平衡構造を持つモジュール単位の階層テスト生成のための RTL データパスのテスト容易化設計法

日大生産工(院) ○齋藤 亮介　日大生産工 細川 利典

広島市立大 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上のテスト生成は，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，ゲートレベル上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，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，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．

平衡構造を満たす部分回路（平衡ブロック）を対象とした階層テスト生成について，川原氏が提案している[3]．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．しかし，テストプランを生成するアルゴリズムの過程で計算が複雑になっている．本論文では，川原氏が提案したテストプラン生成アルゴリズムの計算を簡略化し，同等の性能を持つテストプラン生成のヒューリスティックアルゴリズムを提案，実装し，評価する．

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論理レベルでのテストパターン生成を行う．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が対象となる．以下では，この第1 ステップでのテストパターン生成対象回路を，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 3-1 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．

A design for testability technique of RTL data path for hierarchical test generation of unit module with balanced structure

Ryosuke SAITO, Toshinori HOSOKAWA, Tomoo INOUE

3.1 平衡構造

RTLデータパスは，構造グラフGで表すことができる．頂点が組合せ回路部，辺がレジスタをそれぞれ表している．例えば，図3-1のデータパスを表す構造グラフGは図3-2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という[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(∈V)の間に存在する全経路の順序深度が等しい．
- 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 提案手法の戦略

本論文で提案する手法は，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，テストプランの数も小さくなる．その結果，総テスト実行時間の削減も期待できる．

本論文では，階層の基本単位として平衡構造[1]を採用する．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回路としてテスト生成アルゴリズムを適用可能である．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時刻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3-1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．この平衡構造B2の入力に対応するレジスタは2,3,8,9,10,13，出力レジスタは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3-3に示す組合せ等価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，平衡構造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3-1に示したデータパスで考えると，図3-4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，外部入力から平衡構造の入力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0)，平衡構造をテストするホールドフェーズ(時刻1から2)，出力応答を平衡構造の出力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3-1のデータパスにおいて，平衡構造となる部分回路は，

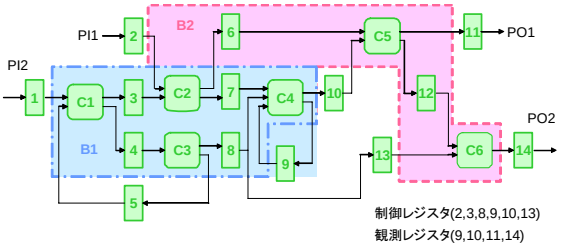

図 3-1. RTL データパス

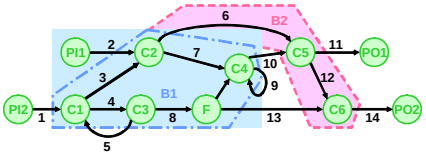

図3-2. 図3-1に対する構造グラフ

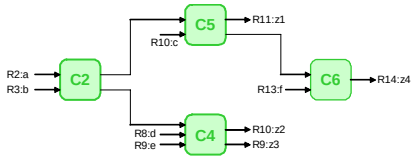

図3-3. 図3-1のB2における組合せ等価

	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			z1	z4								z3	z2	z1		z4
4			z2										z3	z2		
5			z3											z3		

図3-4. 図3-1のB2に対するテストプラン実行例

B2のほかにB1が考えられる．これら2つの平衡構造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．

4. テストプラン生成アルゴリズム

本論文では，与えられた平衡ブロック集合GBに対する総テスト実行時間を削減するために，各平衡ブロックGBiに対するテストプラン長TBiの削減を考える．GBに対するテストプランは，制御プラン集合と観測プラン集合の和集合である．制御プラン集合は，外部入力からGBの制御レジスタまで値の伝搬を行うための制御系列集合，観測プラン集合は，GBの観測レジスタから外部出力まで値の伝搬を行うための制御系列集合とする．

以下ではプラン長を最小化することを目的とした平衡ブロックの制御プラン集合生成アルゴリズムについて説明する．なお，観測プラン集合の生成アルゴリズムは，制御プラン集合生成アルゴリズムと類似したアルゴリズムで生成可能なので割愛する．

4.1 制御プラン集合生成アルゴリズム

本論文で提案するヒューリスティックアルゴリズムは，平衡ブロックGBi に対する制御プランの長さを小さくするために，他の制御レジスタの制御プランのことも考えながら探索を行う．探索は，制御レジスタから外部入力側へ行う．アルゴリズムの方針を以下に示す．

