ISSN 2186-5647

P-8

ハンガリアンマッチングを用いた組込み自己テストにおける比較器のテストパターンデコーダの設計手法

日大生産工（学部） ○飛田奏　日大生産工（院） 徳田晴太　日大生産工　細川利典
京産大　吉村正義　日大生産工　新井雅之

1. まえがき

近年，半導体技術の発展に伴い，超大規模集積回路(Very Large Scale Integrated circuits: VLSI)の大規模化・複雑化・高速化が急速に進行している．それに伴いテストパターン数が増加し，テスタのメモリに格納することが困難であり，製造テストにおけるコスト増大につながっている．

テストコストを削減する手法として，組込み自己テスト方式(Built-In Self-Test: BIST) [1] が提案されている．BIST 方式は，被テスト回路(Circuit Under Test: CUT)の入力にテストパターン発生器(Test Pattern Generator: TPG)，CUT の出力に出力応答解析器(Output Response Analyzer: ORA)などのテスト用回路を VLSI に組み込むことで，自己テストを行う．

一般的に TPG としてフェーズシフタ(Phase Shifter: PS) [2] 付きの線形帰還シフトレジスタ(Linear Feedback Shift Register: LFSR)が用いられている．LFSR に初期値(シード)を与え，PS に LFSR [3] の出力を入力することで規則性が緩和された擬似ランダムパターンを生成することができる．しかしながら，擬似ランダムパターンを用いたテストでは自動テストパターン生成ツール(Automatic Test Pattern Generator: ATPG)によって生成された決定論的なテストパターンと比較すると，故障検出率が低いという課題がある．その要因のひとつとして，ランダムパターンレジスタント(Random Pattern Resistant: RPR)故障 [4] の存在が挙げられる．そのため BIST 方式において高い故障検出率を達成するためには RPR 故障の検出が重要である．

近年，レジスタ転送レベル(Register Transfer Level: RTL)でのテスト容易化設計(Design For Testability: DFT)手法が提案されている [5-6] ．ゲートレベルでの DFT [7] では論理最適化後の回路に対してゲートを追加するため，最適化されているタイミングを損失する可能性がある．よって論理合成前の抽象度の高い RTL での DFT が重要となっている．

本論文では，RTL データパスを構成するランダムパターンテストに耐性のある演算器を研究の対象とする．具体的には，比較器や除算器が対象となる．BIST において，論理回路の故障検出率を向上させるための DFT 手法として，テストポイント挿入法 [5-6, 8] や決定論的 BIST [9] がある．決定論的 BIST の 1 つの手法として，高故障検出率のテスト集合を ATPG で生成しておき，そのテスト集合をメモリに格納することが考えられる．しかしながら，VLSI 内に BIST 用のメモリ [9] を搭載することは面積オーバーヘッド増大の観点から現実的でない．本論文では，擬似ランダムパターンを決定的パターンに変換するデコーダ設計手法を提案し，対象演算器の故障検出率向上を図る．具体的には，対象演算器に対して論理合成を実行し，生成された論理回路に対して単体で ATPG を実行する．テスト時に与えられた擬似ランダムパターンを ATPG によって生成された決定的パターンに変換するようなデコーダを設計する．本手法では，事前に求めた決定的パターンと PS 付き LFSR から出力された擬似ランダムパターンから，ハンガリアンアルゴリズム [10] を用いてマッチング問題を解き，求まったマッチングを満たすようなデコーダを設計する．実験では対象となる演算器とそれに対応するデコーダをまとめて論理合成を実行し，得られた回路に対してデコーダ設計前との故障検出率と面積オーバーヘッドについて比較し，評価する．

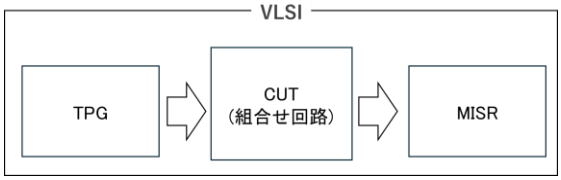

図 1. BIST 構造図

本論文は以下のように構成される．第 2 章では，BIST 方式 について説明し，第 3 章では，デコーダについて述べる．第 4 章では，提案手法の実験結果について説明し，第 5 章では，まとめと今後の課題を述べる．

