

## ハードウェアテストスケジューリング情報に基づくテスト生成法

日大生産工(院) ○浅見 竜輝 日大生産工 細川 利典  
日大生産工 山崎 紘史 日大生産工 新井 雅之 京産大 吉村 正義

## 1. はじめに

近年, 超大規模集積回路 (Very Large Scale Integrated Circuits : VLSI) のテストコスト増大に伴い, テストパターン数の削減が重要視されている[1]. テストパターン数削減手法にはテスト圧縮法[2-3]やテストパターン数削減のためのテスト容易化設計手法 (Design-for-Testability : DFT) [4-5]があり, 多くの故障を並列にテストするテスト並列化手法によってテストパターン数の削減をおこなっている.

文献[4]は, テストポイント挿入とコントローラ拡大[4-5]を用いた手法である. しかしながら, この手法には2つの課題が存在する. 第1に, この手法はテストポイントとしてマルチプレクサ (Multiplexer : MUX) を挿入するため, 回路面積や遅延のオーバーヘッドが増大する可能性がある. 第2に, 演算器以外のハードウェアに対する並列テストを考慮していないため, テストパターン数が大きく削減されていない回路が存在する.

上記の課題点を解決するために, テストパターン数削減のためのコントローラ拡大[6-8]を用いたデータパス中のハードウェアである演算器とMUXの並列テストのためのDFT手法[5]が提案されている. 文献[5]ではテストポイントとしてMUXは用いず, コントローラ中の無効状態に対してハードウェアの並列テストが可能な制御信号を明に記述することによってハードウェアの並列テストを実現している. しかしながら, 実験結果では対象回路すべてのテストパターン数削減を行うことができなかった. これについて著者は使用した市販の自動テストパターン生成ツール (Automatic Test Pattern Generator : ATPG) がハードウェアテストスケジューリングの情報を考慮していない点とデータパス中のハードウェアであるMUXのテストを考慮していない点が挙げられると述べている.

本論文ではハードウェアテストスケジューリングの情報を考慮したATPGを提案する. 故障モデルは単一縮退故障モデルで行う. テスト生成はPartial MAX-SATを用いた多重目標故障テスト生成 (Multiple Target Test Generation : MTTG)[9]を用いて実現する. 本論文の構成は2章でハードウェアテストスケジューリングについて説明し, 3章でテスト生成法について説明する. 4章でまとめと今後の課題について述べる.

## 2. ハードウェアテストスケジューリング

本章ではハードウェアテストスケジューリングについて説明する. 2.1節に演算器のテストレジスタについて述べる. 2.2節にハードウェアのテスト並列化を実現するためのコントローラ拡大について述べる. 2.3節にハードウェアテストスケジューリングについて述べる.

## 2.1 演算器のテストレジスタ

本論文で対象とするレジスタ転送レベル (Register Transfer Level : RTL) 回路はデータパスとコントローラから構成されるものとする. RTL データパスの演算器  $j$  が他のレジスタを介さずに入力方向もしくは出力方向に到達可能なレジスタを演算器  $j$  のレジスタと定義する. ここで, 演算器  $j$  の入力  $i$  から入力方向に到達可能な演算器  $j$  のレジスタを演算器  $j$  の入力  $i$  の入力レジスタ, 演算器  $j$  の出力から出力方向に到達可能な演算器  $j$  のレジスタを演算器  $j$  の出力レジスタとする. また, ある制御信号における演算器  $j$  のテスト実行時に演算器  $j$  の入力  $i$  にテストパターンを印加することが可能なレジスタを演算器  $j$  の入力  $i$  の入力テストレジスタ, その出力応答を観測することが可能なレジスタを演算器  $j$  の出力テストレジスタと定義する. 演算器  $j$  の入力テストレジスタと出力テストレジスタを併せて演算器  $j$  のテストレジスタと定義する.

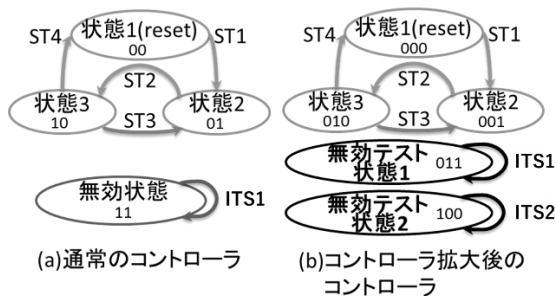
以下に演算器  $j$  のテスト可能の定義をする. 以下の2つの条件を満たす場合, 演算器  $j$  がテスト可能となる.

