

# 平衡構造回路のテストパターン数推定法

日大生産工(院) ○齋藤 亮介

日大生産工 細川 利典  
広島市大 井上 智生

## 1 はじめに

大規模集積回路 (Very Large Scale Integrated circuit : VLSI) に対するテスト費用の削減は重要な課題の1つである。今日のVLSI 設計は、論理合成技術の発展により、レジスタ転送レベル(RTL)で行うことが一般的になってきた。RTLの情報を利用したデータパスのテスト生成法として階層テスト生成[2]がある。階層テスト生成は、(1)データパスを構成する各モジュールに対するゲートレベルでのテスト生成、(2)RTLデータパスの各モジュールに対して(1)で生成されたテストパターンを、外部入力からモジュールの入力に伝搬し、その出力応答をモジュールの出力から外部出力まで伝搬するための制御信号系列(テストプラン)の生成、の2つのステップからなる。

従来のゲートレベルでのテスト生成は、回路中のフリップフロップ(Flip-Flop:FF)をスキャンFFに置き換え、シフトレジスタ状に接続することで、テスト中にFFを可制御・可観測にすることができるスキャン設計が広く使われている[6], [7]。上述した階層テスト生成は、ゲートレベルでのスキャン設計による回路全体のテスト生成に比べて、テスト生成時間、テスト実行時間を短くすることができる。しかしながら、テストプラン生成は、回路の構造によっては多くの時間を要する場合がある。そこで、テストプラン生成を容易に行うための階層テスト容易化設計法[4], [5], [8]–[10]が提案されている。しかしながら、一般的にRTLデータパスには、モジュールが多数存在する。したがって、モジュールごとにテストプランを生成すると、テストプラン数が増加し、テスト実行時間が増加する。

文献[3]では、平衡構造[1]を満たす部分回路(平衡ブロック)を対象とした階層テスト生成について、テストプラン生成アルゴリズムが提案されている。平衡構造を満たすブロックは、組合せ回路用ATPGを適用可能なだけでなく、テストプラン生成も容易になると考えられる。さらに、テスト実行時間の削減も期待できる。文献[3]で提案されたアルゴリズムは、平衡ブロックとして分割済みの回路を入力としている。しかしながら、平衡ブロックの分割によってテストパターン数、テストプラン長が変化する。本論文では、平衡ブロック分割問題を最適化問題として定式化し、平衡ブロック分割を行うヒューリスティックアルゴリズムを提案し、評価する。モジュールをグループ化したときのテストパターン数の推定についても書く。

## 2 階層テスト生成

階層テスト生成は、一般的に2つのステップからなる。第1ステップでは、演算器などの構成要素を対象としたゲートレベルでのテスト生成を行う。一般的には、加算器、乗算器などの演算回路やMUXなどの組合せ回路部(モジュール)が対象となる。以下では、この第1ステップでのテスト生成対象回路を、階層の基本単位(あるいは単に階層単位)と呼ぶ。第2ステップでは、各階層単位について、生成されたテストパターンをデータパスの外部入力から階層単位の入力に伝搬し、その出力応答を階層単位の出力からデータパスの外部出力まで伝搬するための制御信号系列を生成する。この制御信号系列をテストプラン[2]という。

## 3 平衡構造に基づく階層テスト生成

本論文では、RTLのデータパス回路を対象とする。データパス部の制御信号、状態信号は、それぞれデータパス外部から直接制御、観測可能とする。図1に、対象とするRTLデータパスの例を示す。この図において、C1～C6は組合せ回路部(モジュール)を表す。これらは、演算器やマルチプレクサ(MUX)などの組合せ回路のみの接続からなり、レジスタなどの記憶素子は含まない。また、1～14などの矩形はレジスタを表す。PI1, PI2は外部入力、PO1, PO2は外部出力である。

### 3.1 平衡構造

RTLデータパスは、構造グラフ $G(V, E)$ で表すことができる。頂点 $V$ が組合せ回路部、辺 $E$ がレジスタをそれぞれ表している。例えば、図1のデータパスを表す構造グラフ $G$ は図2のようになる。構造グラフ $G$ が以下の条件を満たすとき、対応するデータパスは平衡構造である[1]。

