力が大きくなり、過度な発熱を引き起こすことがある。それによって、半導体論理回路装置を損壊する恐れがある。シフト消費電力の低減手法が盛んに研究されている。

【００１２】

一方、１つのテストベクトルに対して、スキャンキャプチャ時に必要なクロックパルスの数は一般に１つのスキャンチェーンにつき１つである。そのため、スキャンキャプチャ消費電力による発熱は問題にならない。しかし、キャプチャモードにおいて、擬似外部出力線に現れる組合せ回路部のテスト応答がスキャンフリップフロップに取り込まれるとき、テスト応答値とスキャンフリップフロップの現在値が異なれば、対応するスキャンフリップフロップの出力値が変化する。このような出力変化スキャンフリップフロップの数が多ければ、論理ゲートとスキャンフリップフロップのスイッチング動作によって、電源電圧が一時的に低下する。この現象はＩＲ（Ｉ：電流、Ｒ：抵抗）ドロップ現象とも呼ばれる。ＩＲドロップ現象により回路が誤動作し、誤ったテスト応答値がスキャンフリップフロップに取り込まれることがある。これによって、通常時には正常に動作できる半導体論理回路は、テスト時に不良品として判定されてしまうという誤テストが発生する。その結果として、歩留りが低下する。特に、半導体論理回路が超大規模化、超微細化、低電源電圧化した場合、誤テストによる歩留り低下は顕著である。従って、キャプチャ消費電力の低減が必要である。

【００１３】

テスト時に単一クロック信号を用いる場合には、クロックゲーティング手法を用いてスキャンキャプチャ消費電力を低減することができるが、半導体論理回路の物理設計への影響が大きい。また、テスト時に多重クロック信号を用いる場合には、ワンホット手法もしくは多重クロック手法でスキャンキャプチャ消費電力を低減することができるが、前者はテストデータ量が著しく増大し、後者はテストベクトル生成に膨大なメモリ消費が必要になるなどＡＴＰＧへの負担が大きい。従って、スキャンキャプチャ消費電力の低減においては、物理設計への影響、テストデータ量の増加、及びＡＴＰＧへの負担が小さい手法が望ましい。

【００１４】

他方、ドントケアビットを有するテストキューブはＡＴＰＧプログラムによるテストベクトルの生成過程で現れることが多い。これに対して、論理ビット（論理値０又は論理値１を持つビット）のみを含み、ドントケアビットを含まないテスト入力はテストベクトルと呼ばれる。また、ドントケアビットを有しないテストベクトルの集合が与えられる場合、その集合の故障検出率を変えずに、一部のテストベクトルの一部ビットをドントケアビットにすることができる。つまり、ドントケアビット特定プログラムによってテストキューブを得ることもできる。テストキューブが存在する原因は、フルスキャン順序回路の組合せ回路部内の１つ又は複数の対象故障を検出するために、外部入力線と擬似外部入力線における一部のビットに必要な論理値を設定すれば十分であることが多いからである。その残りのビットに０を設定しても１を設定しても、その対象故障の検出に影響を与えないため、そのようなビットはその対象故障にとってドントケアビットになる。

【００１５】

非特許文献１から３はドントケアビットを有しないテストベクトルの集合に対して、その集合の故障検出率を変えずに、一部のテストベクトルの一部ビットをドントケアビットにする技術である。

【００１６】

非特許文献１では、テストベクトルごとにドントケアビットを特定するため、Bit-Strip ingという各ビットに対して順番にドントケアビットになれるかどうかのチェックを行う手法を使っている。この手法では、テストベクトル間の相関関係を完全に無視している。また、この手法は、ビット数に比例して処理時間が長くなる欠点もある。

【００１７】

非特許文献２では、ＸＩＤと呼ばれる手法に基づきドントケアビットを識別する。非特許文献１の技術と異なって、ＸＩＤ手法はテストベクトルごとではなく、与えられるテストベクトル集合内の全てのテストベクトルに対して同時に処理する。具体的には、まず各テストベクトルでしか検出できない故障（必須故障と呼ぶ）を求める。次に、全ての必須故障を検出するために必要な論理値設定をＡＴＰＧの含意操作と論理正当化の手法を応用して求める。その結果、その他の論理ビットをドントケアビットにする。この方法では、全部の入力ビットについてシミュレーションを行なうわけではないため、実行時間においては前述の非特許文献１で提案された手法よりも効率的で高速である。しかし、このドントケアビット手法は制限条件を受けていない。つまり、この手法では、どの論理ビットもドントケアビットにされる可能性がある。

【００１８】

非特許文献３では、前述の非特許文献２の技術と同様、テストベクトルごとではなく、与えられるテストベクトル集合内の全てのテストベクトルに対して同時に処理する。前述の非特許文献２の技術との違いは、どの論理ビットもドントケアビットにされることを許さず、一部の論理ビット（候補ビットと呼ぶ）のみからドントケアビットを特定することにある。候補ビット以外の論理ビット（固定ビットと呼ぶ）からはドントケアビットを特定しない。非特許文献３では、候補ビットと固定ビットからなる制約条件のもとでドントケアビットを特定する。前述の非特許文献２と同様で高速である他、所定の目的を達成するための効率的なドントケアビット特定を行うことができる。明らかに、このような目標達成効率は特定されたドントケアビットの位置に関わるため、候補ビットと固定ビットからなる制約条件を目標に合わせて設定することが大事である。

【００１９】

ドントケアビットを有するテストキューブはあくまでドントケアビットを有しないテストベクトルを生成する過程で現れる中間物である。このため、テストキューブ内のドントケアビットに最終的には論理値０または論理値１を埋め込む必要がある。その埋め込みに際しては、何らかの目的を達成するために必要な論理値（０又は１）をドントケアビットに決定することが一般的に行われる。非特許文献４はスキャンキャプチャ消費電力の低減を目的として、テストキューブのドントケアビットに論理値を決定する技術である。

【００２０】

非特許文献４では、フルスキャン順序回路の組合せ回路部において、様々な手法で得られたドントケアビットを含むテストキューブに対して、３値（論理値０，論理値１，及びドントケアを表すＸ）シミュレーションを行い、そのテストキューブに対するテスト応答をまず求める。次に、擬似入力線ビットと擬似出力線ビットからなるビットペアを、擬似入力線ビットにのみドントケアがあるタイプＡビットペア、擬似出力線ビットにのみドントケアがあるタイプＢビットペア、及び、擬似入力線ビットと擬似出力線ビットの両方にドントケアがあるタイプＣビットペアに分類する。更に、これらのビットペアを順番に１つずつ処理していく。その処理において、タイプＡビットペアの場合、擬似入力線ビットのドントケアビットに対応する擬似出力線ビットの論理値を割り当てることにし、タイプＢビットペアの場合、擬似出力線ビットのドントケアビットに対応する擬似入力線ビットの論理値が現れるように、正当化操作を行って、テストキューブ内のドントケアビットの論理値を決めることにし、タイプＣビットペアの場合、擬似入力線ビットと擬似出力線ビットの両方にあるドントケアビットに同じ論理値（０又は１）が現れるように、擬似入力線に対する論理値の割り当てと擬似出力線に対する正当化操作を行って、テストキューブ内のドントケアビットの論理値を決めることにする。明らかに、非特許文献４の埋め込み技術の特徴は、テストキューブのドントケアビットに論理値を決める際、１つの擬似入力線ビットと１つの擬似出力線ビットからなる１つのビットペアしか考慮していない。このように決定された論理値は全体的に見て必ずしも最適と言えない。

【００２１】

【非特許文献１】R.Sankaralingam and N.A.Touba, “Controlling Peak Power During Scan Testing,” Proceedings of IEEE VLSI Test Symposium, pp.153-15

9,2002.
【非特許文献２】K.Miyase and S.Kajihara, “XID: Don't Care Identification of Test Patterns for Combinational Circuits,” IEEE Transactions on Computer-Aided Design, Vol.23, pp.321-326,2004.
【非特許文献３】K.Miyase, S.Kajihara, I.Pomeranz, and S.Reddy, “Don’t Care Identification on Specific Bits of Test Patterns,” Proceedings of IEEE/ACM International Conference on Computer Design, pp.194-199,2002.
【非特許文献４】X.Wen, H.Yamashita, S.Kajihara, L.-T.Wang, K.Saluja, and K.Kinoshita, “On Low-Capture-Power Test Generation for Scan Testing,” Proceedings of IEEE VLSI Test Symposium, pp.265-270,2005.