- 演算器  $j$  の全ての入力に対して異なるテストレジスタが割当てられている.
- 演算器  $j$  の出力に対して出力テストレジスタが少なくとも1つ以上割当てられている.

## 2.2 コントローラ拡大

通常のコントローラは無効テスト状態 (Invalid Test State : ITS)[9]である, テスト時のみ遷移が可能な状態を明に記述されていない. したがって, 無効テスト状態とその状態遷移を設計するためにコントローラ拡大をおこなう必要がある. コントローラ拡大とは, 回路のテストバリエーションを向上させるために状態や状態遷移をコントローラに追加する手法である.

コントローラには, 機能動作時には遷移しない無効状態が存在する場合がある. 本論文ではスキャンテストを対象とするため, テスト時に



においてコントローラは無効状態にも遷移可能となる。したがって、本手法ではコントローラに存在する無効状態を用いてコントローラ拡大をおこなうことで、テスト時にのみ遷移が可能な状態を ITS とする。なお、コントローラ拡大時に無効状態数が不足する場合はコントローラの状態レジスタにスキャンフリップフロップ(Flip Flop : FF)を追加してコントローラに無効状態を生成する。図 1 にコントローラ拡大の例を示す。図 1(a)に通常のコントローラ、図 1(b)にコントローラ拡大によって ITS が 2 状態追加されたコントローラの例を示す。図 1(a)のコントローラは 2 ビットの FF で構成されているため 4 つの状態が定義可能である。しかしながら、有効状態が 3 つ定義されているため ITS が 1 状態しか定義することができない。そのため、図 1(b)においてコントローラの状態レジスタにスキャン FF を 1 つ追加することで無効状態を意図的に増やし、2 状態目の ITS を設計する。また、コントローラ拡大において追加される ITS を ITS 集合とする。

演算器のハードウェアのテスト並列化を実現する ITS の出力する制御信号は，データパス中の MUX と，割当てられたテストレジスタの情報を用いて，RTL データパスから容易に求めることが可能である．

### 2.3 ハードウェアテストスケジューリング

本節ではハードウェアテストスケジューリングについて、図2のRTLデータパスと図3のハードウェアテストスケジューリングを用いて説明する。図2中の演算器の右下に書かれている数値は演算器のテストパターン数を示す。ハードウェアテストスケジューリングとは演算器を

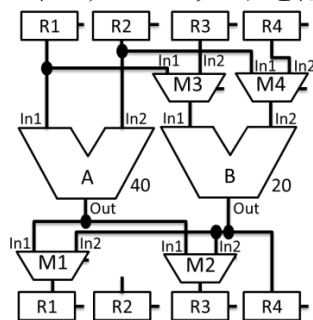


図 2. RTL データパス

テストする際に用いるテストレジスタの割当て結果である。本手法ではハードウェアテストスケジューリングを用いることで最小のテストパターン数で演算器と MUX を並列にテストすることが可能なテストレジスタ割当てを求める。

図3のハードウェアテストスケジューリングの縦軸はテストパターン数を示す。横軸であるState Transitionは状態遷移, Operational Unitは演算器の入出力, MUXはMUXの入力を表す。State Transitionの上部にテストレジスタ割当てに対応する状態遷移名を表す。演算器の各入力の上に演算器に印加するテストパターン, 演算器の出力の上に演算器の出力応答であるテスト応答を矩形で表す。テストパターンの矩形内にテストパターンを印加する入力テストレジスタ名, テスト応答の矩形内にテスト応答を格納する出力テストレジスタ名を表す。MUXの入力の上部にはテストが完了した場合, 「○」が表される。また, MUXの入力上部の矩形は状態遷移に対応している。