- $G$ は無閉路である(フィードバックループを持たない)。
- 任意の頂点対 $u, v (u, v \in V)$ の間に存在する全経路のレジスタの段数が等しい。
- 任意のホールドレジスタを削除すると、 $G$ は非連結になる。

### 3.2 戦略

文献[3]で提案された方法(以後、川原法)は、階層単位を大きくすることを考えている。すなわち、複数のモジュールからなる部分回路を1つの階層単位

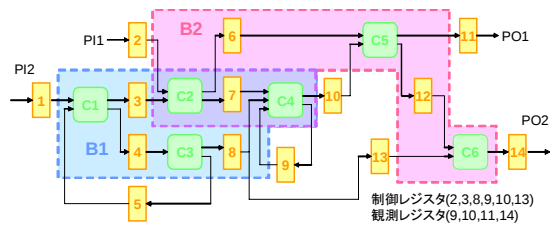


図 1. RTLデータパス

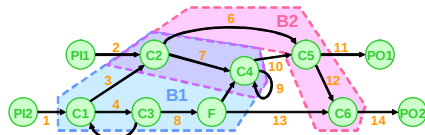


図 2. 図 1 に対する構造グラフ

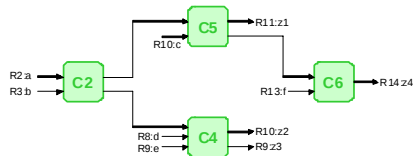


図 3. 図 1 の B2 に対する組合せ等価

| PI | PO | レジスタ |   |   |   |   |   |   |   |   |   |    |    |    |    |
|----|----|------|---|---|---|---|---|---|---|---|---|----|----|----|----|
| t  | 1  | 2    | 1 | 2 | 1 | 2 | 3 | 4 | 7 | 8 | 9 | 10 | 11 | 13 | 14 |
| -4 | c  | f    |   |   |   |   |   |   |   |   |   |    |    |    |    |
| -3 | e  | d    |   |   | f | c |   |   |   |   |   |    |    |    |    |
| -2 |    | b    |   |   | d | e |   | f | c |   |   |    |    |    |    |
| -1 | a  |      |   |   | b |   |   | d | e | f | c |    |    |    |    |
| 0  |    |      |   |   |   |   | a | b |   |   | d | e  | c  |    | f  |
| 1  |    |      |   |   |   |   | a | b |   |   | d | e  | c  |    | f  |
| 2  |    |      |   |   |   |   | a | b |   |   | d | e  | c  |    | f  |
| 3  |    |      |   |   |   |   |   |   |   |   |   |    |    |    |    |
| 4  |    |      |   |   |   |   |   |   |   |   |   |    |    |    |    |
| 5  |    |      |   |   |   |   |   |   |   |   |   |    |    |    |    |

図 4. 図 1 の B2 に対するテストプラン例

とする。階層単位を大きくすれば階層単位数は小さくなり、テストプラン数も小さくなる。その結果、テスト実行時間の削減も期待できる。

川原法では、階層の基本単位として平衡構造[1]を採用している。この基本単位を平衡ブロックと呼び、Gの部分グラフGBとして表される。平衡構造はテスト生成が容易な順序回路のクラスの一つであり、順序回路でありながら組合せテスト生成アルゴリズムが適用可能である。また、平衡構造に対するテスト実行は、1つのテストパターンを平衡構造の入力でdb時間ホールドすることでテスト実行可能(db:平衡構造の順序深度)であり、テストプラン生成が容易になると考えられる。

図1のデータパスを例として説明する。部分回路B2は平衡構造である。この平衡ブロックB2の入力に対応するレジスタは2, 3, 8, 9, 10, 13, 出力に対応するレジスタは9, 10, 11, 14である。それぞれを制御レジスタ、観測レジスタと呼ぶ。この平衡構造B2に対するテスト系列は、図3に示す組合せ等価回路C(B2)に対してテストパターンを求めることで得られる。C(B2)に対するテストパターン(a, b, c, d, e, f)と、対応する出力応答(z1, z2, z3, z4)について、平衡ブロックB2で考えると、制御レジスタ(2, 3, 8, 9, 10, 13)にそれぞれ(a, b, d, e, c, f)を設定し、B2の順序深度db = 2と同じ時間をホールドすることで、3サイクル後に観測レジスタ(9, 10, 11, 14)に(z3, z2, z1, z4)が得られる。このテスト実行を図1に示したデータパスで考えると、図4に示すようなテストプランになる。平衡構造に基づくテストプランは、外部入力か

