社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－112　（６）
2003／11／28

強可検査性に基づくデータパスのテストプラン生成アルゴリズムの改良について

岡本直己† 市原英行†† 井上智生†† 細川利典‡ 藤原秀雄*

† 広島市立大学大学院情報科学研究科　〒731-3194 広島市安佐南区大塚東
†† 広島市立大学情報科学部　〒731-3194 広島市安佐南区大塚東
‡ 日本大学生産工学部　〒275-8575 千葉県習志野市泉町
* 奈良先端科学技術大学院大学情報科学研究科　〒630-0192 けいはんな学研都市
E-mail: †naoki@dsgn.im.hiroshima-cu.ac.jp, ††{ichihara,tomoo}@im.hiroshima-cu.ac.jp,
‡t7hosoka@cit.nihon-u.ac.jp, *fujiwara@is.aist-nara.ac.jp

あらまし　大規模集積回路に対するテスト生成を効率よく行う方法として，階層テスト生成法 [2] がある．本研究では，階層テスト容易化設計法である，強可検査性に基づくレジスタ転送レベルデータパスのテスト容易化設計法 [3] の改良について考察する．従来法 [3] を構成する手続きの１つである制御林生成アルゴリズムに着目し，生成される制御経路のタイミング衝突（１つの外部入力から，１つのモジュールの異なる２つの入力までの制御経路の順序深度が等しいこと）の発生を回避するヒューリスティックアルゴリズムを提案する．この結果，ホールド機能を付加するレジスタ数を削減することができる．さらに，モジュールの入力に接続されたレジスタの情報をタイミング衝突の尺度で表現することで，提案するアルゴリズムが，従来法 [3] の制約条件を満たさないデータパスに対しても，有効なテストプランを生成可能であることを示す．実験により，提案するヒューリスティックアルゴリズムが，追加するホールド機能（ハードウエアオーバヘッド）およびテスト実行時間を削減できることを示す．

キーワード　階層テスト生成，強可検査性，データパス，テストプラン

An Improvement of the Test Plan Generation Algorithm for Strongly Testable Datapaths

Naoki OKAMOTO†, Hideyuki ICHIHARA††, Tomoo INOUE††,
Toshinori HOSOKAWA‡, and Hideo FUJIWARA*

† Graduate School of Information Sciences, Hiroshima City University　Hiroshima, 731-3194
†† Faculty of Information Sciences, Hiroshima City University　Hiroshima, 731-3194
‡ College of Industrial Technology, Nihon University　Izumicho, Narashino, Chiba, 275-8575
* Graduate School of Info. Science, Nara Institute of Science and Technology　Kansai Science City, 630-0192
E-mail: †naoki@dsgn.im.hiroshima-cu.ac.jp, ††{ichihara,tomoo}@im.hiroshima-cu.ac.jp,
‡t7hosoka@cit.nihon-u.ac.jp, *fujiwara@is.aist-nara.ac.jp

Abstract　Hierarchical test generation [2] is an efficient method of test generation for VLSI circuits. In this work, we study an improvement of the DFT method [3] based on strong testability of Register-Transfer level (RTL) datapaths. We focus on the algorithm, which is a part of the DFT/test plan generation algorithm [3], for generating a controlling forest in a given RTL datapath, we propose a heuristic algorithm for finding a controlling forest without time conflict. As a result, it can reduce the number of registers with hold operation. Furthermore, we show that our proposed algorithm can be applied to datapaths that do not satisfy the constraint for the previous method [3], by expressing the information about input registers of a module as a measure of time conflict. Experimental results show that the proposed algorithm is effective in reducing additional hold functions(or hardware overhead), as well as test application time.

Key words　Hierarchical test generation, strong testability, datapath, test plan.