【発明の開示】
【発明が解決しようとする課題】
【００２２】
前述のように、フルスキャン順序回路のテストの場合、キャプチャモードにおいて、擬似外部出力線に現れる組合せ回路部のテスト応答がスキャンフリップフロップに取り込まれるとき、テスト応答値とスキャンフリップフロップの現在値が異なれば、対応するスキャンフリップフロップの出力値が変化する。このような出力変化スキャンフリップフロップの数が多ければ、論理ゲートとスキャンフリップフロップのスイッチング動作によって、電源電圧が一時的に低下するＩＲドロップ現象が起こり、それにより回路が誤動作し、誤ったテスト応答値がスキャンフリップフロップに取り込まれることがある。これによって、通常時には正常に動作できる半導体論理回路は、テスト時に不良品として判定されてしまうという誤テストが発生する。その結果として、歩留りが低下する。特に、半導体論理回路が超大規模化、超微細化、低電源電圧化した場合、誤テストによる歩留り低下は顕著である。従って、キャプチャ消費電力の低減が必要である。
【００２３】
上記問題が起こらないようにするためには、図１３において組合せ回路部８０２からスキャンフリップフロップ８０４に取り込まれる論理値と、スキャンフリップフロップ８０４が現在持っている論理値との間での相違数ができるだけ小さい低キャプチャ時消費電力のテストベクトルをテストに用いることが重要である。このようなテストベクトルを生成するために、何らかの手法でドントケアビットを含むテストキューブを作り、更にその中に含まれるドントケアビットに最適な論理値を決定し埋め込む形で最終テストベクトルに仕上げる必要がある。
【００２４】
テストキューブはＡＴＰＧプログラムによるテストベクトルの生成過程で現れることが多い。また、ドントケアビットを有しないテストベクトルの集合が与えられる場合、その集合の故障検出率を変えずに、一部のテストベクトルの一部のビットをドントケアビットにすることができる。特に、与えられたテストベクトルの集合からドントケアビットを特定することによってテストキューブを作ることは、テスト生成フローに与える影響が少ないため、応用において有利である。
【００２５】
しかしながら、テストキューブを作るための従来技術は、様々な欠点を持っている。非特許文献１の技術は、実行時間が長く、作られたテストキューブも必ずしもキャプチャ時消費電力を効果的に低減することができない。非特許文献２の技術は、テスト集合全体を処理対象にするため実行時間が比較的早く、より多くのドントケアビットを特定することができるが、作られたテストキューブは必ずしもキャプチャ時消費電力を効果的に低減することができない。非特許文献３の技術は、テスト集合の論理ビットの一部のみよりドントケアビットを特定することによって、キャプチャ時消費電力を効果的に低減することができるテストキューブを作ることが可能であるが、候補ビットと固定ビットで制約されるドントケアビット特定範囲を如何に定めるかについては何も示されていない。
【００２６】

更に、キャプチャ時消費電力を効果的に低減するために、テストキューブ内のドントケアビットに論理値０または論理値１を埋め込む技術がある。特に、非特許文献４の技術は、擬似入力線ビットのみならず擬似出力線ビットをも考慮して、対応する擬似入力線ビットと擬似出力線ビットの間の論理値の相違を削減するため、他の技術より有利である。しかし、非特許文献４の技術は、ドントケアビットに論理値（０又は１）を決定するときに、一回に１つの擬似出力線ビットと１つの擬似出力線ビットからなる１つのビットペアしか考慮していない。このため、このように決定された論理値は全体的に見て必ずしも最適と言えない。

【００２７】

なお、ここまではキャプチャ時の消費電力低減の問題を説明したが、テストデータ量削制、或いはテストデータの欠陥検出においても、制約が必要な同様の問題がある。

【００２８】

ゆえに、本発明は、上記問題を解決するため、例えば、フルスキャン順序回路に含まれるスキャンセルの出力について、スキャンキャプチャの前後において発生する論理値の相違が低減されるようなテストベクトル集合の生成を行う生成装置、生成方法、生成方法をコンピュータに実行させることが可能なプログラム、及び、このプログラムを記録した記録媒体を提供することを目的とする。

【課題を解決するための手段】

【００２９】

請求項１に係る発明は、論理回路に対する初期テストベクトル集合を変換して新しいテストベクトル集合を生成する生成装置であって、前記論理回路はフルスキャン設計された順序回路であって、前記初期テストベクトル集合の各テストベクトルのうち、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数について所定の基準を超えている対象テストベクトルを特定する特定手段と、前記特定手段が特定した対象テストベクトルの代わりとして、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる新しいテストベクトルを生成する生成手段とを備える。

【００３０】

請求項２に係る発明は、請求項１記載の生成装置であって、前記生成手段は、前記特定手段が特定した対象テストベクトルに対し、対象故障を選択する選択手段を有し、前記選択手段が選択した対象故障を全て検出でき、且つ、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる対象テストベクトルに代わる新しいテストベクトルを生成する。

【００３１】

請求項３に係る発明は、請求項２記載の生成装置であって、前記選択手段は、前記特定手段が特定した対象テストベクトルのみで検出される各故障を、一つの対象テストベクトルのみで検出されるベクトル必須故障と複数の対象テストベクトルで検出されるセット必須故障に区別し、前記ベクトル必須故障についてそれを検出する対象テストベクトルの対象故障として選択するとともに前記セット必須故障についてそれを検出する複数の対象テストベクトルのうちのいずれかの対象故障として選択する。

【００３２】

請求項４に係る発明は、請求項１から３のいずれかに記載の生成装置であって、前記生成手段は、前記順序回路に対応する組合せ回路に対し、テスト生成過程において前記組合せ回路のある擬似外部入力の値と前記擬似外部入力に対応する擬似外部出力の値とが異なることを表すキャプチャ衝突を定義し、前記キャプチャ衝突が生じたときにもバックトラックを行って、前記特定手段が特定した対象テストベクトルに代わる新しいテストベクトルを生成する。

【００３３】

請求項５に係る発明は、論理回路に対する初期テストベクトル集合を変換して新しいテ

ストベクトル集合を生成する生成装置であって、前記論理回路はフルスキャン設計された順序回路であって、前記順序回路に対応する組合せ回路に対し、テスト生成過程において前記組合せ回路のある擬似外部入力の値と前記擬似外部入力に対応する擬似外部出力の値とが異なることを表すキャプチャ衝突を定義し、前記キャプチャ衝突が生じたときにもバックトラックを行って、前記初期テストベクトル集合のうちの対象テストベクトルの代わりとして、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる新しいテストベクトルを生成する生成手段を備える。