ら平衡ブロックの制御レジスタへテストパターンを伝搬する制御フェーズ(時刻-4から-1)、平衡ブロックをテストするテストフェーズ(時刻0から2)、出力応答を平衡ブロックの観測レジスタから外部出力まで伝搬する観測フェーズ(時刻3から5)の3つからなる。

図1のデータパスにおいて、平衡構造となる部分回路は、B2のほかにB1が考えられる。これら2つの平衡ブロックを階層の基本単位としてテスト生成、テストプラン生成を行うことで、データパス全体のテスト生成が可能となる。階層単位は重複があってもよい。重複部分の回路の故障は、いずれかの階層単位の故障として検出されればよい。

## 4 平衡ブロック分割問題

平衡ブロックは、その分割の仕方によってテスト実行時間に変化が生じる。本論文では、与えられたRTL接続情報からより良いブロック分割を得るため、最適化問題を考える。以下では、文献[3]での最適化問題と本論文で提案する最適化問題を説明する。

### 4.1 テストプラン生成問題

以下に文献[3]で定式化されている最適化問題を示す。

- ・入力：RTL接続情報、ブロック情報
- ・出力：テストプラン
- ・制約：制御・観測レジスタのみホールド可能
- ・最適化：テスト実行時間

入力は回路の組合せ回路部、レジスタの接続情報が記されているRTL接続情報と、分割化済みの平衡ブロックが記されているブロック情報とする。出力は各ブロックのテストプラン、制約は制御・観測レジスタのみホールド可能とする。テスト実行時間を最小化する。その評価関数は

$$\sum (\text{平衡ブロック数} \times \text{テストパターン数} \times \text{テストプラン長})$$

とする。

### 4.2 提案する最適化問題

以下に提案する最適化問題を示す。

- ・入力：RTL接続情報、モジュール情報
- ・出力：テストプラン、ブロック情報
- ・制約：ホールド数(ただし、制御・観測レジスタのみホールド可能)。
- ・最適化：テスト実行時間

入力は、平衡ブロックのテストパターン数を推定するために必要となる各モジュールの外部出力数、ゲート数とテストパターン数が記されているモジュール情報とRTL接続情報、出力はテストプランとブロック情報である。面積の増加を防ぐためにホールド数に制約を与える。最適化と評価関数は文献[3]で定式化されている最適化問題と同じである。