1. はじめに

今日の大規模集積回路 (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．テスト生成に要する費用を削減するためのテスト容易化設計法として，完全スキャン設計 [1] が広く用いられている．完全スキャン設計では，被テスト回路を組合せ回路としてテスト生成を行うため，テスト生成が容易になるが，得られたテスト集合を被テスト回路に入力するためには，スキャンの操作が必要となり，結果として，膨大なテスト実行時間が必要となることが問題となっている．一方，論理合成技術の発展により，今日の VLSI 設計は，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．レジスタ転送レベルのデータパス部に対するテスト生成法として，**階層テスト生成** [2] がある．階層テスト生成法は，大きく２つの手続きからなる：

(1) データパスを構成する各モジュール（論理回路）に対するテスト生成，

(2) 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，データパスの外部入力からモジュールの入力へ伝達し，その応答結果を外部出力で観測するための，データパスに対する制御系列 (**テストプラン**) の生成．

上述のように，階層テスト生成では，データパスを構成する比較的小さいモジュールごとにテスト生成を行うため，テスト生成に要する時間を小さくできる．また，データパスに対するテストプラン生成は，対象とする回路要素数が，ゲートレベルのテスト生成に比べてずっと小さいので，比較的小さい計算時間で得られることが期待される．さらに階層テストでは，テストパターンの伝達に既存のデータパスを利用するため，完全スキャン設計に対するテスト実行に比べて，ずっと小さい時間でテスト実行を行うことが出来る．

しかしながら，テストプラン生成問題は，組合せ回路のテスト生成問題と同様に探索問題の１つであり，回路の構造によっては多くの時間を要する場合がある．そのため，テストプラン生成を容易に行うためのテスト容易化設計法や，テスト容易化高位合成法が提案されている [3]- [6].

文献 [3] は，テストプラン生成が容易に行える RTL データパスの性質として**強可検査性**を示している．強可検査性を満たすデータパスに対するテストプラン生成は，テスト対象のモジュールが必要とするテストパターンやその出力応答に独立して行うことが出来る．さらに文献 [3] では，強可検査性に基づくデータパスのテストプラン生成法，および，テスト容易化設計法を提案しており，その時間計算量は $O(n^2)$ (n はデータパス中のモジュール数) と高速である．しかし，文献 [3] のテストプラン生成法，テスト容易化設計法（以下，従来法）は，その時間計算量の削減が中心に考えられているため，結果として，付加するハードウエアオーバーヘッドが増加する傾向が強い．また，従来法が適用可能なデータパスに対する制約条件が厳しいため，実用的な回路に対して従来法を適用するためには，あらかじめ制約条件を満たすように設計を変更する必要があり，さらなるハードウエアオーバーヘッドが必要となることが多い．

本研究では，従来法，すなわち，文献 [3] で提案されている，

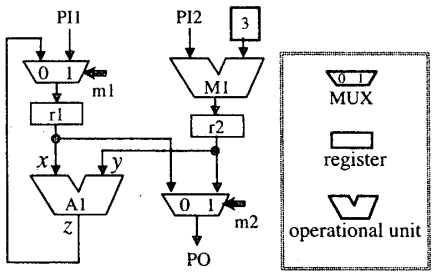

図 1　階層テスト生成

強可検査性に基づくデータパスのテストプラン生成法，および，テスト容易化設計法の改良について考察する．従来法を構成する手続きの１つである**制御林生成アルゴリズム**に着目し，生成される制御経路の**タイミング衝突** (１つの外部入力から，１つのモジュールの異なる２つの入力までの制御経路の順序深度が等しいこと) の発生を回避するヒューリスティックアルゴリズムを提案する．この結果，**ホールド機能** を付加するレジスタ数を削減することができる．さらに，レジスタを持たない再収斂経路を重みの大きな衝突として表現することで，提案するアルゴリズムが，従来法 [3] の制約条件を満たさないデータパスに対しても，有効なテストプランを生成可能であることを示す．