図4(a)にITS1におけるテストレジスタ割当ての結果を示す。演算器Aの入力In1(In2)の入力テストレジスタはR1, (R2), 出力テストレジスタはR1である。したがって、演算器Aの入出力テストレジスタが割当てられ、演算器Aがテスト可能となる。また、このテストレジスタ割当てにおいて演算器Aの20パターン分のテストと、演算器Aのテストの経路上に存在するR1, (R2), 出力テストレジスタはR3に割当てられ演算器Bのテストが可能となる。しかしながら、テストレジスタ割当てにおいて演算器AとBのテストが可能であるがレジスタR1, R2において入力テストレジスタの衝突が発生している。そのため、演算器Bをテストすると全体のテストパターン数が増加してしまうため

| テストパターン数 |                       |                  |     |     |     |     |       |     |     |     |     |     |     |     |
|----------|-----------------------|------------------|-----|-----|-----|-----|-------|-----|-----|-----|-----|-----|-----|-----|
| 60       |                       |                  |     |     |     |     |       |     |     |     |     |     |     |     |
| 50       |                       |                  |     |     |     |     |       |     |     |     |     |     |     |     |
| 40       |                       |                  |     |     |     |     |       |     |     |     |     |     |     |     |
| 30       | ITS2                  | R1               | R2  | R3  | R3  | R4  | R1,R4 | ○   | ○   |     |     |     | ○   | ○   |
| 20       |                       |                  |     |     |     |     |       |     |     |     |     |     |     |     |
| 10       | ITS1                  | R1               | R2  | R1  | R1  | R2  | R3    | ○   |     |     |     | ○   | ○   | ○   |
| 0        |                       |                  |     |     |     |     |       |     |     |     |     |     |     |     |
|          | State<br>Transi<br>on | In1              | In2 | Out | In1 | In2 | Out   | In1 | In2 | In1 | In2 | In1 | In2 | In1 |
|          |                       | A                |     |     | B   |     |       | M1  |     | M2  |     | M3  |     | M4  |
|          |                       | Operational Unit |     |     |     |     |       | MUX |     |     |     |     |     |     |

図 3.ハードウェアテストスケジューリング

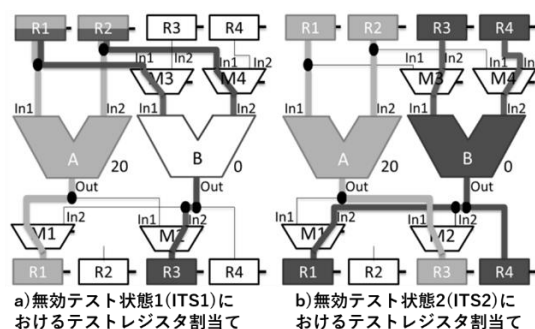


図4. テストレジスタ割り当て

ITS1 において演算器 B のテストはおこなわれない。しかしながら、演算器 B のテストレジスタを割当てることによって演算器 B のテストが可能となり、テストレジスタと演算器間の経路上の MUX のテストが可能となる。これによって、MUX M3 の入力 In1, MUX M4 の入力 In1, MUX M2 の入力 In2 のテストがおこなわれるため演算器 B のテストレジスタが割当てられている。

図 4(b)に ITS2 におけるテストレジスタ割当ての結果を示す。演算器 A の入力 In1(In2)の入力テストレジスタは R1, (R2), 出力テストレジスタは R3 に割当てられている。したがって、演算器 A の入出力テストレジスタが割当てられ、演算器 A のテスト可能となる。このテストレジスタ割当てにおいて演算器 A の 20 パターン分のテストと MUX M2 の入力 In1 のテストがおこなわれる。また、演算器 B の入力 In1(In2)の入力テストレジスタは R3, (R4)出力テストレジスタは R1, R3 に割当てられている。このテストレジスタ割当てによって、演算器 B の 20 パターン分のテストと演算器 B と MUX M3 の入力 In2, MUX M4 の入力 In2, MUX M1 の入力 In2 のテストがおこなわれる。以上のテストレジスタ割当てによってすべての演算器と MUX のテストがおこなわれる。

### 3. テスト生成法

本章ではテスト生成法について説明する。3-1 節に最大充足割当問題について述べる。3-2 節に多重目標故障テスト生成について述べる。3-3 節に Partial MAX-SAT を用いた MTTG について述べる。3-4 節でハードウェアテストスケジューリング情報に基づくテスト生成法について述べる。

#### 3.1 最大充足割当問題

最大充足割当問題 (Maximum Satisfiability Problem : MAX-SAT) とは、論理和標準形 (Conjunctive Normal Form : CNF) の命題論理式が与えられたとき、真となる節数の総和を最大化する問題である。MAX-SAT の節に重みを考慮した問題を Partial MAX-SAT と呼ぶ。