2. BIST 方式

本章では，VLSI のテスト手法のひとつである BIST 方式について説明する．図 1 に BIST の構造を示す．テスト用回路として， TPG， MISR [3] が追加されている．TPG では，LFSR と PS を用いて擬似ランダムパターンを生成する．また，出力側では多入力シグネチャレジスタ(Multiple Input Signature Register: MISR)の出力を用いて出力応答系列を圧縮し，符号化を行う．

図 2 に TPG の構造を示す．LFSR [3] はオール 0 以外の初期値をシードとして与えることで擬似ランダムパターンを生成する．

PS [2] は LFSR の後段に接続され，LFSR の規則性を緩和させるために LFSR の出力同士で排他的論理和演算を行う．

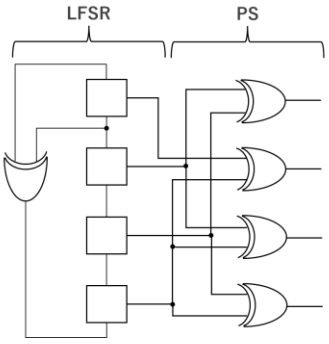

図 2. TPG 構造図

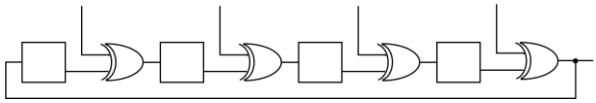

図 3. MISR 構造図

A Design Methodology for Test Pattern Decoders of Comparators in Build-in Self-Test Using Hungarian Matching
Kanaru TOBITA, Haruta TOKUTA, Toshinori HOSOKAWA, Masayoshi YOSHIMURA and Masayuki ARAI

図 3 に MISR [3] の構造を示す．MISR は，LFSR の構造をベースに複数の入力を持たせたものであり，CUT の出力応答を圧縮する役割を持つ．MISR によって圧縮された出力応答をシグネチャと呼び，シグネチャが期待値と一致するかを解析する．

3. デコーダ

本章では，本論文の提案手法である擬似ランダムパターンを決定的パターンに変換するデコーダについて説明する．3-1 節でランダムパターンテストに耐性のある演算器について説明し，3-2 節で全体のテストアーキテクチャについて説明する．3-3 節でデコーダ設計のために解く，ハンガリアンアルゴリズム [10] について説明し，最後に 3-4 節でデコーダ設計について説明する．

3-1. ランダムパターンテストに耐性のある演算器

本節では，ランダムパターンテストに耐性のある演算器について説明する．BIST 方式において，RPR 故障の存在が故障検出率低下を引き起こしていると考えられている．本論文では，BIST 環境での故障検出率が低いことで知られている比較器をランダムパターンテストに耐性のある [11] 演算器として説明する．

3-2. テストアーキテクチャ

本節では，テストアーキテクチャについて説明する．図 4 に 32 ビット比較器に対し，従来の BIST 方式を適用したときのアーキテクチャを示す．比較器左右の 32 ビット外部入力に対し，それぞれ 32 ビット出力の PS 付き LFSR が接続されている．また，外部出力では，比較器の 1 ビット出力が MISR に接続されている．

次に，図 5 で提案するテストアーキテクチャを示す．デコーダは，各 32 ビット出力の LFSR と PS から出力される擬似ランダムパターンと 7 ビットのテストパターンカウンタから出力されるパターンインデックスを入力とする．ATPG によって生成された決定的パターンをドントケア(Don't Care: X)判定をした結果である 0, 1, X で構成されるテストキューブを基に，擬似ランダムパターンを変換する．擬似ランダムパターンと決定的パターンのマッチングアルゴリズムとして，本手法ではハンガリアンアルゴリズムを用いる．マッチングを行う理由として，パターンを変換する際のビット反転数を減らすためである．