実験により，提案するヒューリスティックアルゴリズムが，追加するホールド機能 (ハードウエアオーバヘッド) およびテスト実行時間を削減できることを示す．

2. 背　　景

2.1 階層テスト生成

階層テスト生成 [2] は，大きく２つのステップからなる．第１のステップでは，データパスを構成する各モジュールに対して，テスト生成を行う．ここでは，モジュールは演算器やマルチプレクサなどの組合せ回路とする．第２のステップでは，第１ステップで生成されたテストパターンを用いて，データパス中のそれぞれのモジュールに対するテストプラン（テストパターンを外部入力からモジュールの入力に伝達し，その出力応答をモジュールの出力から外部出力まで伝達するための制御信号の時系列）を生成する．図１を例に，モジュール $A1$ のテストを行うためのテストプランについて考える．まず，$A1$ 単体に対してテスト生成を行い，テストパターン x, y，および出力応答 z を求める．次に，モジュール $A1$ の２つの入力に，それぞれ，求めた値 x, y を入力するためには，２つのレジスタ $r1, r2$ にそれらの値を設定する必要がある．x を $r1$ に入力するためには，１つ前の時刻に x を $PI1$ に入力しなければならない．この時 $m1$ に対しては，その右入力からの値をスルーさせるための制御信号が必要となる．また，y を $r2$ に入力するためには，x を $PI1$ に入力するのと同時刻に $y/3$ を $PI2$ に入力しなければならない ($M1$ は定数入力を持つ乗算器)．一方，x, y の入力に対する出力 z はレジスタ $r1$ に取り込まれ，次時刻に外部出力 PO で観測される．この時，$m1, m2$ それぞれに対してその左入力からの値をスルーさせるための制御信号が必要となる．以上をまとめたテストプランを表１（空欄はドントケア）に示す．

表 1　モジュール $A1$ のテストプラン

時刻	$PI1$	$PI2$	PO	$r1$	$r2$	$m1$	$m2$
1	x	$y/3$				1	
2				x	y	0	
3			z	z			0

2.2　強可検査性に基づくテスト容易化設計

上述のように，階層テスト生成では，モジュールに対して生成されたテストパターン，及び，出力応答の伝達のための制御系列（テストプラン）がレジスタ転送レベルデータパスで計算される．しかし，回路が大規模，複雑になるとテストプランの生成が困難になる．文献 [3] では，ステップ 1 で生成されるテストパターンに依存せずに，効率よくテストプランを生成できるデータパスの性質として，強可検査性を定義している．

[強可検査性]

データパス中のすべてのモジュールに対して以下の 2 点を満たすとき，そのデータパスは強可検査であるという．

- **強可制御性**：モジュールの入力端子に対して外部入力から任意の値を伝達可能
- **強可観測性**：モジュールの出力端子のとりうる任意の値を外部出力まで伝達可能

文献 [3] ではこの定義に基づき，与えられたデータパスが強可検査性を満たすためのテスト容易化設計法，並びにテストプラン生成を行うアルゴリズムを提案している（以下，従来法と記す）．そのアルゴリズムは，主に次の 3 つのステップからなる．

1　**制御林の生成**：外部入力から各モジュールのすべての入力までの制御経路を決定．
2　**観測林の生成**：各モジュールの出力から外部入力までの観測経路を決定．
3　**DFT 回路の決定**：得られた制御経路，観測経路を実現するための DFT 要素の追加．

DFT 要素には，

(1) レジスタに対する**ホールド機能**：任意の時間，値を保持可能にする，

(2) 演算モジュールなどに対する**スルー機能**：他の入力に依存せず，入力を出力へ通過（スルー）させる，

の 2 つがある．図 2 に強可検査 DFT 後のデータパスの例を示す．DFT によって，レジスタ $r2$ に対してホールド機能 (hold) が，モジュール $A1$ の左入力および，$A2$ の右入力に対してスルー機能（thru）が付加されている（図 2 では，スルー機能はマスク素子を挿入することで実現している）．

従来法における階層テスト容易化設計アルゴリズムは，各モジュールごとにテストプランを生成するのではなく，すべてのモジュールに対して共通の制御経路，観測経路を生成するため，計算量を小さくすることができる．また，以下に示すように，順序深度に基づく幅優先探索で制御林の生成を行い，バックトラックなしで制御経路，観測経路，DFT 決定を行うので，階層テスト生成のための DFT およびテストプラン生成を効率よく行うことができる．

