

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2007-263724

(P2007-263724A)

(43) 公開日 平成19年10月11日(2007.10.11)

(51) Int. Cl.	F I	テーマコード (参考)
GO 1 R 31/3183 (2006.01)	GO 1 R 31/28	2 G 1 3 2
GO 1 R 31/28 (2006.01)	GO 1 R 31/28	G

審査請求 未請求 請求項の数 19 O L (全 23 頁)

(21) 出願番号	特願2006-88695 (P2006-88695)	(71) 出願人	503360115 独立行政法人科学技術振興機構 埼玉県川口市本町4丁目1番8号
(22) 出願日	平成18年3月28日(2006.3.28)	(71) 出願人	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
		(71) 出願人	504125469 株式会社システム・ジェイディー 福岡県福岡市早良区百道浜三丁目8番33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

最終頁に続く

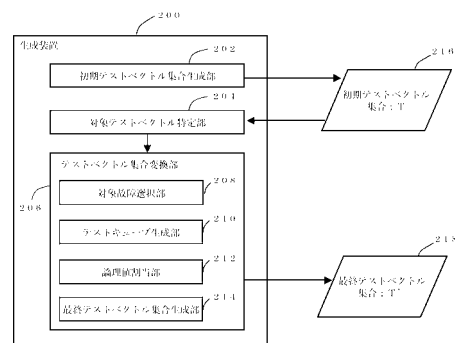
(54) 【発明の名称】 生成装置、生成方法、生成方法をコンピュータに実行させることが可能なプログラム、及び、このプログラムを記録した記録媒体

(57) 【要約】

【課題】 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等を提供する。

【解決手段】 論理回路に対する初期テストベクトル集合216を生成する生成装置200であって、前記初期テストベクトル集合216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(遷移ビット数)について所定の基準を満たして選択されるべきテストベクトルを特定する対象テストベクトル特定部204と、前記対象テストベクトル特定部204が特定した選択されるべきテストベクトルに対し、前記順序回路に含まれるスキャンセルの出力について遷移ビット数を減少させるように変換するテストベクトル集合変換部206とを備える。

【選択図】 図2



**【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た、生成装置。

10

**【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。

20

**【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、請求項 1 から 3 のいずれかに記載の生成装置。

30

**【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た、生成装置。

40

**【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

50

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、生成方法。

10

【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、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。

20

【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、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、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、請求項 7 から 10 のいずれかに記載の生成方法。

30

【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回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、生成方法。

40

【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。

50

**【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。

10

**【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、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。

20

**【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】**

30

**【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。

40

**【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

50

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。

10

【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。

20

30

【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。

40

【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

50

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。

10

#### 【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。

20

#### 【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。

30

#### 【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。

40

#### 【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

50

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-Stripping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。

10

【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。

20

【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。

30

【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

40

50

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.153-159,2002.

【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. 10

【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】 20

【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。 30

【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。 40

【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 50



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。

10

【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、テストデータ量削減、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。

20

【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。

30

【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。

40

【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】

50

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。

10

【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。

20

【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。

30

【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。

40

【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

50

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回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。

10

【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。

20

【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。

30

【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。

40

【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

50

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。

10

【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。

20

【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(P I)、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(P P I)、外部出力線(P O)、フリップフロップの入力線である擬似外部出力線(P P O)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 $\langle v : P I \rangle$ と、擬似外部入力線を介して与えられる部分 $\langle v : P P I \rangle$ からなる。 $\langle v : P P I \rangle$ 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 $\langle f(v) : P O \rangle$ と、擬似外部出力線に現れる部分 $\langle f(v) : P P O \rangle$ からなる。 $\langle f(v) : P P O \rangle$ 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。

30

【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 $\langle v : P P I \rangle$ 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 $\langle f(v) : P P O \rangle$ 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。

40

【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

50

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216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。