Partial MAX-SAT とは、CNF で与えられた命題論理式が真となる論理変数の組合せが存在するか否かに加えて、各 CNF の重みの総和を最大化する問題である。CNF を構成する各論理積または単項を節、必ず真とする節をハード節、偽になった際に設定した重みが発生する節をソフト節と呼ぶ。ソフト節は必ず真とする必要がなく、任意の値に重みを設定できる。ハード節は、必ず真となるため重みは無限大である。また、CNF で与えられたハード節の命題論理式が真となる論理変数の組合せが存在しない場合、命題論理式は充足不能 (Unsatisfiability : UNSAT) であ

るという。

#### 3.2 多重目標故障テスト生成

本節では、多重目標故障テスト生成 (Multiple Target Test Generation : MTTG) [10] について説明する。MTTG とは複数の目標故障を対象に実行するテスト生成手法である。ただし、複数の故障影響は同時に考慮せず、それぞれ個別に故障影響を考慮してテスト生成を行う。

MTTG では対象回路に対してテスト生成に必要な範囲特定を行う。範囲特定はテスト生成対象の故障箇所から出力方向に到達可能な範囲 (Transitive Fan-Out: TFO) で行う。故障回路は故障箇所に対する TFO とするが、MTTG では複数の故障を対象としているため、各故障に対して故障回路の範囲を特定する。この範囲特定を行うことにより、故障回路の規模を収縮することができ、テスト生成時間を短縮することができる。テスト生成モデルの正常回路はすべての故障回路の外部出力に対するトランシティブファンイン (Transitive Fan-In: TFI) となる。

故障回路に対する範囲特定後、1 つの正常回路と目標故障数分の故障回路を用意し、各故障回路の外部出力に対応する正常回路の外部出力と EXOR 演算を行う。1 つの故障回路に複数の外部出力が含まれる場合は EXOR 演算した結果に対して OR 演算を行う。ここでは、各故障回路に対応する正常回路の外部出力との EXOR 演算 (もしくは OR 演算) を各故障回路に対する故障検出回路と呼ぶ。MTTG では、すべての目標故障の同時検出が目的であるため、各故障検出回路の出力をすべて 1 とする必要がある。

目標故障数が増減することにより、故障回路数や故障検出回路数が増減する。故障回路や故障検出回路が増加することにより、ゲート数や信号線が増加し、CNF の総節数が増加する。

#### 3.3 Partial MAX-SAT を用いた MTTG

Partial MAX-SAT を用いた MTTG では、各故障検出回路の出力をソフト節とする。ソフト節 MUX M1 の入力 In1(In2) の入力テストレジスタには  $\infty$  以外の同じ重みとし、例えば 1 を与える。回路情報の CNF と縮退故障による故障値の制約節にすべて重み  $\infty$  を与え、ハード節とする。故障検出回路の各外部出力をソフト節とすることにより、Partial MAX-SAT は真となるソ

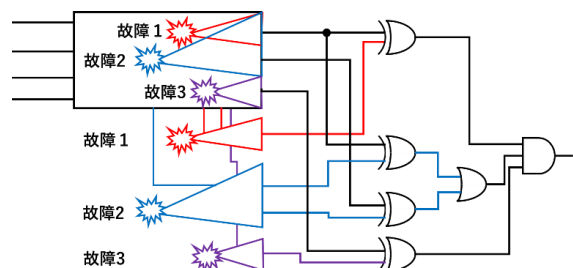


図 5. MTTG 例

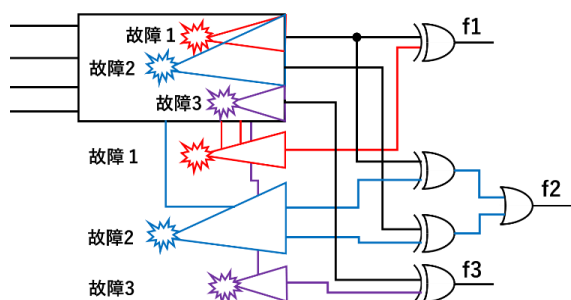


図 6.テスト生成モデル

フト節の総和の最大値を探索する．充足可能となった場合は，真となるソフト節に対応する故障が同時検出できるテストパターンを生成する．充足不能となった場合は，すべての故障が検出不能故障となる．そのため Partial MAX-SAT を用いた MTTG ではテスト生成モデルの再生成が不必要である．