デコーダによって変換された決定的パターンは，比較器の入力となる．

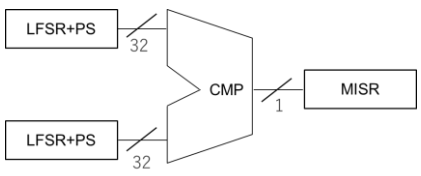

図 4. 従来の BIST 適用時のアーキテクチャ

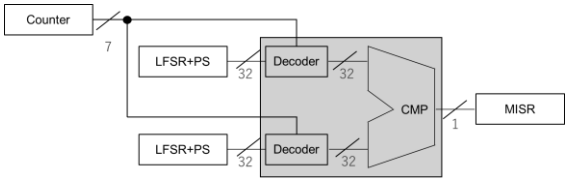

図 5. 提案テストアーキテクチャ

3-3. ハンガリアンアルゴリズム

本節では，デコーダ設計の変換アルゴリズムとして用いるハンガリアンアルゴリズム [10] について説明する．ハンガリアンアルゴリズムは，集合$X = \{x_1, x_2, \dots\}$と集合$Y = \{y_1, y_2, \dots\}$においてコストが最小になるような最大マッチングを得るアルゴリズムである．本手法では，コストを決定的パターンと擬似ランダムパターンのビット反転数とし，このビット反転数が最小となるような最大マッチングを求める．

表 1 にパターン数を 4 とし，擬似ランダムパターンの集合RPの各パターンと決定的パターンの集合DPの各パターンのビット反転数を示したコスト行列を示す．表 1 の例を用いて，ハンガリアンアルゴリズムは以下のステップで行う．

STEP1　二部グラフ作成

RPとDPの各パターン同士のビット反転数をコストとしたコスト行列から二部グラフを作成する．

STEP2　初期辺部分グラフ作成

① 各行の最小コストを重みとして，各行から引く．
② 各列の最小コストを重みとして，各列から引く．
③ コストが 0 の組み合わせで辺部分グラフを作成する．

STEP3　ラベル付け

① マッチングしていないx_iにラベル(*)を付ける．もしラベル(*)が付かないならば，アルゴリズムを終了する．
② ラベル付けされたx_iにおいて，マッチングに使われていない辺で結ばれるy_iにラベル(x_i)を付ける．
③ ラベル付けされたy_iにおいて，マッチングに使われている辺で結ばれるx_iにラベル(y_i)を付ける．
④ この STEP をマッチングしていないy_iにラベルが付くか，ラベル付けができなくなるまで繰り返す．マッチングしていないy_iにラベルが付くことを進展があるといい，その頂点を進展の頂点という．進展があったならば STEP4 へ，ラベル付けができないならば STEP5 へ遷移する．

STEP4　マッチング改善

① 進展の頂点からラベル(*)まで，ラベルで示す頂点へ進む．ここで進んだ道を交代道と呼び，交代道に属さないマッチングとマッチングに属さない交代道から，新しいマッチングを構成する．
② 改善の可否を見るために STEP3 へ遷移する．

STEP5　辺部分グラフ修正

① 二部グラフ内の現在コストが 0 でない，x_iのラベル付き頂点から始まる辺とy_iのラベルなし頂点で終わる辺の最小コスト δ を求める．
② x_iのラベル付きの重みに δ を足し，y_iのラベルなしの重みから δ を引く．また，現在コストが 0 でないx_iのラベル付き且つy_iのラベルなしのコストから δ を引き，現在コストが 0 でないx_iのラベルなし且つy_iのラベル付きのコストに δ を足す．
③ 新たにコスト 0 になった組み合わせを辺部分グラフに追加する．
④ 改善の可否を見るために STEP3 へ遷移する．

表 1. コスト行列例

	DP1	DP2	DP3	DP4
RP1	6	12	15	15
RP2	4	8	9	11
RP3	10	5	7	8
RP4	12	10	6	9

表 1 にコスト行列の例を示す. 例に対して, STEP1 で作成する二部グラフを図 6 に示す．次に図 7 に STEP2 で作成した初期辺部分グラフとコスト行列を示す．最初に各行の最小コストを求め，行毎に最小コストを引く．次に各列の最小コストを求め，列毎に最小コストを引く．STEP3 でラベル付けを行うと進展の頂点が現れ，STEP4 で図 8 に示すマッチングが作成される．STEP3 に遷移し新たにラベル付けを行った結果を図 9 に示す．進展がないため STEP5 に遷移し，辺部分グラフの修正を行う．辺部分グラフ修正に伴うコスト行列の修正を図 10 に示す．二部グラフの最小コスト δ=4 を求め，コスト行列を修正する．新たにコストが 0 になった組み合わせを辺部分グラフに追加し，再度 STEP3 に遷移する．ラベル付けを行うと，図 11 に示す進展の頂点が求められ，STEP4 で進展の頂点から新たなマッチングを求める. STEP3 で改善の可否を見るが，マッチングしていない頂点が存在しないためアルゴリズムを終了する．最終的なマッチング結果を図 12 に示す．結果として最小コストである 28 が求められる．