- 制御レジスタから外部入力側に探索する．
- テストプラン長が短くなるように制御プランを選択する．
- 同時刻，同ノードに異なる値を設定することはできないので，他の制御プランと競合する可能性が高い経路を避けるための指標として，混雑度を求める．
- 探索の前に一意に決められる制御プランは前もって決定することによって，探索範囲を小さくする．

以上を考慮した制御プラン集合生成アルゴリズムを以下に示す．

(1) 外部入力から制御レジスタの各ノード(外部入力，組合せ回路部)に対して，そのノードから制御レジスタに到達可能な数を表す混雑度を計算する．

(2) 外部入力の混雑度が1の経路は一意に制御プランと決定する．一意に決められる制御プランが複数存在する場合，時間的移動度が低い(制御プラン長が長い)ものから決定する．制御プランが決定したことによって，他の制御プランとして利用することが不可能になった組合せ回路部を調べ，以後選ばないようにする．

(3) 制御レジスタから探索するにあたって，一意に決められる制御プランを決定する．一意に決められる制御プランが複数存在する場合，時間的移動度が低い(制御プラン長が長い)ものから決定する．制御プランが決定したことによって，他の制御プランとして利用することが不可能になった組合せ回路部を調べ，以後選ばないようにする．

(4) 制御プランが未決定な制御レジスタから入力側に向かって幅優先で探索を行う．分岐まではそれぞれ一意に仮決定し，分岐では順序深度が短い経路を優先，順序深度が等しい経路が複数存在する場合は混雑度が低い経路を選択する．外部入力まで辿った時点で制御プランと決定する．

(5) 利用したい制御プランがすでに他の制御プランとして利用されている場合，バックトラックを行い他の制御プランを探す．バックトラックを尽くしても制御プランが見つからなければ，探索を打ち切る．

(6) すべての制御レジスタの制御プランが決定すれば終了する．それ以外は，制御プラン未決定の制御レジスタのホールド回数を1だけ増やして，(1)に戻る．

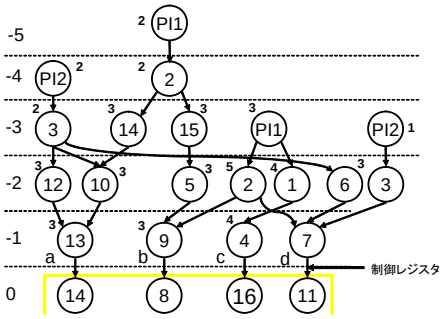

図4-1. 制御プラン集合生成アルゴリズム適用例

4.2 適用例

図4−1を用いて制御プラン集合生成アルゴリズムの適用例を示す．左に記入されている0〜−4の数字は時刻を表し，ノードと辺に関しては図3−2の構造グラフと等しいとする．レジスタa,b,c,dは制御レジスタを表し，それより下は平衡ブロックとする．

まず，アルゴリズムにしたがって各ノードに対して混雑度を計算する．図4-1では，ノードの傍に数字で記載されている．混雑度は外部入力側のノードから計算され，各ノードから制御レジスタに到達可能な数を表している．時刻-4のPI2から到達可能な制御レジスタはaとdなので，混雑度は2となる．制御レジスタに向かって分岐がない場合は，到達可能な制御レジスタの数は変わらないので混雑度もそのままに(組合せ回路部3も混雑度は2)，分岐がある場合は，それまでの混雑度と，新しく計算する分岐先の混雑度を合計した値となる．組合せ回路部3からの分岐ノードである組合せ回路部12から到達可能な制御レジスタはaだけなので，組合せ回路部3までの混雑度と足し合わせて3となる．上位ノードの混雑度を下位ノードに足し合わせることによって，制御レジスタから外部入力に向かって探索する際に，分岐点でどちらが他の制御プランと競合する可能性が高いかを判断することができる．

時刻-3のPI2の混雑度は1となっているので，制御レジスタdの制御プランを一意に決定する．その結果，(PI2, 3, 7)が制御レジスタdの制御プランとなる．また，この制御プランを決定したことによって，制御レジスタに到達するには必ず組合せ回路部7を通らなければならない組合せ回路部6は他の制御プランで利用不可とする．

制御レジスタから外部入力に向かって探索するにあたって，一意に決められる制御プランがあるか調べる．図4-1では制御レジスタcが該当するので，制御レジスタcの制御プランは(PI1, 1, 4)と決定する．また，この制御プランを決定したことによって，時刻-2の組合せ回路部2は必ず時刻-3のPI1から値を伝搬しなければならないので，組合せ回路部2は他の制御プランで利用不可とする．組合せ回路部2が利