【００３４】

請求項６に係る発明は、請求項４又は５記載の生成装置であって、前記生成手段は、テスト生成過程における主含意スタックとは別に、前記キャプチャ衝突が生じたときに更新される回復含意スタックを用いる。

【００３５】

請求項７に係る発明は、論理回路に対する初期テストベクトル集合を変換して新しいテストベクトル集合を生成する生成方法であって、前記論理回路はフルスキャン設計された順序回路であって、特定手段が、前記初期テストベクトル集合の各テストベクトルのうち、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数について所定の基準を超えている対象テストベクトルを特定する特定ステップと、生成手段が、前記特定手段が特定した対象テストベクトルの代わりとして、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる新しいテストベクトルを生成する生成ステップとを含む。

【００３６】

請求項８に係る発明は、請求項７記載の生成方法であって、前記生成ステップは、前記特定された対象テストベクトルに対し、対象故障を選択する第１のステップと、前記特定手段が特定した対象テストベクトルに対し、前記選択された対象故障を全て検出でき、且つ、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させるように対象テストベクトルに代わる新しいテストベクトルを生成する第２のステップとを含む。

【００３７】

請求項９に係る発明は、請求項８記載の生成方法であって、前記第１のステップは、前記特定された対象テストベクトルのみで検出される各故障を、一つの対象テストベクトルのみで検出されるベクトル必須故障と複数の対象テストベクトルで検出されるセット必須故障に区別し、前記ベクトル必須故障についてそれを検出する対象テストベクトルの対象故障として選択するとともに前記セット必須故障についてそれを検出する複数の対象テストベクトルのうちのいずれかの対象故障として選択する。

【００３８】

請求項１０に係る発明は、請求項９記載の生成方法であって、前記セット必須故障がいずれの対象テストベクトルについての対象故障として選択されるかについての判断に、前記複数の選択されるべき故障の間の重なり具合を示す演算式に基づく結果が用いられる。

【００３９】

請求項１１に係る発明は、請求項７から１０のいずれかに記載の生成方法であって、前記生成ステップでは、前記順序回路に対応する組合せ回路に対し、テスト生成過程において前記組合せ回路のある擬似外部入力の値と前記擬似外部入力に対応する擬似外部出力の値とが異なることを表すキャプチャ衝突を定義し、前記キャプチャ衝突が生じたときにもバックトラックを行って、前記特定手段が特定した対象テストベクトルに代わる新しいテストベクトルを生成する。

【００４０】

請求項１２に係る発明は、論理回路に対する初期テストベクトル集合を変換して新しいテストベクトル集合を生成する生成方法であって、前記論理回路はフルスキャン設計され

た順序回路であって、生成手段が、前記順序回路に対応する組合せ回路に対し、テスト生成過程において前記組合せ回路のある擬似外部入力の値と前記擬似外部入力に応答する擬似外部出力の値とが異なることを表すキャプチャ衝突を定義し、前記キャプチャ衝突が生じたときにもバックトラックを行って、前記初期テストベクトル集合のうちの対象テストベクトルの代わりとして、前記順序回路に含まれるスキャンセルの出力についてスキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させる新しいテストベクトルを生成する生成ステップを含む。

【００４１】

請求項１３に係る発明は、請求項１１又は１２記載の生成方法であって、前記生成ステップでは、テスト生成過程における主含意スタックとは別に、前記キャプチャ衝突が生じたときに更新される回復含意スタックリストを用いる。

【００４２】

請求項１４に係る発明は、請求項１３記載の生成方法であって、前記生成ステップでは、前記キャプチャ衝突が生じれば、現在の主含意スタックの複製が前記キャプチャに対応する回復含意スタックとして前記回復含意スタックリストの先頭に追加される。

【００４３】

請求項１５に係る発明は、請求項１３又は１４記載の生成方法であって、前記生成ステップでは、前記主含意スタックが空であって前記回復含意スタックリストが空ではないときには、前記回復含意スタックリストの先頭にある回復含意スタックが前記回復含意スタックリストから削除され、新たな主含意スタックとされ、前記回復含意スタックに対応するキャプチャ衝突は以降のテスト生成過程において無視されてテストベクトルの生成が再開されるリストアステップを含む。

【００４４】

請求項１６に係る発明は、請求項１５記載の生成方法であって、前記リストアステップは、初期テストベクトル集合で得られる故障検出率が低下しないために、対象故障を検出するまで繰り返される。

【００４５】

請求項１７に係る発明は、請求項７から１６のいずれかに記載の生成方法であって、前記新しいテストベクトルに未定値が含まれている場合、スキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させるように前記未定値に論理値が割り当てられる。

【００４６】

請求項１８に係る発明は、請求項７から１７のいずれかに記載の生成方法をコンピュータに実行させることが可能なプログラム。

【００４７】

請求項１９に係る発明は、請求項１８記載のプログラムをコンピュータが実行することが可能にて記録した記録媒体。

【発明の効果】

【００４８】

本発明によれば、例えば、論理回路におけるフルスキャン順序回路に含まれるスキャンセルの出力について、スキャンキャプチャの前後において発生する論理値の相違がより効果的に低減させることができる。それにより、スキャンキャプチャ時の消費電力を抑えることで、誤テストを回避できる。従って、本来正常な動作を行う半導体論理回路をテストにおいて不良品として廃棄するような良品率の低下を解消することができる。

【００４９】

また、本発明における生成装置及び生成方法は、論理回路のテスト設計フローを変更することもハードウェア追加によって回路面積を増加させることもない。このため、本発明における生成装置及び生成方法は、キャプチャ時の誤テストを回避するために非常に効果的であると言える。

【００５０】

さらに、本発明における生成装置及び生成方法は、クロックの種類に依存しないので、テスト時に単一クロック信号を用いる場合にクロックゲーティング手法を採用する場合のようにテストデータ量が著しく増大することはなく、テストデータ量の低減にも効果がある。

【００５１】

さらに、本発明における生成装置及び生成方法は、論理回路の故障検出率の低下はないため、テストデータの欠陥検出能力の向上に効果がある。

【発明を実施するための最良の形態】

【００５２】

本発明の実施の形態を以下に示す。

【００５３】

図１を参照して、本発明の背景技術となる一般的なフルスキャン回路について説明する。

【００５４】

図１（ａ）は、一般的なフルスキャン回路の構成を示した模式図である。このフルスキャン回路は、組合せ回路部分１００とフルスキャン順序回路のスキャンフリップフロップ１０２とから構成される。組合せ回路部分１００は、外部入力線（ＰＩ）、スキャンフリップフロップの出力線である擬似外部入力線（ＰＰＩ）、外部出力線（ＰＯ）、フリップフロップの入力線である擬似外部出力線（ＰＰＯ）を有する。組合せ回路部分１００へのテストベクトルｖは、外部入力線より直接与えられる部分＜ｖ：ＰＩ＞と、擬似外部入力線を介して与えられる部分＜ｖ：ＰＰＩ＞からなる。＜ｖ：ＰＰＩ＞はスキャンシフトによってスキャンフリップフロップ１０２に設定される。また、組合せ回路部分１００からの出力はテストベクトルｖに対するテスト応答ｆ（ｖ）であり、外部出力線に直接現れる部分＜ｆ（ｖ）：ＰＯ＞と、擬似外部出力線に現れる部分＜ｆ（ｖ）：ＰＰＯ＞からなる。＜ｆ（ｖ）：ＰＰＯ＞はスキャンキャプチャによってスキャンフリップフロップ１０２に取り込まれる。