2.3　制御林生成アルゴリズム

本論文の考察対象は，制御林生成のアルゴリズムであるため，

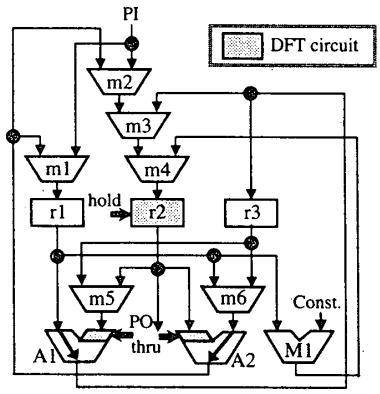

図 2　強可検査 DFT データパス

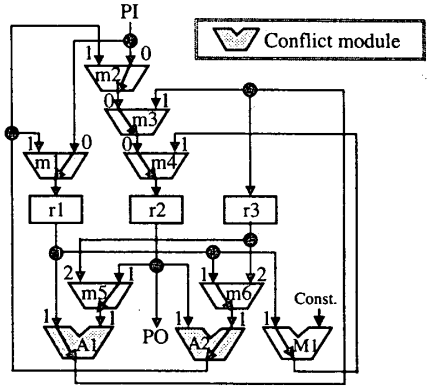

図 3　制御林生成後のデータパス

ここでは，従来法における制御林生成アルゴリズムの概略を説明する．制御林生成アルゴリズムは，外部入力を始点として，すべてのモジュールのすべての入力までの制御経路を決定する（結果として，制御経路からなる木の集合，すなわち，林になる）．演算モジュールの入力から出力までの経路は，スルー機能の付加によって実現する．制御林の生成は，順序深度に基づく幅優先探索によって行われる（幅優先探索は，各モジュールまでの制御経路の順序深度を小さくするためのヒューリスティック．結果として，テストプラン長が小さくなることを期待している）．探索の単位は，レジスタから，(そのレジスタから）到達可能な次のレジスタまでとなる．

図 3 に，制御林生成後のデータパスの例を示す．図 3 において，モジュールの各入力に示された数字は，外部入力 PI からの順序深度を示し，外部入力 PI からモジュールの入力までに，テストパターンを伝達するのに要する時間に対応する．もし，制御林生成の結果，あるモジュールへの 2 つの入力までの制御経路が同じ外部入力から到達し，かつ，同じ順序深度であれば，その 2 つの入力に異なるテストパターンを入力できない（図 3 の $A1$ と $A2$）．このように，制御経路が，1 つのモジュールの 2 つ（以上の）入力に，同じ外部入力から同じ順序深度で到達するとき，その経路は **衝突** しているという．このような場合，上述のアルゴリズムのステップ 3 において，2 つの制御経路のいずれかにある 1 つのレジスタにホールド機能が追加される．これにより，同じ外部入力からでも，時刻をずらして 2 つのテストパターンを入力可能にする．

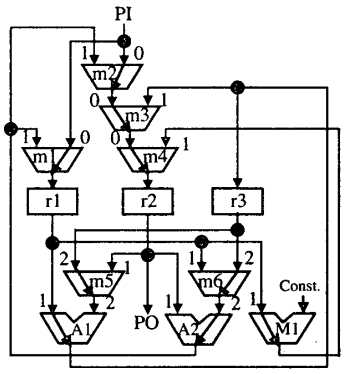

図4 タイミング衝突のない制御林

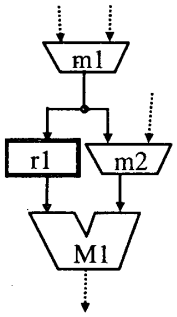

図5 再収れん経路上の独立レジスタ

図3に示す制御林では，2つのモジュール $A1, A2$ の入力において，タイミング衝突が起こっている．これを解決するために，レジスタ $r2$ にホールド機能が追加される．

2.4 従来法の問題点

上述のように，従来法の階層テスト容易化設計／テストプラン生成アルゴリズムは，強可検査性に基づくため，モジュールのテストパターンに依存せずテストプラン生成が可能で，階層テスト生成にとって効果的であるといえる．しかしながら，結果的に負荷されるDFTオーバーヘッドは，必ずしも最適とは言えない．例えば，図4に示すように制御林が生成できればタイミング衝突が起こらず，ホールド機能の追加が必要ない．

従来法の制御林生成アルゴリズムでは，順序深度に基づく幅優先探索を行って制御林を決定しているため，本質的にタイミング衝突を起こす可能性が高く，必要以上にタイミング調整のためのホールド機能を追加する場合が多いと考えられる．

また，従来法には，解を保証する（全てのモジュールに対するテストプランを必ず生成する）ための2つの制約条件がある．

[制約条件]

C1:モジュールの入力数　データパスを構成するモジュールは，2入力または1入力とする．

C2:再収れん経路の独立レジスタ　データパス中に存在する再収れん経路には，いずれか一方に経路にのみ存在するレジスタがある（図5において，$m1$ から $M1$ に至る経路のうち，左の経路にのみ $r1$ がある）．