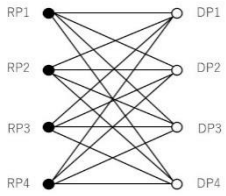

図 6. 二部グラフ

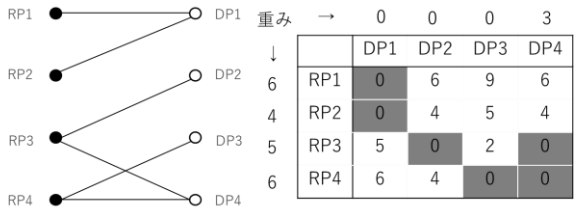

図 7. 初期辺部分グラフとコスト行列

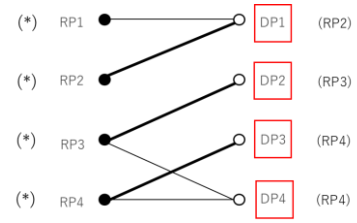

図 8. 初期マッチング

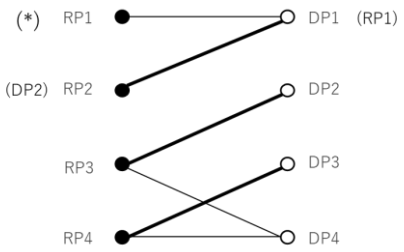

図 9. ラベル付け

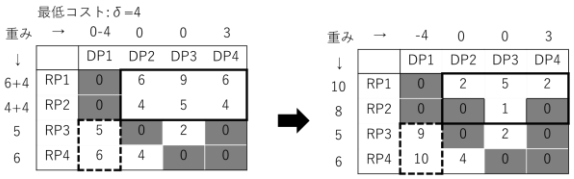

図 10. コスト行列の修正

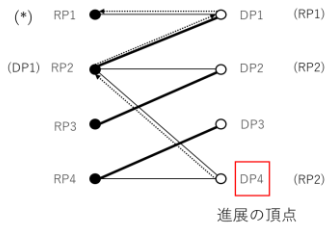

図 11. 進展の頂点

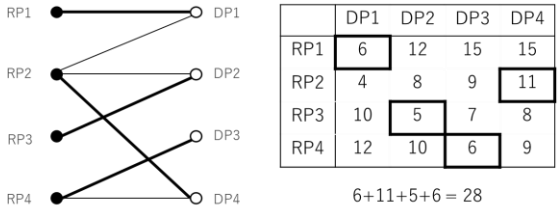

図 12. 最適マッチング

3-4. デコーダ設計

本節では，デコーダの設計について説明する．デコーダは PS 付き LFSR から入力される 64 ビットの擬似ランダムパターンとパターンカウンタから入力される 7 ビットのパターンインデックスを入力とし，決定的パターンを出力する．

デコーダにはハンガリアンアルゴリズムで求めた，擬似ランダムパターンと決定的パターンのペアを排他的論理和演算した結果であるビット反転箇所を表すパターン *CP* を格納している．

擬似ランダムパターンとパターンインデックスが入力されると，デコーダはパターンインデックスを参照し，入力されたパターンがどの決定的パターンとマッチングしているかを判別し，そのマッチングに対応する *CP* を出力する．最後に *CP* と擬似ランダムパターンで再度排他的論理和演算を行うと，X 割当てされた決定的パターンが出力され，このパターンが比較器に入力される．

4. 実験結果

本章では実験結果を説明する．ハンガリアンマッチングに用いるテストキューブは，Synopsys 社の TetraMAX を用いて ATPG を行った結果を X 判定したものを使用した．

はじめに PS 付き LFSR のみを用いて，通常の 32 ビット比較器の故障検出率を評価する．LFSR は 64 ビットに指定し，ランダムパターンの規則性を緩和させるため PS が追加されている．ランダムパターン数を 1,000，10,000，100,000 として実験を行った．故障シミュレーションは，Tetra MAX を使用した．表 2 に従来の BIST 手法を用いた比較器の故障検出率を示す．