【００５５】

図１（ｂ）は図１（ａ）におけるスキャンフリップフロップ１０２におけるスキャンキャプチャの前と後において論理値に相違が発生する場合の一例を示す。

【００５６】

図１（ｂ）において、テストベクトル＜ｖ：ＰＰＩ＞の要素である一つのビットａと、それに対応するテスト応答＜ｆ（ｖ）：ＰＰＯ＞が、スキャンフリップフロップ１０２で異なる論理値を取ると、キャプチャモードの際に論理値の相違（以下、遷移とする）が発生する。ある一つのテストベクトルに対する遷移の数は、そのテストベクトルを原因とした組合せ回路部分１００を含めた回路全般で発生する消費電力と深く関係しているため、テストベクトルに対するキャプチャ時の遷移の数を削減することで、キャプチャ時の消費電力を低減することができる。

【００５７】

次に、本発明の実施の形態に係る生成装置の構成を説明する。図２は、本発明の実施の形態に係る生成装置の概略ブロック図である。

【００５８】

生成装置２００は、初期テストベクトル集合生成部２０２と対象テストベクトル特定部２０４とテストベクトル集合変換部２０６とから構成されている。テストベクトル集合変換部２０６は、対象故障選択部２０８とテストキューブ生成部２１０と論理値割当部２１２と最終テストベクトル集合生成部２１４とから構成される。また、入出力データとして初期テストベクトル集合２１６（Ｔ）と最終テストベクトル集合２１８（Ｔ’）があり、初期テストベクトル集合２１６は初期テストベクトル集合生成部２０２により予めＡＴＰＧ等の手段により生成されたテストベクトル集合である。

【００５９】

図２の生成装置２００の処理について簡単に説明する。まず初期テストベクトル集合２

１６が生成されると、対象テストベクトル特定部２０４により初期テストベクトル集合２１６の中から変換の対象とされるテストベクトルが特定される。次に、対象故障選択部２０８が、対象テストベクトル特定部２０４により特定されたテストベクトルにより検出される故障を選択し、その選択された故障はベクトル必須故障とセット必須故障に区別される。テストキューブ生成部２１０は選択された故障の中で未検出となる故障がないようにテストキューブを生成する。生成されたテストキューブは、論理値割当部２１２によりドントケアビットに論理値が割り当てられ、最終テストベクトル集合生成部２１４により最終テストベクトル集合が生成される。

【００６０】

なお、この時変換の対象となる対象テストベクトル以外の非対象テストベクトルで検出される故障も検出されるため、初期テストベクトル集合での検出率が低下することなくテストベクトルを生成することができる。

【００６１】

次に、図３を参照して本発明の実施の形態に係る生成装置の処理を説明する。

【００６２】

図３は、本発明の実施の形態に係るテストベクトル生成手法による生成装置の処理フロー図である。

【００６３】

ここで、ＬＣＰ（Ｌｏｗ　Ｃａｐｔｕｒｅ　Ｐｏｗｅｒ）とはスキャンキャプチャ時に消費電力が低いことであり、本発明の実施の形態に係る生成装置の処理は、大きく分けて図３における段階１と段階２の２つの処理により構成される。段階１は、ステップＳＴ３００の処理であり、初期テストベクトル集合Ｔを生成する手段として、従来の縮退故障検出用ＡＴＰＧを用いる。この縮退故障検出用ＡＴＰＧは故障検出率を満たす必要最低限のサイズを持つテストベクトル集合を生成する。

【００６４】

段階２は、初期テストベクトル集合Ｔに含まれる、ＨＣＰ（Ｈｉｇｈ　Ｃａｐｔｕｒｅ　Ｐｏｗｅｒ）の原因となる全てのテストベクトルが特定される。ＨＣＰとは、スキャンキャプチャ時に消費電力が高いことであり、このＨＣＰとなるテストベクトルが、ＣＴ（ｖ”）＜ｃ＿ｌｉｍｉｔとなるように新しいテストベクトルｖ”に置き換えられる。

【００６５】

ここで、ＣＴ（ｖ”）は、テストベクトルｖ”によるスキャンキャプチャの前後において論理値に相違が発生するビット（以下遷移ビットとする）の数であり、ｃ＿ｌｉｍｉｔは、遷移ビット数の目標とすべき上限値であり、遷移ビット数がｃ＿ｌｉｍｉｔ以下であれば、誤テストが起こらないと想定できる数値であり、遷移ビット数がｃ＿ｌｉｍｉｔ以上であれば誤テストを起こす要因となり得る数値である。このｃ＿ｌｉｍｉｔは回路設計時の消費電力量の見積もりまたは経験則等から決定される。

【００６６】

図３に戻って、まずステップＳＴ３００では、図２における初期テストベクトル集合生成部２０２により、初期テストベクトル集合Ｔが従来の縮退故障検出用ＡＴＰＧを用いて生成される。ステップＳＴ３０１では、図２における対象テストベクトル特定部２０４により、ＣＴ（ｖ）＞ｃ＿ｌｉｍｉｔとなる初期テストベクトル集合Ｔに含まれる全てのテストベクトルｖを要素に持つテストベクトル集合Ｔｔａｒを得る。次に、ステップＳＴ３０２では、図２における対象故障選択部２０８により、テストベクトル集合Ｔｔａｒに含まれる各テストベクトルｖに対して、少なくとも故障リストＦｔａｒ（ｖ）に含まれる全ての故障が検出されると、故障検出率が低下しない故障リストＦｔａｒ（ｖ）を得る。

【００６７】

次に、ステップＳＴ３０３では、Ｔｔａｒ内に未処理のテストベクトルｖがあるか否かが判定され、なければ処理を終了する。Ｔｔａｒ内に未処理のテストベクトルｖがあれば、次のステップＳＴ３０４に進む。ステップＳＴ３０４では、各Ｔｔａｒに含まれる各テストベクトルｖに対する故障リストＦｔａｒ（ｖ）に未検出の故障ｆが含まれるか否かが

判定される。未検出の故障ｆがあればステップＳＴ３０５に進む。

【００６８】

ステップＳＴ３０５では、図２におけるテストキューブ生成部２１０により、スキャンキャプチャ時の消費電力の削減を考慮した故障リストＦｔａｒ（ｖ）に含まれる全ての故障ｆを検出するドントケアビットを含むテストキューブｖ’がＬＣＰを目的とした不定値割当手法により新しく生成される。次にステップＳＴ３０４に戻って、再び各Ｔｔａｒに含まれる各テストベクトルｖに対する故障リストＦｔａｒ（ｖ）に未検出の故障ｆが含まれるか否かが判定される。未検出の故障ｆがなければステップＳＴ３０６に進み、図２における論理値割当部２１２によりテストキューブｖ’に含まれるドントケアビットに対して、スキャンキャプチャ時の消費電力が削減されるように論理値が割り当てられ、ドントケアビットを含まないテストベクトルｖ”が得られる。そして、ステップＳＴ３０７で、図２における最終テストベクトル集合生成部２１４により、初期テストベクトル集合Ｔに含まれるテストベクトルｖをテストベクトルｖ”で置き換えることにより、最終テストベクトル集合Ｔ’を得る。以上の処理により、初期テストベクトル集合Ｔと最終テストベクトル集合Ｔ’の故障検出率は同じであり、Ｔ’のほうがスキャンキャプチャ時の消費電力が低くなる。