10

【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。

20

【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。

30

【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。

40

【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ST3

50

02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

10

20

30

40

50

証する。

【0077】

【数1】

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

【0078】

(条件2)  $F_{tar}(v)$  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。

10

【0079】

(条件3)  $F_{tar}(v)$  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  $F_{tar}(v)$  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  $T = \{v_1, v_2, v_3, v_4, v_5\}$  とし、 $v_1$ 、 $v_4$  及び  $v_5$  をHCPであるテストベクトルとする。すなわち、 $T_{tar} = \{v_1, v_4, v_5\}$  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  $f_{11}$  及び  $f_{12}$  はテストベクトル集合  $T_{tar}$  に含まれないテストベクトル  $v_2$  及び  $v_3$  によって検出されるため、故障  $f_{11}$  及び  $f_{12}$  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  $T_{tar}$  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  $TA = \{f_1, f_4 \sim f_{10}\}$  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。

20

【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  $T_{tar}$  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  $T_{tar}$  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  $T_{tar}$  のテストベクトル集合を除いたテストベクトル集合( $T - T_{tar}$ )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。

30

【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  $F_{tar}(v)$  に含まれるべきである。例えばベクトル必須故障  $f_1$  及び  $f_6$  は、 $F_{tar}(v_1)$  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  $F_{tar}(v)$  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、 $f_9$  はテストベクトル  $v_1$  及び  $v_4$  によって検出されるセット必須故障であり、この  $f_9$  は  $F_{tar}(v_1)$  または  $F_{tar}(v_4)$  のいずれかに含まれる必要がある。

40

【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  $F_{tar}(v)$  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  $f_a$  と  $f_b$  があるとする。故障  $f_a$  及び故障  $f_b$  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  $RI(b)$

50

とし、故障  $f_a$  及び故障  $f_b$  から構造的に到達可能な擬似外部出力 ( P P O s ) の集合をそれぞれ  $RO(a)$  及び  $RO(b)$  とする。故障  $f_a$  及び故障  $f_b$  から到達可能な P P I s 及び P P O s の重なり度合いを  $od(f_a, f_b)$  と表して以下のように定義する。

【 0 0 8 6 】

【 数 2 】

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

10

【 0 0 8 7 】

図 5 は、故障  $f_a$  及び故障  $f_b$  から到達可能な P P I s 及び P P O s の重なり度合いの概念を示した図である。

【 0 0 8 8 】

図 5 において、 $od(f_a, f_b)$  の値が大きくなると、故障  $f_a$  及び故障  $f_b$  から到達可能な P P I s 及び P P O s の重なり度合いが大きくなる。また、故障  $f_a$  及び故障  $f_b$  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。

【 0 0 8 9 】

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  $Ftar(v) = \{fn1, fn2, \dots, fnp\}$  であるテストベクトル  $v$  で検出可能とする。まず、 $od(f, fn1)$ 、 $od(f, fn2)$ 、 $\dots$ 、 $od(f, fnp)$  を計算し、以下の式により各故障から到達可能な P P I s 及び P P O s の重なり度合いの平均を以下のように求める。

20

【 0 0 9 0 】

【 数 3 】

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

30

【 0 0 9 1 】

テストベクトル  $vm1$ 、 $vm2$ 、 $\dots$ 、 $vm_s$  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、 $aod(f, Ftar(vm1))$ 、 $aod(f, Ftar(vm2))$ 、 $\dots$ 、 $aod(f, Ftar(vm_s))$  を計算し、各故障から到達可能な P P I s 及び P P O s の重なり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。

【 0 0 9 2 】

例えば、図 4 において  $v_1$  及び  $v_4$  の両方で検出されるセット必須故障  $f_9$  を含める故障リストを決定する際、 $Ftar(v_1)$  及び  $Ftar(v_4)$  はそれぞれ  $\{f_1, f_6\}$  及び  $\{f_4, f_7, f_8\}$  である。仮に  $aod(f_9, Ftar(v_1)) < aod(f_9, Ftar(v_4))$  であれば、セット必須故障  $f_9$  は  $Ftar(v_1)$  に含まれる。図 4 には最終的な結果を示す。

40

【 0 0 9 3 】

なお、本発明の実施の形態においては、故障  $f_a$  及び故障  $f_b$  から到達可能な P P I s 及び P P O s の重なり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。

【 0 0 9 4 】

50



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、初期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と初期含意スタック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。