用不可になったため, 制御レジスタbも制御プランを一意に決定できるようになる．したがって，(PI1, 2, 15, 5, 9)が制御レジスタbの制御プランと決定し，組合せ回路部10が他の制御プランで利用不可とする．

制御プランが決定していない制御レジスタaが存在するので, 制御レジスタaから外部入力に向かって幅優先探索を行う．まず，分岐があるまで探索するので時刻-1まで仮決定する．分岐では順序深度が短い経路を優先，順序深度が等しい経路が複数存在する場合は混雑度が低い経路を選択するが．この場合どちらも同じ順序深度，混雑度なのでどちらを選んでも構わないものとする．仮に組合せ回路部12を選んだとすると，あとは外部入力まで分岐がないので，(PI, 3, 12, 13)が制御レジスタaの制御プランとなる．

もし，この時点で制御プランが決まっていない制御レジスタがあれば, そのレジスタに対してホールド回数を1増やして，再度アルゴリズムを実行する．

5. 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（16bit）を用いて文献[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]テストパターン生成時間[s]，テスト実行時間[cycle]，故障検出効率[%]を示す．なお，muxはマルチプレクサ数，addは加算器数，multは乗算器数，subは減算器数，B1～B2は平衡ブロックを示す．階層単位について，文献[5]ではモジュール，提案手法では平衡ブロックを示す．また，論理合成にはSynopsys社のDesign Compiler, テスト生成にはTetra Maxを用いた．テスト実行時間の計算は，文献[5]については，各モジュールのテストパターン数に，そのモジュールのテストプラン長を乗算したものの総和で求めた．提案手法に関しては平衡ブロックGBiごとに，テストプラン長にテストパターン数を乗算したものの総和で得られる．

提案手法は従来法と比較し，階層単位が小さくなっている．また，提案手法によって生成されるテストパターン数は，文献[5]に比べて小さいことがわかる．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．

テストパターン生成時間を比較すると, 文献[5]に対して, 提案手法は増加の傾向を示している．これはテスト生成の単位を大きくしたことが原因であると考えられる．また，文献[5]は，モジュール単位でテスト生成を行うので，同じ型のモジュールに対するテストパターンを共有できることが理由として考えられる．

表1. 実験結果

ベンチマーク回路[16bit]	手法	階層単位	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]	故障検出効率[%]
ex1	和田法	mux	13	12	72	864	0.03	100.00
		add	4	22	21	462	0.03	100.00
		mult	2	66	12	792	0.27	100.00
		total	**19**	**376**	**105**	**2118**	**0.33**	**100.00**
	提案手法	B1	1	88	13	1144	0.70	100.00
		B2	1	51	11	561	0.17	100.00
		total	**2**	**139**	**24**	**1705**	**0.87**	**100.00**
ex2	和田法	mux	12	12	67	804	0.03	100.00
		add	3	22	15	330	0.03	100.00
		mult	2	66	13	858	0.27	100.00
		sub	1	20	5	100	0.03	100.00
		total	**18**	**362**	**100**	**2092**	**0.36**	**100.00**
	提案手法	B1	1	107	12	1284	0.80	100.00
		B2	1	31	11	341	0.24	100.00
		total	**2**	**138**	**23**	**1625**	**1.04**	**100.00**
4thIIR	和田法	mux	3	12	24	288	0.03	100.00
		add	2	22	17	374	0.03	100.00
		mult	3	66	21	1386	0.27	100.00
		total	**8**	**278**	**62**	**2048**	**0.33**	**100.00**
	提案手法	B1	1	103	16	1648	0.89	100.00
		total	**1**	**103**	**16**	**1648**	**0.89**	**100.00**

テスト実行時間について，提案手法は，全ての回路において大きく削減できることがわかる．これは上述したとおり，階層の単位数を削減したことによってテストプラン数を削減したこと，さらに本論文で提案したアルゴリズムによってテストプラン長を効果的に削減したことが，原因であると考えられる．

6. まとめと今後の課題

本論文において，提案手法は効果的にテスト実行時間を削減可能であることを示した．今後の課題として，テスト実行時間のさらなる削減を実現するために，提案したアルゴリズムがより効果的に機能する平衡ブロックの性質の考察が考えられる．

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]川原佑大, 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, "完全故障検出容易性を保証する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 MIT Press, 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.