【００６９】

図３において、ステップＳＴ３０１、ステップＳＴ３０２及びステップＳＴ３０５が本発明での提案手法であり、それ以外は従来手法である。

【００７０】

次に、図３におけるステップＳＴ３０１の処理について詳細に説明する。

【００７１】

図３におけるステップＳＴ３０１は、初期テストベクトル集合Ｔに含まれる、ＨＣＰの原因となる全てのテストベクトルを特定し、それらのテストベクトル集合をＴｔａｒとして保存する。その目的は、すでにＬＣＰとなるテストベクトルに対する不必要な処理を避けるためである。

【００７２】

この処理は、消費電力解析に基づいて行うのが最良であるが、このアプローチでは時間が掛かり、レイアウト情報がこの段階では使用することができないかもしれない。従って、ＣＴ（ｖ）＞ｃ＿ｌｉｍｉｔとなるテストベクトルｖを特定する。

【００７３】

次に、図３におけるステップＳＴ３０２の処理について詳細に説明する。

【００７４】

この処理では、テストベクトル集合Ｔｔａｒに含まれる各テストベクトルがＬＣＰとなるテストベクトルにそれぞれ置き換えられる必要がある。また、テストベクトル集合Ｔｔａｒに含まれるテストベクトルｖと置き換える新しいテストベクトルを生成するために、故障検出率が低下しない故障リストＦｔａｒ（ｖ）を選択する必要がある。

【００７５】

故障を選択する場合、以下の条件を満たす必要がある。

【００７６】

（条件１）以下の式は、テストベクトル集合Ｔｔａｒに含まれるテストベクトルによってのみ検出される全ての故障を含むべきである。これは故障検出率が低下しないことを保証する。

【００７７】

【数１】

$$\sum_{v \in T_{tar}} Ftar(v)$$

【００７８】

（条件２）Ｆｔａｒ（ｖ）は、ＬＣＰとなるテストキューブで検出され易い故障を含むべきである。

【００７９】

（条件３）Ｆｔａｒ（ｖ）は、できるだけ小さい集合であるべきである。

【００８０】

図４は、上記条件１から条件３の３つの条件を満たすＦｔａｒ（ｖ）を得るために、図３におけるステップＳＴ３０２の処理が行われた結果の一例を示す。

【００８１】

図４において、初期テストベクトル集合Ｔ＝｛ｖ１，ｖ２，ｖ３，ｖ４，ｖ５｝とし、ｖ１、ｖ４及びｖ５をＨＣＰであるテストベクトルとする。すなわち、Ｔｔａｒ＝｛ｖ１，ｖ４，ｖ５｝である。ここで、故障が１２個存在することとし、故障シミュレーションによって得られた故障検出の情報を図４に示す。ここで、対象テストベクトルの遷移ビットの数を減らすためには、対象テストベクトルが対象とする故障数が少ないほうがよい。従って、図４中の下線で示される故障ｆ１１及びｆ１２はテストベクトル集合Ｔｔａｒに含まれないテストベクトルｖ２及びｖ３によって検出されるため、故障ｆ１１及びｆ１２は対象テストベクトルが対象とする故障集合から除外される。つまり、テストベクトル集合Ｔｔａｒに含まれるテストベクトルによってのみ検出される故障集合はＴＡ＝｛ｆ１，ｆ４～ｆ１０｝となる。このことから、ＴＡに含まれる全ての故障が検出できれば初期テストベクトル集合から得られる故障検出率と比較して低下しない。

【００８２】

ＴＡに含まれる全ての故障は２つの種類に分類される。一方は、テストベクトル集合Ｔｔａｒに含まれる唯一のテストベクトルでのみ検出される故障（以下ベクトル必須故障とする）であり、図４に示す丸で囲まれた故障が、ベクトル必須故障である。また、他方は、テストベクトル集合Ｔｔａｒ内の複数のテストベクトルで検出され、初期テストベクトル集合ＴからＴｔａｒのテストベクトル集合を除いたテストベクトル集合（Ｔ－Ｔｔａｒ）では検出できない故障（以下セット必須故障とする）であり、図４に示す四角で囲まれた故障が、セット必須故障である。

【００８３】

テストベクトルｖの全てのベクトル必須故障は、故障リストＦｔａｒ（ｖ）に含まれるべきある。例えばベクトル必須故障ｆ１及びｆ６は、Ｆｔａｒ（ｖ１）に含まれる必要がある。一方、セット必須故障は、故障リストＦｔａｒ（ｖ）または故障検出率を低下させずにそのセット必須故障を検出する他のテストベクトルの故障リストに含まれるべきである。例えば、ｆ９はテストベクトルｖ１及びｖ４によって検出されるセット必須故障であり、このｆ９はＦｔａｒ（ｖ１）またはＦｔａｒ（ｖ４）のいずれかに含まれる必要がある。

【００８４】

各セット必須故障は、新しくＬＣＰとなるテストキューブを生成する際に、故障の検出が容易となるようなＦｔａｒ（ｖ）に含めるようにする。故障検出の容易性を算出するために以下のような新しい概念を用いる。

【００８５】

フルスキャン回路に２つの故障ｆａとｆｂがあるとする。故障ｆａ及び故障ｆｂから構造的に到達可能な擬似外部入力（ＰＰＩｓ）の集合をそれぞれＲＩ（ａ）及びＲＩ（ｂ）とし、故障ｆａ及び故障ｆｂから構造的に到達可能な擬似外部出力（ＰＰＯｓ）の集合をそれぞれＲＯ（ａ）及びＲＯ（ｂ）とする。故障ｆａ及び故障ｆｂから到達可能なＰＰＩｓ及びＰＰＯｓの重なりの度合いをｏｄ（ｆａ，ｆｂ）と表して以下のように定義する。

【００８６】

【数２】

$$od(fa,fb)=\sum_{i=a,b}\frac{|RO(a)\cap RO(b)|}{RO(i)}+\sum_{i=a,b}\frac{|RI(a)\cap RI(b)|}{RI(i)}$$

【００８７】
図５は、故障ｆａ及び故障ｆｂから到達可能なＰＰＩｓ及びＰＰＯｓの重なりの度合いの概念を示した図である。
【００８８】
図５において、ｏｄ（ｆａ，ｆｂ）の値が大きくなると、故障ｆａ及び故障ｆｂから到達可能なＰＰＩｓ及びＰＰＯｓの重なりの度合いが大きくなる。また、故障ｆａ及び故障ｆｂを同時に検出するテストキューブを生成する際に、スキャンキャプチャ時の遷移ビット数を削減することは困難である可能性を示している。
【００８９】
現時点でＦｔａｒ（ｖ）に含まれないセット必須故障ｆを、現時点での故障リストがＦｔａｒ（ｖ）＝｛ｆｎ１，ｆｎ２，・・・，ｆｎｐ｝であるテストベクトルｖで検出可能とする。まず、ｏｄ（ｆ，ｆｎ１）、ｏｄ（ｆ，ｆｎ２）、・・・、ｏｄ（ｆ，ｆｎｐ）を計算し、以下の式により各故障から到達可能なＰＰＩｓ及びＰＰＯｓの重なりの度合いの平均を以下のように求める。
【００９０】
【数３】