従来法は，これら2つの制約を満たしていないデータパスに対しても適用可能ではある．しかし，上述の通り，制御経路の生成が順序深度に基づく単純な幅優先探索を基本としているために，たとえ従来法を適用して得られた解がテストプランとして正しくても，必要となる追加ホールド機能の数は増加傾向にある．また，テストプランとして正しくない制御経路を生成する可能性も高いと思われる．

3. タイミング衝突を回避するヒューリスティックアルゴリズム

ここでは，従来のアルゴリズムの高速性を生かしながら，タイミング衝突を回避することのできる制御林生成アルゴリズムを考察する．提案するヒューリスティックアルゴリズムの基本方針は以下の通りである．

- 強可検査性に基づく階層テスト生成／テスト容易化設計（モジュールのテストパターンに独立してテストプランを生成）．
- バックトラックのない，順序深度に基づく幅優先探索を原則とする（従来法と同じ．高速化のため）．
- モジュールの入力から出力へ通過する経路を決定する際に，衝突をチェックする（従来法の場合，無条件で入力から出力への経路を決める）．
  - 出力先のモジュールの入力でタイミング衝突が起こらない場合，入力から出力への経路を決定（スルーの決定）．
  - タイミング衝突が発生するとき，決定を延期．他の入力からの経路を待つ．
  - いずれの入力からの経路も，出力先でタイミング衝突が起こるときは，最小の衝突数となる経路を選択．

3.1 ヒューリスティックアルゴリズム

提案する制御林生成アルゴリズムを示す．

[制御林生成アルゴリズム]

1 各外部入力に接続するモジュールの入力をキュー que_reg, que_wire に追加．（2つのキューの違いは後述．）

2 que_wire から入力 e を取り出す．$que\_wire = \phi$ ならば que_reg から．どちらのキューも空ならば，ステップ4へ．

3 e を入力とするモジュールを v とし，v の出力先モジュール $u1, u2, ...$ の入力での衝突をチェック（未決定入力は ドントケア 扱い）．以下の判定を行った後ステップ2に戻る．

(a) 衝突がないとき：入力 e から v をスルーし，出力先モジュール $u1$,$u2$,... への経路を決定．モジュール $u1, u2, ...$ の，v の出力に接続された入力をキュー que_reg，または，que_wire のいずれかに追加．

(b) 衝突があるとき：

i モジュール v の全ての入力で衝突が起こったとき：最小の衝突数となる入力を e とし，ステップ3-(a)と同様の手続きを行う．

ii それ以外の時：モジュール v を別のキュー $que\_suspend$ に追加．

4 キュー $que\_suspend = \phi$ ならば，終了．そうでなければ，$que\_suspend$ よりモジュール v を取り出す．もし，v のスルーが未決定ならば，その時点での最小衝突数となる入力を e として，ステップ3の(a)と同様の手続きを実行後，ステップ2へ．　□

このアルゴリズムでは，順序深度に基づく幅優先のために2つのキュー que_reg, que_wire を用いる．探索は，モジュールの入力（または外部入力）から，そのモジュールの出力先にあるモジュールの入力（または外部出力）まで経路を単位として行う．その経路上にレジスタがあるときは，que_reg に，レジスタがない場合は que_wire にその経路の終端であるモジュールの入力を追加する．

アルゴリズムを図4を用いて説明する．外部入力 PI から到達可能なすべてのモジュールは $m1$ と $m2$ の2つであり，どちらの経路もレジスタを含まないため，初期値として各モジュールの入力がキュー que_wire に追加される（ステップ1）．キューの処理は，レジスタを通過しない経路，すなわち，que_wire から行うため，キュー que_wire からモジュール $m1$ の右入力が取り出さ