Partial MAX-SAT を用いた MTTG では図 3 で示したようなテスト生成モデルを CNF に変換し，Partial MAX-SAT 判定を行うことで MTTG を実現している．各ゲートを CNF へ変換し，変換した各ゲートの CNF 式を論理積することでテスト生成モデルの変換は可能である．各故障検出回路の外部出力である  $f1$ ,  $f2$ ,  $f3$  には  $\infty$  以外の同値の重みを与え，ソフト節とすることでテスト生成モデルが生成される．

### 3-4.ハードウェアテストスケジューリング情報に基づくテスト生成法

本節ではハードウェアテストスケジューリング情報に基づくテスト生成法を説明する．テスト生成は 2-3 節で説明したハードウェアテストスケジューリング情報に基づいて行う．ハードウェアテストスケジューリング情報の時刻で選択している状態遷移でテスト可能であるハードウェアに該当する信号線の故障を未検出故障集合から選択する．選択した故障を目標故障に設定し，Partial MAX-SAT MTTG に入力する．Partial MAX-SAT MTTG で検出した故障は未検出故障集合から削除し，再度検出されないようにする．ここまでの動作，未検出故障集合が空集合になるまで行う．

図 2 で示した RTL データパスと図 3 で示したハードウェアテストスケジューリングを用いて例を示す．ハードウェアスケジューリングでは ITS1 の遷移の際に演算器 A と M1 の In1, M2 の In2, M3 の In1, M1 の In1 がテスト可能状態となっている．テストレジスタは R1, R2, R3 である．したがって演算器 A に該当する信号線の故障，M1 の In1, M2 の In2, M3 の In1, M1 の In1 に該当する信号線の故障，テストレジスタ R1, R2, R3 に該当する信号線の故障を未検出故障集合から選択し，目標故障集合とする．作成した目標故障集合を Partial MAX-SAT MTTG の

入力とする．Partial MAX-SAT MTTG で生成したテストパターンで故障シミュレーションを行い，検出故障を未検出故障集合から削除する．削除後，未検出故障集合が空集合となれば本手法は終了する．空集合とならない場合は再度，目標故障集合の作成から行う．

## 4. まとめと今後の課題

本論文ではハードウェアスケジューリング情報に基づくテスト生成法について説明した．本手法は文献[5]で市販の自動テストパターン生成ツールが考慮できなかったハードウェアテストスケジューリング情報を考慮した手法である．

今後の課題は，プログラム上で動作が可能になるような実装を行う．

## 参考文献

- [1]. Y. Sato, T. Ikeya, M. Nakao, and T. Nagumo, "A BIST Approach for Very Deep Sub-Micron (VDSM) Defects," Proc. ITC, pp. 283-291, Oct. 2000.
- [2]. S. Kajihara, I. Pomeranz, K. Kinoshita, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," IEEE TCAD Vol.14 Issue12, pp.1496-1504, Dec.1995.
- [3]. S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "On Compaction Test Sets by Addition and Removal of Test Vectors," VLSI Test Symposium, 1994 Proceedings 12th IEEE, pp.202-207, NJ, USA, Apr.1994.
- [4]. 武田 俊, 細川 利典, 山崎 紘史, 吉村 正義, 「コントローラ拡大を用いたレジスタ転送レベルにおけるテストパターン数削減のためのハードウェアのテストレジスタ割当て法」, IEICE Technical Report, 2017
- [5]. L. M. Flottes, B. Rouzeyre, L. Volpe, "A controller resynthesis based method for improving datapath testability," IEEE International Symposium on Circuits and Systems, pp.347-350, Geneva, Switzerland, May.2000.
- [6]. T. Masuda, J. Nishimaki, T. Hosokawa, and H. Fujiwara, "A Test Generation Method for Datapaths Using Easily Testable Functional Time Expansion Models and Controller Augmentation," IEEE the 24th Asian Test Symposium, pp.37-42, Mumbai, India, Nov.2015.
- [7]. 浅見竜輝, 細川利典, 山崎紘史, 新井雅之, 吉村正義「Partial MAX-SAT を用いた M バイ N テスト圧縮法」, 第 81 回 FTC 研究会, July, 2019
- [8]. G. J. Tromp, "Minimal Test Sets for Combinational Circuits," Proc. ITC, pp. 204-209, 1991.
- [9]. S. Ohtake, T. Masuzawa, and H. Fujiwara, "A non-scan approach to DFT for Controllers Achieving 100% Fault Efficiency," Journal of Electronic Testing: Theory and Applications (JETTA) Vol. 16 No.5, pp.553-566, Taipei, Taiwan, Oct. 2000