$$aod(f,Ftar(v))=\sum_{i=1,2,\ldots,p}od(f,fn_i)\ /|Ftar(v)|$$

【００９１】
テストベクトルｖｍ１、ｖｍ２、・・・、ｖｍｓで検出されるセット必須故障ｆを何れのテストベクトルで検出すべきかを決定するために、ａｏｄ（ｆ，Ｆｔａｒ（ｖｍ１））、ａｏｄ（ｆ，Ｆｔａｒ（ｖｍ２））、・・・、ａｏｄ（ｆ，Ｆｔａｒ（ｖｍｓ））を計算し、各故障から到達可能なＰＰＩｓ及びＰＰＯｓの重なりの度合いの平均が最小となるテストベクトルの故障リストに故障ｆを含める。
【００９２】
例えば、図４においてｖ１及びｖ４の両方で検出されるセット必須故障ｆ９を含める故障リストを決定する際、Ｆｔａｒ（ｖ１）及びＦｔａｒ（ｖ４）はそれぞれ｛ｆ１，ｆ６｝及び｛ｆ４，ｆ７，ｆ８｝である。仮にａｏｄ（ｆ９，Ｆｔａｒ（ｖ１））＜ａｏｄ（ｆ９，Ｆｔａｒ（ｖ４））であれば、セット必須故障ｆ９はＦｔａｒ（ｖ１）に含められる。図４には最終的な結果を示す。
【００９３】
なお、本発明の実施の形態においては、故障ｆａ及び故障ｆｂから到達可能なＰＰＩｓ及びＰＰＯｓの重なりの度合いによりセット必須故障を割り振ったが、各テストベクトルが検出する故障数により割り振るようにしてもよい。例えば、割り振る対象となるテストベクトルの故障数を比較して、故障数が少ない方にセット必須故障を割り振り、対象とする故障数がほぼ同数になるようにする。
【００９４】
次に、図３におけるステップＳＴ３０５の処理について詳細に説明する。
【００９５】
対象故障リストＦｔａｒ（ｖ）が、ＨＣＰの原因となる１つのテストベクトルｖに対して得られた後、次のステップにおいて、図３におけるステップＳＴ３０５のキャプチャ時

の消費電力を感知するテストキューブ生成処理（以下ＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）とする）が、Ｆｔａｒ（ｖ）中の全ての故障を検出するＬＣＰとなるテストキューブを生成する。

【００９６】

図６は、図２におけるテストキューブ生成部２１０の構成を示したブロック図である。

【００９７】

テストキューブ生成部２１０は、バックトラック判定処理部６００とバックトラック処理部６０２とテストキューブ生成処理部６０４とから構成される。バックトラック判定部６００は、Ｄ衝突検出部６０６とＣ衝突検出部６０８とから構成される。テストキューブ生成処理部６０４は、主含意スタック処理部６１０と回復含意スタック処理部６１２と回復含意スタックリスト処理部６１４と主含意スタック６１６（Ｓｉ）と回復含意スタック６１８（Ｓ）と回復含意スタックリスト６２０（Ｌ（Ｓ））とから構成される。

【００９８】

図６において、初期状態のテストキューブｖを基にバックトラック判定処理部６００によりバックトラック処理を行うか否かが判定され、バックトラック行わないと判定された場合はテストキューブ生成部６０４により、テストキューブが生成される。

【００９９】

図７は、ＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）の処理の手順を示したフロー図である。

【０１００】

一般的に、ＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）はＰＯＤＥＭのアルゴリズムを基本としているが、本発明においてはステップＳＴ７０２、ステップＳＴ７０５、ステップＳＴ７０７、ステップＳＴ７０８及びステップＳＴ７０９の処理が改良点である。この改良点はキャプチャ衝突（以下、Ｃ衝突とする）と回復含意スタックという２つの基本概念を基にしており、ＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）が、故障ｆを検出できるテストキューブを生成し、同時にできるだけ多くのテストキューブ中のビットに関して遷移ビットの数を減らすようにする。以下に詳細に説明する。

【０１０１】

従来からある、ＰＯＤＥＭアルゴリズムに基づいた組合せ回路用ＡＴＰＧアルゴリズムは、Ｘパスチェック（Ｘ－ｐａｔｈ－ｃｈｅｃｋｉｎｇ）の際に故障検出情報の衝突（以下、Ｄ衝突とする）を発見したときには、後戻り（以下、バックトラックとする）を生じる。Ｄ衝突とは、故障検出において経路を活性化するための未定値を含む経路が、Ｄ先頭信号線（Ｄフロンティア）のゲートといずれのＰＯまたはＰＰＯにも存在しないことである。

【０１０２】

本発明の実施の形態におけるＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）においては、Ｃ衝突という新しいバックトラック条件を導入する。Ｃ衝突とは、あるＰＰＩとそれに対応するＰＰＯが異なる値を持つことであり、Ｃ衝突が起こるということはスキャンキャプチャの前後において遷移が発生していることを意味する。Ｃ衝突について図８を用いて詳細を説明する。

【０１０３】

図８は、Ｃ衝突の概念を示した図である。

【０１０４】

図８において、フリップフロップ数がｎでありＣ衝突の数もｎである。ｉ番目のフリップフロップに対応するＰＰＩとＰＰＯの信号線上のＣ衝突はＣｉである。どのＸパスチェックの失敗に対しても、Ｄ衝突は１回だけである。

【０１０５】

スキャンキャプチャ時の電力消費の影響を考慮して、Ｃ衝突がＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）でチェックされる。Ｃ衝突Ｃiの影響を評価する単純な

発見法は、スキャン回路内のi番目のフリップフロップの出力から到達可能な組合せ回路部分のゲートの数を数えることである。

【０１０６】

図７に戻って、まずステップＳＴ７００で検出対象の故障が検出できたか否かが判定される。検出できればテストベクトル集合の生成に成功して処理を終了する。検出できなければ次のステップＳＴ７０１の処理に進む。次に、ステップＳＴ７０１でＤ衝突またはステップＳＴ７０２でＣ衝突が検出された場合、ＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）は、ステップＳＴ７０４で主含意スタックが空か否かを判定し、空である場合は、Ｃ衝突があれば現在の主含意スタックの複製を回復含意スタックリストの先頭に追加して、ステップＳＴ７０６でバックトラックを行なう。しかしながら、Ｄ衝突とＣ衝突は以下の理由で根本的に異なる。

【０１０７】

Ｄ衝突により全ての探索空間が探索された場合は、テスト生成は失敗する。しかしながら、全ての探索空間が探索される前に少なくとも一つのＣ衝突があれば、Ｃ衝突が無視される時テスト生成は成功するかも知れない。Ｃ衝突を無視して生成されたテストキューブは、検出対象の故障を検出できるが、遷移ビットの数を減らすことはできない。

【０１０８】

Ｃ衝突のチェックは、遷移ビット数を削減するために役立つが、対象故障検出のためのテストキューブ生成を阻止するＣ衝突を防ぐ必要がある。従って、本発明の実施の形態では、以下の２種類の含意スタックを用意する。１つは従来のＰＯＤＥＭに基づいたＡＴＰＧアルゴリズムで使用されるものと同様の主含意スタックであり、探索空間の管理に使用される。一方は、回復含意スタックであり、Ｃ衝突が見つかった際に生成される主含意スタックの複製である。複数のＣ衝突が発生する可能性があるため、複数の回復含意スタックが存在するかもしれない。これらのスタックは、回復含意スタックリストと呼ばれるリストに置かれる。

【０１０９】

図７のステップＳＴ７０１でＤ衝突またはステップＳＴ７０２でＣ衝突が検出された場合、ステップＳＴ７０４で主含意スタックが空か否かが判定され、空であればステップＳＴ７０７で回復含意スタックリストが空か否かが判定される。回復含意スタックリストが空であれば、テスト生成に失敗して処理を終了する。回復含意スタックリストが空でなければ、少なくとも一つのＣ衝突が発生し、現在のテスト生成処理の失敗の原因であることを意味する。この場合、ステップＳＴ７０８で回復含意スタックリストの先頭または至近のスタックＳが回復含意スタックリストから削除され、主含意スタックとして回復される。次にステップＳＴ７０９で、スタックＳに一致するＣ衝突は無視され、ステップＳＴ７１０の処理を行ってテスト生成は再開される。このような処理により検出対象となる故障を検出し、同時にできるだけ多くの遷移ビット数が削減されるようなテストキューブが生成される。