10

20

30

40

50

## 【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で初期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初期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初期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初期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で初期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、初期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

10

20

30

40

50

ープ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、初期含意スタックを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 >となつた場合、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。

10

【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、初期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。

20

【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。

30

【0118】

なお、上記まででは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。

40

【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。

50

【図5】故障から到達可能な擬似外部入力（出力）線の重なり具合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。

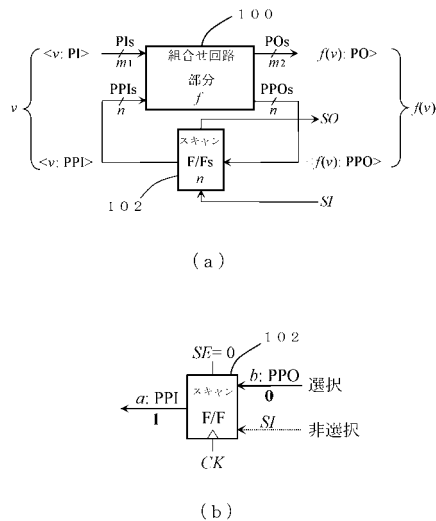
【図13】一般的な論理回路におけるフルスキャン順序回路の模式図である。

【符号の説明】

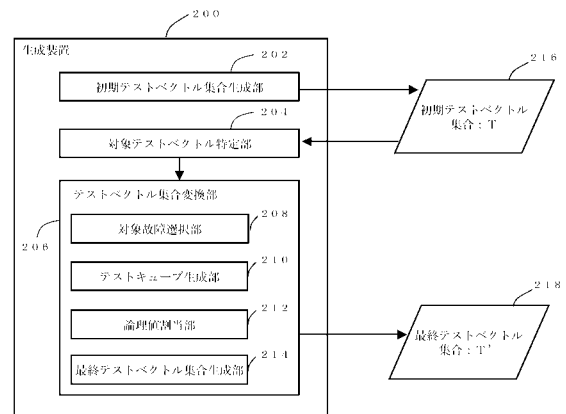
【0120】

- 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

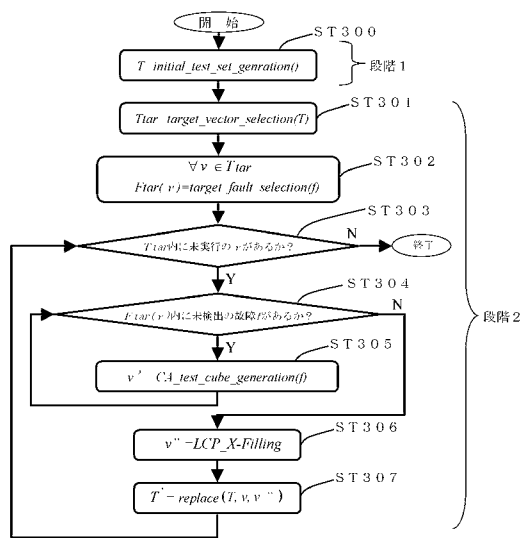
【図1】



【図2】



【 図 3 】

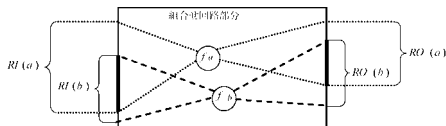


【 図 4 】

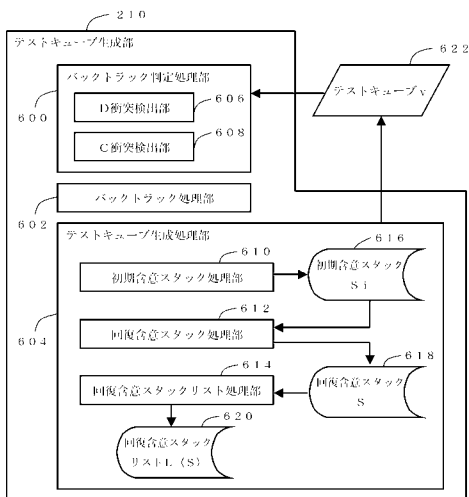
	HCP ベクトル	故障の種類	対象故障リスト Tiar(v)
v1	Y	(f1) (f6) (f9) (f10)	f1 f6 f9
v2	N	f2 f11	
v3	N	f3 f12	
v4	Y	(f4) (f7) (f8) (f9) f12	f4 f7 f8
v5	Y	(f5) (f10) f11	f5 f10