### 4.3 テストパターン数推定法

テスト実行時間の評価関数で必要となる平衡ブロックのテストパターン数を推定する式を以下に示す。

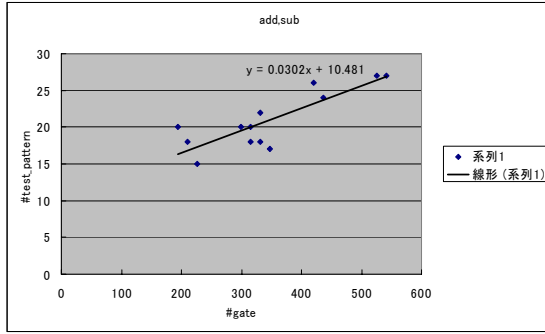


図 5. mul を含まない回路の傾向

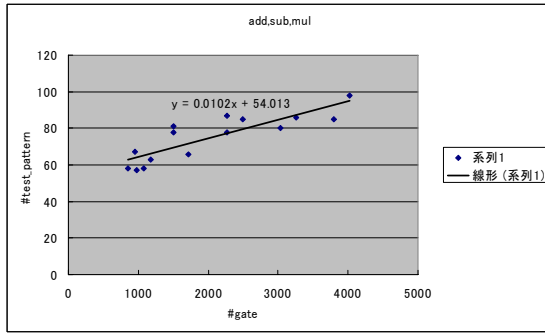


図 6. mul を含む回路の傾向

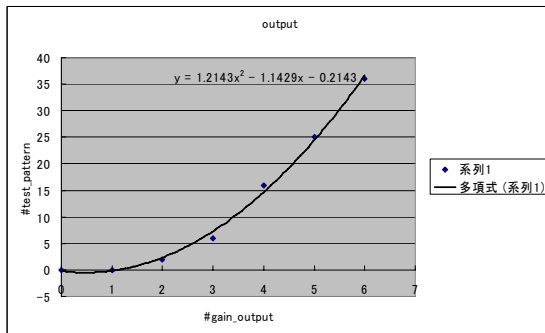


図 7. 出力数減少に伴う傾向

(Case1) モジュールmulを含まない場合

$$etp = (0.0302gate + 10.5) + (1.21dec^2 - 1.14dec - 0.214)$$

(Case2) モジュールmulを含む場合

$$etp = (0.0102gate + 54.0) + (1.21dec^2 - 1.14dec - 0.214)$$

式の変数は、etpが平衡ブロックの推定テストパターン数、gateが平衡ブロックのゲート数、decが平衡ブロックを構成するモジュールのブロック化前の出力数から、ブロック化することによって減少した出力数を表している。平衡ブロックのテストパターン数は平衡ブロックのゲート数に比例する。しかし、ブロック化することによって出力数が減るとテストパターン数は増加する。そのため、ゲート数から算出した値に、ブロック化することによって減少した出力数を考慮した値を足し合わせている。推定式を導出するのに用いたデータを図5, 6, 7に示す。図5はmulを含まない回路に対するゲート数とテストパターン数の傾向、図6はmulを含む回路に対するゲート数とテストパターン数の傾向、図7はブロック化に伴って減少した出力数とテストパターン数の傾向をそれぞれ示す。各図のx軸はゲート数、y軸はテストパターン数を表している。

## 5 平衡ブロック分割アルゴリズム

本論文では、4.2で提案した最適化問題を基に平衡ブロック分割アルゴリズムを提案する。平衡ブロック分割アルゴリズムに関する説明を以下に示す。

### 5.1 基本方針

平衡ブロック分割アルゴリズムを提案するにあたっての2つの基本方針を設ける。

- (1) 制御・観測レジスタは可能な限り、それぞれ外部入出力の近傍に決定
- (2) 制御は同一の外部入力に対する順序深度が異なるように決定し、観測レジスタは同一の外部出力に対する順序深度が異なるように決定

1つ目はホールド数の削減, 2つ目はテストプラン長の削減を目標としている。

### 5.2 ヒューリスティックアルゴリズム

図5にヒューリスティックアルゴリズムの疑似コードを示す。それぞれ変数は、RTLがRTL接続情報、moduleがモジュール情報、blockがブロック情報、tpがテストプランを意味している。

単純ブロック化は、始めに一意にブロック化を行うことを目的としている。各外部入出力からブロック化を行っていき、順序深度の制約を考慮して、可能な限りブロック同士をグループ化することによって、制御もしくは観測プランが極端に短い平衡ブロックとなる。ホールド数削減指向ブロック化は、ホールド必要箇所に対して平衡ブロックを拡大・縮小させて、値の衝突を防ぐことでホールドを削減する。テストプラン長削減指向ブロック化は、平衡ブロック同士の重なりを許容し、可能な限り平衡ブロックを外部入出力に近づけることでテストプラン長を短くする。

図6の回路ex1を用いてホールド数0, 最大順序深度2以下という制約のもとでヒューリスティックアルゴリズムの例を以下に示す。

まず、単純ブロック化を行う。外部入出力から順序深度1ずつブロック化を行っていき、入力側からのブ

```

1 algorithm(RTL, module)
2 {
3   block = 単純ブロック化(RTL);
4   tp = テストプラン生成(RTL, module, block);
5   if(制約を満たしていない)
6   {
7     while(制約を満たしていない)
8     {
9       block = ホールド数削減指向ブロック化(RTL, module, block);
10      tp = テストプラン生成(RTL, block);
11
12      block = テストプラン長削減指向ブロック化(RTL, module, block);
13      tp = テストプラン生成(RTL, block);
14    }
15  }
16  テストプラン・ブロック情報出力(tp, block);
17  return(0);
18 }