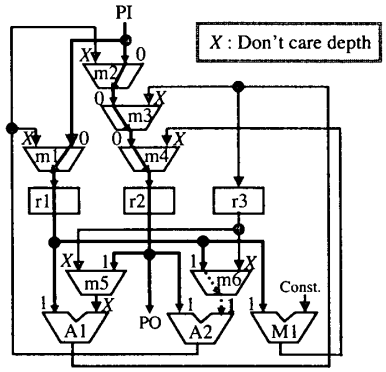

図 6　タイミング衝突が起こる場合

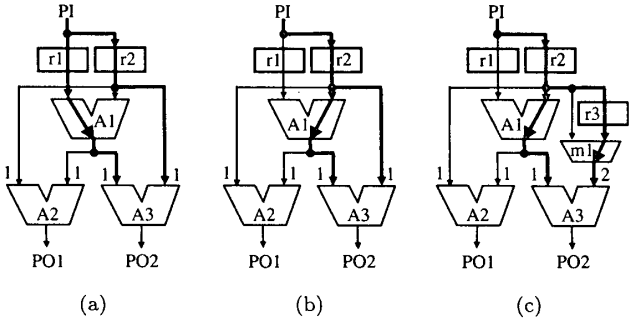

図 7　A3 に対するテスト可能経路とテスト不能経路 (a) A1 左からのスルー (テスト可能). (b) A1 右からのスルー (テスト不能). (c) バイパスレジスタの挿入 (テスト可能)

れる（ステップ 2）．次に，この $m1$ の右入力から到達可能なモジュールを調べ，そのモジュールの入力でタイミング衝突があるかどうかを調べ，経路を決定するかどうかを決める（ステップ 3）．この例では，到達可能な 3 つのモジュール $A1$，$m6$，$M1$ でタイミング衝突がないため，$m1$ からこれらのモジュールへの経路を決定し，さらに $A1$ の左入力，$m6$ の左入力，$M1$ の左入力をキュー *que_reg* に追加する（ステップ 3 の (a)）．

以上のように処理を進めると，タイミング衝突が発生する場合がある．図 6 は，ステップ 3 において $m6$ の左入力を e としてタイミング衝突をチェックしている図であるが，このとき $A2$ の入力で衝突が発生するため，モジュール $m6$ をキュー *que_suspend* に追加する（ステップ 3 の (b)）．なお，キュー *que_wire* と *que_reg* の両キューが空になると，キュー *que_suspend* からモジュールが取り出され，そのモジュールのスルーが未決定であれば，そこから到達可能なモジュールの入力をキュー *que_wire* と *que_reg* に追加する．(ステップ 4)．(この例では，処理を進めても，キュー *que_suspend* から取出したモジュールのスルーが未決定になる場合は生じない.) 以下，処理を進めると図 4 の制御林が得られる．

3.2　非独立レジスタ経路の回避

2 節で述べたように，一般的なデータパスには，独立したレジスタがない再収れん経路（非独立レジスタ経路）も存在する．このような制約条件 (C2) :「再収れん経路の独立レジスタ」を満たさないデータパスであっても，タイミング衝突しているモジュールの各入力の直前に独立したレジスタが存在すれば，階層テストは実行可能となる．ここでは，そのような一般的なデータパスに対して，提案する制御林生成アルゴリズムがテストプランとして正しい制御経路を生成すること（なるべく非独立レジスタ経路を回避すること）を考える．

図 7 は，上述の条件 C2 を満たさないデータパスであるが，外部入力 PI からモジュール $A3$ への再収斂経路においてモジュール $A1$ の左右どちらの入力から経路決定すべきかを考慮している状態を表している．ここでは，図 7(a), (b) のどちらにおいても出力先モジュール $A2, A3$ の入力において，タイミング衝突が発生する（衝突数はどちらも 2）．図 7(a) の場合，外部入力 PI からモジュール $A3$ の左側入力までの経路において，モジュール $A3$ の右側入力までの経路にはない独立したレジスタ $r1$ が存在する．一方，同図 (b) の場合では，モジュール $A3$ の両入力に対する経路において独立したレジスタが存在しない．つまり，非独立レジスタ経路を構成しており，$A3$ に対してはテストプランが存在しないことになる．もし，モジュール $A3$ に対するテストをこの制御経路を用いて行うのであれば，追加 DFT オーバヘッドとして，同図 (c) のように一方の経路にバイパスレジスタを挿入することが必要となる．以上より，この例では同図 (a) のような経路を選択することが望ましい．このような経路を選択するために，提案手法では，非独立レジスタ経路を構成する場合（図 7(b)）の衝突数を十分に大きな値（例えばデータパス中のモジュール数）として扱っている．こうすることにより，上記の例で図 7(a) の経路が選択されるようになる．なお，この手続きはステップ 3-(b)-i における，最小衝突数の入力からの経路を選択するところで行う．