【０１１０】

図７のステップＳＴ７０１でＤ衝突またはステップＳＴ７０２でＣ衝突が検出されなかった場合ステップＳＴ７０３でｏｂｊｅｃｔｉｖｅ（）、ｂａｃｋｔｒａｃｅ（）、及び、ｉｍｐｌｙ（）のＰＯＤＥＭを基にした一部のテスト生成処理が行われ、ステップＳＴ７００に戻り再び検出対象となる故障を検出できたか否かが判定される。また、同様にステップＳＴ７０６でバックトラックが行われた場合及びＳＴ７１０でテスト生成が再開された場合もステップＳＴ７００に戻り再び検出対象となる故障を検出できたか否かが判定される。

【０１１１】

図９に、ＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）による、テストキュ

ーブ生成過程の例を示す。図９において、Ａ、Ｂ、・・・、ＧはＰＰＩである。ｂａｃｋｔｒａｃｅ（）はＡからＧの探索順でテストキューブを生成する間、それらの論理値を決定すると仮定する。これに加えて、主含意スタックをＰＳとする。

【０１１２】
図９において、ＰＳ＝＜Ａ：０，Ｂ：１，Ｃ：０＞とし、Ｄ衝突が発生した場合を示す。これを「Ｄ」とする。この時、バックトラックでＣに論理値１を割り当てる。次に、ｂａｃｋｔｒａｃｅ（）はＤに論理値０を割り当て、ＰＳ＝＜Ａ：０，Ｂ：１，Ｃ：１，Ｄ：０＞となった場合、Ｃ衝突「Ｃ１」が発生する。この場合、Ｃ衝突「Ｃ１」で示されるＰＳの複製が回復含意スタックリストに置かれる。次に、バックトラックでＤに論理値１を割り当てる。同様にして、ＰＳ＝＜Ａ：０，Ｂ：１，Ｃ：１，Ｄ：１，Ｅ：０＞となった場合、Ｃ衝突「Ｃ２」が発生する。そして、Ｃ衝突「Ｃ２」で示されるＰＳの複製が回復含意スタックリストに置かれる。最後に、ＰＳはＤ衝突のため空になる。

【０１１３】
図１０において、回復含意スタックリストの一番上にあるスタックＣ２が、主含意スタックとして回復し、Ｃ衝突Ｃ２を無視してテスト生成を再開する。テスト生成の結果、テストキューブは＜Ａ，Ｂ，Ｃ，Ｄ，Ｅ，Ｆ，Ｇ＞＝＜０，１，１，１，０，１，Ｘ＞となる。このテストキューブは、検出対象故障を検出するだけでなく、Ｃ衝突「Ｃ１」に一致する遷移を回避する。

【０１１４】
なお、上記実施の形態において、新しく生成されたテストベクトルに未定値が含まれている場合、スキャンキャプチャの前後において論理値に相違が発生するビットの数を減少させるように未定値に論理値が割り当てられる。

【０１１５】
以下に、本発明の実験結果を示す。

【０１１６】
図３に示されるＬＣＰとなるテスト生成の新しい手法を実装し、ＩＳＣＡＳ’８９回路に対して実験を行った。その結果を図１１に示す。

【０１１７】
キャプチャ時に遷移する最大ビット数は、平均３１．２％削減された。これは、従来技術のＸ－ｆｉｌｌｉｎｇ（X.Wen, H.Yamashita, S.Kajihara, L.-T.Wang, K.Saluja, K.Kinoshita、「On Low-Capture-Power Test Generation for Scan Testing」、Proc.VLSI Test Symp.、2005、pp.265-270）で示される２１．６％に比べて大きい数値である。本発明による手法では、テストベクトル数が増加するが、これは現在の実装では故障検出の順序を無視しており、テストベクトルｖに対する対象故障Ｆｔａｒ（ｖ）を検出するために複数のテストパターンが生成される可能性があるからである。しかし、初期テストベクトルを生成した際の故障検出の順序を利用すれば、この問題が解決され得る。

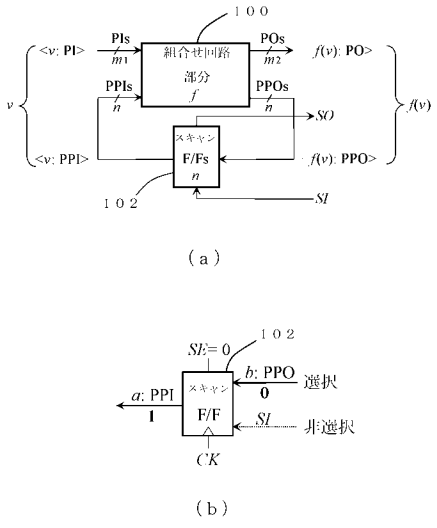
【０１１８】
なお、上記まででではＰＯＤＥＭアルゴリズムに基づいたＡＴＰＧアルゴリズムに適用して説明したが、Ｄアルゴリズム、ＦＡＮアルゴリズム、拡張Ｄアルゴリズム、９値法、ＲＴＰ（Ｒｅｖｅｒｓｅ Ｔｉｍｅ Ｐｒｏｃｅｓｓｉｎｇ）法等の他のＡＴＰＧアルゴリズムにも適用可能である。

【図面の簡単な説明】

【０１１９】
【図１】本発明の背景技術となる一般的なフルスキャン回路の構成を示した模式図である。
【図２】本発明の実施の形態に係る生成装置の概略ブロック図である。
【図３】本発明の実施の形態に係るＬＣＰとなるテストベクトル生成手法による生成装置の処理フロー図である。

【図４】図３におけるステップＳＴ３０２の処理が行われた結果の一例を示した図である。
【図５】故障から到達可能な擬似外部入力（出力）線の重なりの度合いの概念を示した図である。
【図６】図２におけるテストキューブ生成部２１０の構成を示したブロック図である。
【図７】ＣＡ＿ｔｅｓｔ＿ｃｕｂｅ＿ｇｅｎｅｒａｔｉｏｎ（ｆ）の処理の手順を示したフロー図である。
【図８】Ｃ衝突の概念を示した図である。
【図９】テストキューブ生成過程の一例を示した第１の図である。
【図１０】テストキューブ生成過程の一例を示した第２の図である。
【図１１】本発明の実験結果を示した図である。
【図１２】半導体論理回路が市場に出荷されるまでの工程を示した模式図である。
【図１３】一般的な論理回路におけるフルスキャン順序回路の模式図である。
【符号の説明】
【０１２０】
２００　生成装置
２０４　対象テストベクトル特定部
２０６　テストベクトル集合変換部
２１６　初期テストベクトル集合