```

図 5. ヒューリスティックアルゴリズム

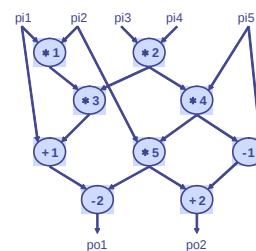


図 6. 回路ex1

表 1. 実験結果

| ベンチ<br>マーク回<br>路[16bit] | 手法          | 階層<br>単位 | 単位数 | 順序<br>深度 | テスト<br>パターン数 | テストプラン長<br>[cycle] | テスト実行時<br>間[cycle] | テストパターン<br>生成時間[s] | ホールド<br>付加数 | 故障検出効率<br>[%] |
|-------------------------|-------------|----------|-----|----------|--------------|--------------------|--------------------|--------------------|-------------|---------------|
| ex1                     | 文献[5]       | add      | 2   | 0        | 16           | 10                 | 160                | 0.04               | 0           | 100.00        |
|                         |             | sub      | 2   | 0        | 13           | 10                 | 130                | 0.04               | 0           | 100.00        |
|                         |             | mul      | 5   | 0        | 66           | 30                 | 1980               | 0.24               | 0           | 100.00        |
|                         |             | total    | 9   |          | 388          | 50                 | 2270               | 0.32               | 0           | 100.00        |
|                         | 文献[3]       | B1       | 1   | 3        | 100          | 8                  | 800                | 0.68               | 0           | 100.00        |
|                         |             | B2       | 1   | 3        | 126          | 10                 | 1260               | 1551.08            | 0           | 100.00        |
|                         |             | total    | 2   |          | 226          | 18                 | 2060               | 1551.76            | 0           | 100.00        |
|                         | 提案手法<br>制約1 | B1       | 1   | 2        | 94           | 7                  | 658                | 3.33               | 0           | 100.00        |
|                         |             | B2       | 1   | 2        | 94           | 7                  | 658                | 0.65               | 0           | 100.00        |
|                         |             | total    | 2   |          | 188          | 14                 | 1316               | 4.00               | 0           | 100.00        |
|                         | 提案手法<br>制約2 | B1       | 1   | 1        | 97           | 7                  | 679                | 3.00               | 1           | 100.00        |
|                         |             | B2       | 1   | 1        | 62           | 7                  | 434                | 0.32               | 1           | 100.00        |
|                         |             | total    | 2   |          | 159          | 14                 | 1113               | 3.32               | 2           | 100.00        |

ロック化を行っていき、入力側からのブロックと出力側のブロックが重なる直前までブロック化を続ける (B1: {\*1, \*2, \*3, \*4}, B2: {\*5, +1, +2, -1, -2}, 推定テストパターン数(以後etp):160, テストプラン長(以後tp):14, ホールド数(以後H):2). 次に、制約を満たしていないのでホールド数削減指向ブロック化を行う. B1の\*4の出力が観測の際に値の衝突が起こり、ホールドが必要になる. そこで、B1に+1を加えることで出力される時刻を異なるようにし、ホールド数を減らす (B1: {\*1, \*2, \*3, \*4, +1}, B2: {\*5, +1, +2, -1, -2}, etp:166, tp:14, H:1). ただし、+1ではなく\*5または-1をB1に加えてもホールド数を減らすことができる. そこで、最も評価関数が最小となるモジュールを加える. この例では\*5を加える (B1: {\*1, \*2, \*3, \*4, \*5}, B2: {\*5, +1, +2, -1, -2}, etp:168, tp:14, H:0). 同様に、B2に対してホールド数削減指向ブロック化を行う (B1: {\*1, \*2, \*3, \*4, \*5}, B2: {\*4, \*5, +1, +2, -1, -2}, etp:175, tp:14, H:0). 次に、順序深度削減指向ブロック化を行う. 各平衡ブロックの制御・観測プラン長が長いレジスタに隣接するモジュールを加えることでテストプラン長の削減を図れるが、この例ではホールド数が増えてしまうので、モジュールを加えない. 結果、分割された平衡ブロックは B1: {\*1, \*2, \*3, \*4, \*5}, B2: {\*4, \*5, +1, +2, -1, -2}, テスト実行時間は1316cycleとなる.