4.　実験結果

提案した制御林生成アルゴリズムの有効性を確認するために，ベンチマーク回路（データパス）に対する計算機実験を行った．実験では，従来法で DFT を行った結果と，提案アルゴリズムを用いて制御林を生成した後，従来法による観測林生成，DFT 要素の追加，テストプラン生成を行った結果を比較した．実験に用いた 8 つのベンチマーク回路の特性を表 2 に示す．左から順に，それぞれの回路の外部入力数，外部出力数，マルチプレクサ数，演算器数，レジスタ数である．なお，8 つの回路のうち Paulin, Mpeg 以外は 2.4 節で述べた制約条件 (C2) :「再収れん経路の独立レジスタ」を満たさないデータパスであるため，従来法では対象にできないデータパスである．しかし，今回は従来法との比較のためにあえて実験の対象回路とした．

表 3 に実験結果を示す．表 3 において，prev. および ours の欄は，それぞれ従来法，提案法によって得られた結果を示している．#conflicts における ctrl. および obsv. の欄は，それぞれ生成した制御林と観測林中に存在するタイミングが衝突するモ

表 2　ベンチマーク回路

	#PIs	#POs	#MUXs	#ops.	#regs.
Paulin	2	2	11	4	7
Mpeg	7	16	207	161	241
Lwf.1	1	1	6	3	3
Lwf.2	2	2	5	3	5
Jwf.1	1	1	25	3	14
Jwf.2	5	5	25	3	14
Risc	1	3	92	16	40
DCT-F	4	7	26	40	11

表 3　タイミング衝突数と追加ホールド機能数

		#conflicts		#holds			#un-	plan
		ctrl.	obsv.	ctrl.	obsv.	tot.	tests	len.
Paulin	prev.	2	0	2	0	2	0	68
	ours	**1**	0	**1**	0	**1**	0	**67**
Mpeg	prev.	0	0	0	0	0	0	11,343
	ours	0	0	0	0	0	0	**11,005**
Lwf.1	prev.	2	**1**	1	**1**	**1**	0	**33**
	ours	**0**	2	**0**	2	2	0	34
Lwf.2	prev.	1	1	1	1	1	0	27
	ours	1	1	1	1	1	0	27
Jwf.1	prev.	13	10	8	2	8	0	148
	ours	**5**	**2**	**5**	**2**	**7**	0	**140**
Jwf.2	prev.	4	0	4	0	4	0	92
	ours	**0**	0	**0**	0	**0**	0	92
Risc	prev.	83	32	44(7)	25	44	7	346
	ours	**67**	**0**	**36**(2)	**0**	**36**	**2**	**315**
DCT-F	prev.	4	0	3(2)	0	2	2	166
	ours	**2**	0	**2**(2)	0	2	2	**164**

ジュール数を示している．#holds における ctrl. および obsv. の欄は，それぞれ前述のタイミング衝突を回避するために追加ホールド機能が必要となるレジスタ数を示しており，#untests 解消のためのバイパスレジスタ数 (括弧で示した数字) を含めたものである．tot. は回路全体で必要なホールドレジスタ数を意味する．(制御林，観測林は別々に生成するため同じレジスタをホールドすることがある．したがって，tot. は必ずしも ctrl. と obsv. の和にならない.) また，#untests の欄は，生成された制御経路ではテスト不能なモジュールの数を示している．さらに，Plan len. の欄には生成した制御林，観測林から得られるテストプランの長さを示している．なお，上述のとおり，Paulin, Mpeg 以外は制約条件 (C2):「再収れん経路の独立レジスタ」を満たさないデータパスであるが，Lwf.1, Lwf.2, Jwf.1, Jwf.2 の 4 つの回路に対しては従来法と提案法のどちらでもテストプランを生成することができた（#untests が 0）．

表 3 より，提案するアルゴリズムが，Lwf.2, Mpeg を除くすべての回路において，タイミング衝突数 (ctrl.) をできた．これにともない，ホールド機能を追加する必要のあるレジスタ数 (ctrl.) も減少していることがわかる．また，観測林およびテストプランの生成はともに従来法を適用した結果であるが，Jwf.1, Risc では，観測林におけるタイミング衝突数を大幅に削減しており，Risc については，観測林における追加ホールド機能が必要なレジスタ数も減少している．これより，タイミング衝突の少ない制御経路を生成することは，値の観測におけるタイミング衝突数の削減にも効果的であることがいえる．なお，すべての回路において，最終的に必要となるホールドレジスタ数が増えたものはない．