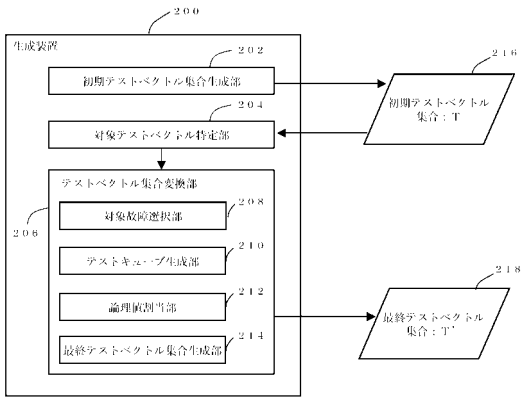
【図１】



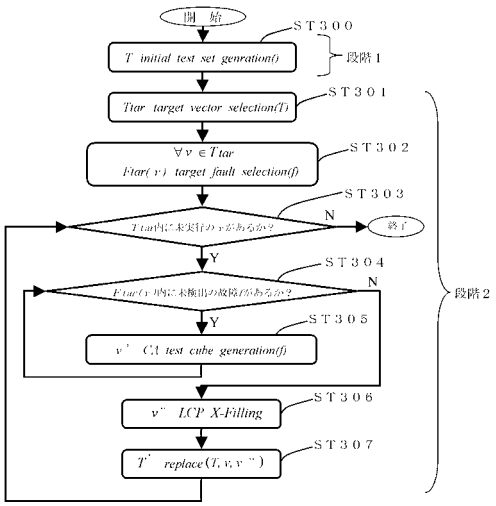
（ａ）

（ｂ）

【図２】



【図３】

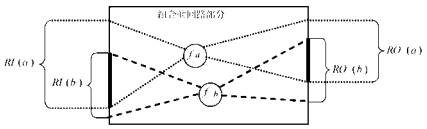


【図４】

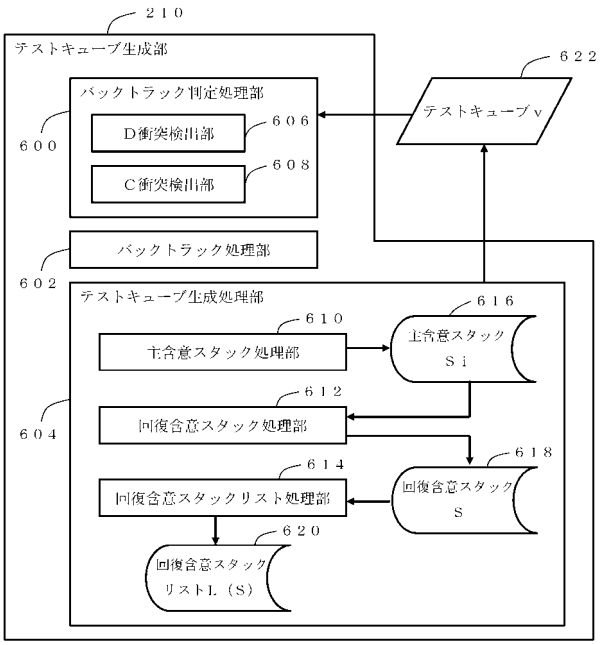
	HCP ベクトル?	故障の分類	対象故障リスト Ftar(vi)
v1	Y	(f1) (f6) [f9] [f10]	f1 f6 f9
v2	N	f2 f11	
v3	N	f3 f12	
v4	Y	(f4) (f7) (f8) [f9] f12	f4 f7 f8
v5	Y	(f5) [f10] f11	f5 f10

○：ベクトル必須故障　　□：セット必須故障

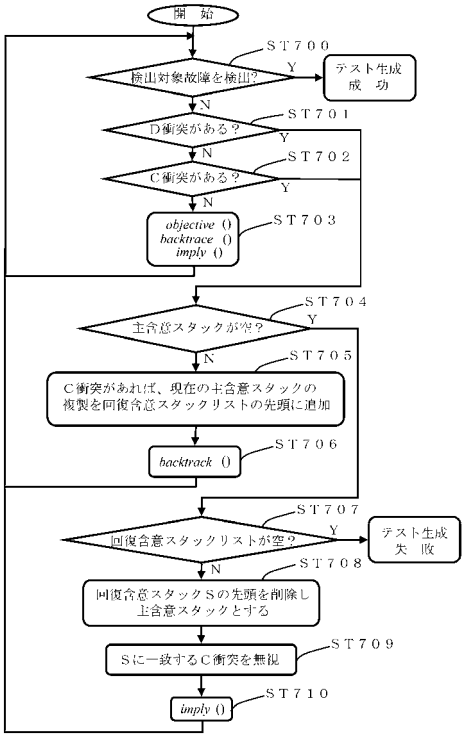
【図５】



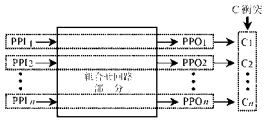
【図６】



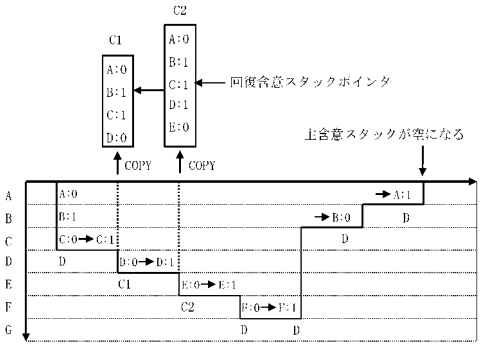
【図７】



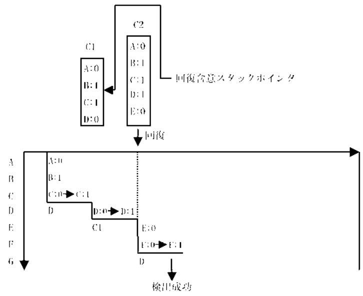
【図８】



【図９】



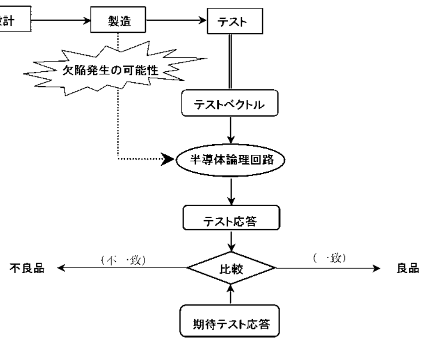
【図１０】



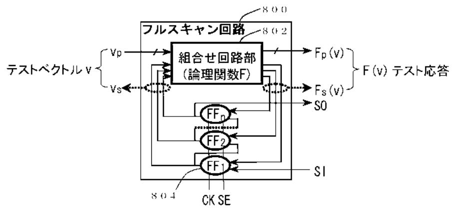
【図１１】

Circuit	Fault Cov. (%)	Original		LCP Test Generation (c limit 50% Ori. Max. Trans.)			
		# of Vec.	Max. Trans.	# of Vec.	Max. Trans.	Red. Rate (%)	CPU (Sec.)
s1238	94.9	125	18	132	12	**33.3**	0.4
s1423	99.1	24	49	38	29	**40.8**	0.6
s5378	99.1	100	102	112	86	**15.7**	1.5
s9234	93.5	111	124	158	100	**19.4**	5.0
s13207	98.5	235	380	236	287	**24.5**	16.9
s15850	96.7	97	282	140	171	**39.4**	36.2
s35932	89.9	12	1548	15	922	**40.4**	73.0
s38417	99.5	87	590	172	492	**16.6**	88.6
s38584	95.9	114	925	155	459	**50.4**	141.1

【図１２】



【図１３】



フロントページの続き

(72)発明者　皆本　義弘
福岡県北九州市八幡西区三ヶ森２－８－１８　４０３号
(72)発明者　伊達　博
福岡県福岡市早良区百道浜３－８－３３　株式会社システム・ジェイディー内

審査官　堀　圭史

(56)参考文献　特開２００４－３１７５０５（ＪＰ，Ａ）
特開２００６－０６６８２５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０１Ｒ　３１／２８－３１９３
Ｈ０１Ｌ　２１／６６，
Ｇ０６Ｆ　１１／２２－２７７，１２／１６