表 2 より，32 ビット比較器はランダムパターンを 100,000 パターン印加したとしても故障検出率が 50% 以下であり，比較器はランダムパターンテストに耐性のある演算器ということがわかる．

表 2. 32 ビット比較器の故障検出率

パターン数	故障検出率(%)
87	19.00
1,000	25.57
10,000	40.27
100,000	47.17

表 3. デコーダ付き 32 ビット比較器の故障検出率

パターン数	故障検出率(%)
87	100.00

表 4. 面積の比較

回路	面積
比較器	157
デコーダ付き比較器	1,152

次に，デコーダを追加した 32 ビット比較器に対して実験を行い，故障検出率を評価する．実験条件は完全に同一とし，対称故障を比較器のみに限定して故障シミュレーションを実行した．表 3 にデコーダ付き比較器の故障検出率を示す．表 3 より，デコーダを追加した比較器では 87 パターンで故障検出率を 100%にすることができた．このパターン数は ATPG で生成された決定的パターン数と同じ数値である．

最後に面積オーバーヘッドについて評価する．表 4 に 32 ビット比較器とデコーダを追加した 32 ビット比較器の面積を示す．

表 4 より，比較器の面積が 157 に対し，デコーダ付き比較器が 1,152 となり，デコーダ追加前後で面積が約 7.3 倍増加していることが確認できる．

5. まとめと今後の課題

本論文では，ハンガリアンマッチングを用いた組込み自己テストにおける比較器のテストパターンデコーダの設計手法を提案した．

実験結果として，故障検出率を 100%にすることができた．しかしながら，面積オーバーヘッドが約 7.3 倍という結果が得られた．

今後の課題として，故障検出率を維持しつつ，低面積オーバーヘッド指向なテストアーキテクチャ設計の提案が挙げられる．

6. 参考文献

1) 藤原秀雄，"ディジタルシステムの設計とテスト"，工学図書株式会社，2004.

2) J. Rajski, N. Tamarapalli, and J. Tyszer, “Automated Synthesis of Phase shifters for Built-In Self-Test Applications,” IEEE Trans. Comput. Aided Design Int. Circuits & Syst., vol. 19, no.10, pp. 1175–1188, 2000.

3) Michael L. Bushnell and Vishwani D. Agrawal. “Essentials of Electronic Testing for Digital, Memory & Mixed -Signal VLSI Circuits,” Kluwer Academic Publisher, 2000.

4) M. Abramovici, M.A. Breuer, and A.D. Friedman, Digital Systems Testing and Testable Design. IEEE Press, 1990.

5) Hiroyuki Iwata, Yoichi Maeda, Jun Matsushima, Oussama Laouamri, Naveen Khanna, Jeff Mayer, Nilanjan Mukherjee, “A New Framework for RTL Test Points Insertion Facilitating a “Shift-Left DFT” Strategy”, 2023 IEEE International Conference (ITC)

6) Kedarnath J. Balakrishnan, Lei Fang, “RTL Test Point Insertion to Reduce Delay Test Volume”, 25th IEEE VLSI Test Symposium (VTS’07), 2007

7) Yang Sun, Spencer K. Millican, and Vishwani D. Agrawal,” Special Session: Survey of Test Point Insertion for Logic Built-in Self-test”, IEEE 38th VLSI Test Symposium (VTS), 2020

8) Chih-chang Lin, Marek-Sadowska, Kwang-Ting Chen and Mike Tien-Chien Lee, Test Point Insertion: Scan Paths through Combinational Logic, 33rd Design Automation Conference, 1996.

9) Lijian Li, Yinghua Min, “An efficient BIST design using LFSR-ROM architecture”, Asian Test Symposium (ATS), 2000

10) アラン・ドーラン，ジョーン・オールダス，大石泰彦(訳)，” よくわかるネットワークのアルゴリズム”，日本評論社，2003

11) Indradeep Ghosh, Niraj K. Jha, Sudipta Bhawmik, “A BIST Scheme for RTL Controller-Data Paths Based on Symbolic Testability Analysis”, Design and Automation Conference. 35th DAC., 1998