#untests の欄に示されるように，制約条件 (C2) を満たしていない Risc, DCT-F では，従来法，提案法ともに適切な制御経路を選択できないモジュールがあった．しかしながら，Risc におけるテスト不能モジュールの数は，従来法よりも提案法のほうが少ない．これより，提案するヒューリスティックアルゴリズムは非独立レジスタ経路の回避という点においても効果的であると考えられる．テスト不能モジュールの数は，生成した制御経路を用いてのテストを可能とするために必要となるバイパスレジスタの数に相当するため，その削減はハードウエアオーバヘッドの削減につながる．

また，Paulin, Jwf.1, Risc, Mpeg, DCT-F ではテストプラン長も削減しているが，これは，提案手法によりタイミング衝突数を削減できたことで，値の伝達に用いるにホールド機能数が減少したことが影響していると思われる．以上より，提案法は有効なヒューリスティックアルゴリズムであると考えられる．

5. まとめと今後の課題

本研究では，強可検査性に基づくデータパスのテストプラン生成法，および，テスト容易化設計法の改良について考察した．従来法を構成する手続きの 1 つである制御林生成アルゴリズムに着目し，生成される制御経路のタイミング衝突の発生を回避するヒューリスティックアルゴリズムを提案した．さらに，独立したレジスタをもたない再収斂経路を構成するモジュールをタイミング衝突数の大きなモジュールとして表現することで，提案するアルゴリズムが，従来法の制約条件を満たさないデータパスに対しても，実験に用いた多くの回路で有効なテストプランを生成可能であることを示した．実験結果は，提案するヒューリスティックアルゴリズムが効果的にタイミング衝突および，テストプラン実行のために必要なホールド機能数を削減できることを示し，さらに，テスト実行時間の削減も可能であることを示している．本研究では，従来法の一部である制御林生成アルゴリズムの改良のみを行ったが，観測林生成アルゴリズムを合わせて考察することで，より効果的なテストプラン生成／テスト容易化設計が行えると思われる．また，テストコントローラ（テストプラン生成回路）を含めたハードウエアオーバーヘッドの評価も今後の課題の 1 つである．

謝　辞

本研究に際し，多くの貴重なご意見を頂いた半導体理工学研究センターの宮崎政英氏，および，奈良先端科学技術大学院大学情報科学研究科の井上美智子助教授，大竹哲史助手，米田友和助手を始めとするコンピュータ設計学講座の皆さん，並びに広島市立大学情報科学部の橘啓八郎教授，および，松浦義則助手を始めとする設計工学講座の皆さんに感謝します．また，実験プログラムの作成，並びにデータ収集に関して，多くのご協力を頂いたシステム・ジェイディーの平尾智也氏，松尾茂則氏，浜本征志氏，山田幸英氏に深く感謝します．なお，本研究は一部，広島市立大学特定研究費（一般研究費）の研究助成による．

文　献

[1] M. Abramovici, M.A. Breuer and A.D. Friedman: Digital Systems Testing and Testable Design, IEEE Press, 1990.
[2] B.T. Murray and J.P. Hayes, "Hierarchical test generation using pre computed tests for modules," IEEE Trans. Comput.-Aided Des., Integrated Circuits & Syst., vol.9, no.6, pp.594-603, June 1990.
[3] 和田弘樹，増澤利光，K.K. Saluja，藤原秀雄，"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.
[4] S. Bhatia and N.K. Jha, "Genesis: A behavioral synthesis system for hierarchical testability," Proc. European Design and Test Conference, pp. 272-276, 1994.
[5] I. Ghosh, A. Raghunath and N.K. Jha, "Design for hierarchical testability of RTL circuit obtained by behavioral synthesis," Proc. Int. Conf. on Computer Design, pp.173-179, 1995.
[6] I. Ghosh, A. Raghunath and N.K. Jha, "A design for testability technique for RTL circuits using control/dataflow extraction," Proc. Int. Conf. on CAD, pp.329-336, 1996.