○: ベクトル必須故障 □: セット必須故障

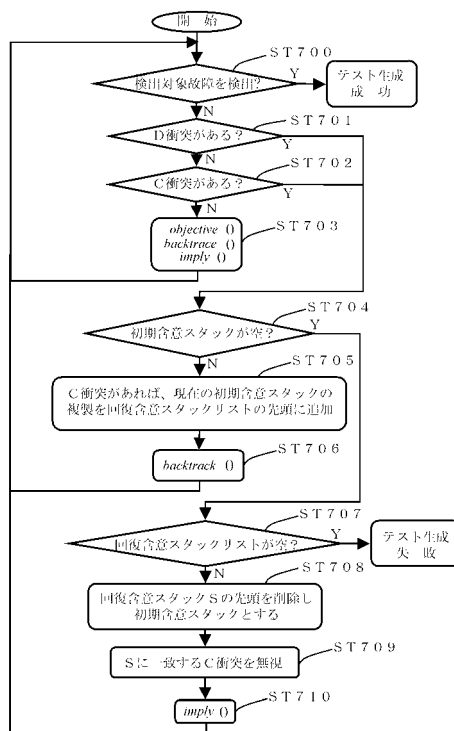
【 図 5 】



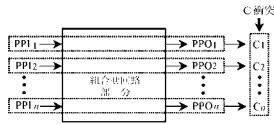
【 図 6 】



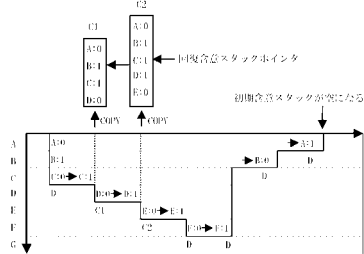
【 図 7 】



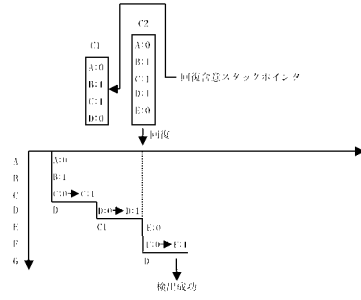
【 図 8 】



【 図 9 】



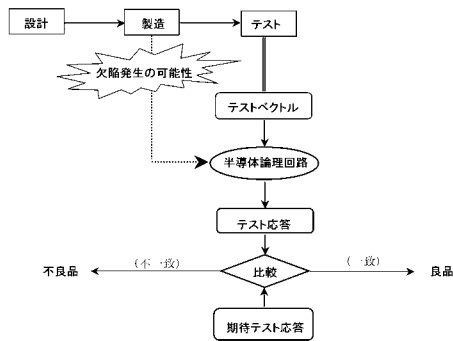
【 図 10 】



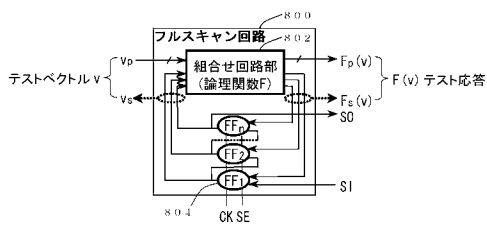
【 図 11 】

Circuit	Fault Cov. (%)	Original		ICP 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	95.0	114	925	155	450	50.4	141.1

【 図 12 】



【 図 13 】



---

フロントページの続き

- (72)発明者 梶原 誠司  
福岡県飯塚市大字川津 6 8 0 - 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 - 2 7 4 0 2 号
- (72)発明者 皆本 義弘  
福岡県北九州市八幡西区三ヶ森 2 - 8 - 1 8 4 0 3 号
- (72)発明者 伊達 博  
福岡県福岡市早良区百道浜 3 - 8 - 3 3 株式会社システム・ジェイディー内
- Fターム(参考) 2G132 AA01 AB01 AC14 AG12 AK23