## 6 実験結果

提案手法の有効性を確認するために、実験用回路 ex1を用いて文献[3], [5]との比較実験を行った. 実験結果を表1に示す. 手法それぞれの項目は、左から階層の単位名、階層の単位数、テストパターン数、テストプラン長[cycle], テスト実行時間[cycle], テストパターン生成時間[s], ホールド付加数、故障検出効率[%]を示す. 手法は、文献[5]は各モジュールを階層単位とし、文献[3]は前もってブロック分割がされていて、提案手法制約1はホールド数0, 最大順序深度2以下, 提案手法制約2はホールド数3以下, 最大順序深度は2以下という制約のもとで、ヒューリスティックアルゴリズムを用いてブロック分割を行った. 論理合成にはSynopsys社のDesign Vision, テスト生成には同社のTetra Maxを用いた.

文献[5]と提案手法を比べると、テストパターン数が削減できていることがわかる. これは、提案手法はテストパターン生成の対象となる階層の単位を大きくすることで、1つのパターンに対して検出できる故障数が増加することに起因すると思われる. また、テストプラン長も削減できており、結果としてテスト実行時間の削減が実現できている. 文献[3]と提案手法を比べると、ヒューリスティックアルゴリズムによっ

てテストプラン長が削減できていることがわかる. 面積OHに関しては、文献[5], [3]に比べ、提案手法制約1ではテスト実行時間の削減に成功し、ホールド付加数は同等に抑えることができた. また、制約1と2の結果より、ホールド数に制約を与え少なくすればテスト実行時間が長くなり、ホールド数に制約を与えなければテスト実行時間が短くなることがわかる. 以上のことから、テスト実行時間とホールド数の増減はトレードオフの関係にあると思われる.

## 7 おわりに

本論文では、平衡ブロック分割問題を最適化問題として定式化し、平衡ブロック分割を行うヒューリスティックアルゴリズムを提案し、評価を行った. 今後の予定は、平衡構造の性質を調査し、テストパターン数推定式の精度を上げることが挙げられる.

### 「参考文献」

- [1] Rajesh. Gupta, Rajiv Gupta, and Melvin A. Breuer, "The BALLAST Methodology for Structured Partial Scan Design," IEEE Trans. Com, Vol.39, No.4, April 1990.
- [2] B.T. Murray and J.P. Hayes, "Hierarchical test generation using pre computed tests for modules," IEEE Trans. Computer-Aided Design, Integrated Circuits & Syst., vol.9, no.6, pp.594-603, June 1990.
- [3]川原侑大, 市原英行, 井上智生, "平衡構造に基づく階層テストにおけるテストプラン生成法," 信学技報 (DC2006-42), Vol. 106, No. 390, pp. 23-28, 2006年11月
- [4] I. Ghosh, A. Raghunathan, and N.K. Jha, "A design for testability technique for register-transfer level circuits using control/data flow extraction," IEEE Trans. Computer-Aided Design, vol. 17, pp. 706-723, Aug. 1998.
- [5] 和田, 増澤, K.K. Saluja, 藤原, "完全故障検出効率を保証するRTL データパスの非スキャンテスト容易化設計法," 信学論(D-I), vol.J82-D-I, no.7, pp. 843-851, July 1999.
- [6]H. Fujiwara, "Logic Testing and Design for Testability," The MIEtpess, 1985.
- [7]M. Abramovici, M. A. Breuer, and A. D. Friedman, "Digital systems testing and testable design," IEEE Transactions on Computers, vol.C-39, pp.538-544, April 1995.
- [8] S. Ravi, G. Lakshminarayana and N. K. Jha, "High-level test compaction techniques," IEEE Trans. on Computer-Aided Design, Vol.21, No.7, pp.827-841, July 2002.
- [9] 永井, 大竹, 藤原, "テスト実行時間削減のためのデータパスの強可検査性に基づくテスト容易化設計法," 信学技報 (DC2002-84), Vol. 102, No. 658, pp.31-36, Feb. 2003.
- [10] H. Ichihara, N. Okamoto, T. Inoue, T. Hosokawa, and H. Fujiwara, "An Effective Design for Hierarchical Test Generation Based on Strong Testability," Proc. IEEE Asian Test Symposium, pp. 288-293, Dec. 2005.