論　文

レジスタ転送レベルデータパスの単一制御並行可検査性に基づく組込み自己テスト法

山口　賢一[†]　　和田　弘樹[††]　　増澤　利光[†††]　　藤原　秀雄[†]

A BIST Based on Concurrent Single-Control Testability of RTL Data Paths

Ken-ichi YAMAGUCHI[†], Hiroki WADA[††], Toshimitsu MASUZAWA[†††], and Hideo FUJIWARA[†]

あらまし　レジスタ転送レベルデータパスの組込み自己テスト法として，単一制御可検査性に基づく方法が提案されている [2]．この手法では，小さいハードウェアオーバヘッドで 100%近い故障検出率が得られるが，組合せ回路要素を一つずつテストするためテスト実行時間が大きい．そこで本論文では，複数の組合せ回路要素を同時にテスト（並行テスト）できるように，単一制御可検査性を拡張した単一制御並行可検査性を提案し，この可検査性に基づく組込み自己テスト方式を提案する．ベンチマーク回路を用いた実験の結果により，提案手法は単一制御可検査性に基づく手法に比べて，ハードウェアオーバヘッドがほとんど増加することなく，テスト実行時間を縮小できることを示す．

キーワード　テスト容易化設計，レジスタ転送レベルデータパス，組込み自己テスト，単一制御可検査性，階層テスト

1. まえがき

VLSI の大規模化，複雑化に伴い，論理回路のテストとして組込み自己テスト法（Built-In Self-Test．以下，BIST）が重要視されている．BIST を実現するために，テスト対象回路の外部入力（Primary Input. PI），外部出力（Primary Output. PO）に，それぞれ，テストパターン生成器（Test Pattern Generator. TPG），応答解析器（Response Analyzer. RA）を付加する．しかし，テスト対象回路に閉路が含まれている場合には，PI と PO に TPG，RA を付加するだけでは高い故障検出率を得ることができない．そのため，高い故障検出率を得るために，回路内部にテストのためのハードウェアを付加する方法が数多く提案されている [1]．このように，回路の設計をテスト容易となるように変更することをテスト容易化設計（Design For Testability. 以下，DFT）と呼ぶ．

BIST は，test per scan 方式と test per clock 方式に分類できる．test per scan 方式では，回路中の（一部の）レジスタをスキャンレジスタに変更し，スキャン操作により，TPG で生成したテスト系列をスキャンレジスタにシフトインし，スキャンレジスタに格納された応答を RA にシフトアウトする．test per scan 方式では，スキャン操作によりテスト系列を 1 ビットずつシフトインするので，連続したシステムクロックでテスト系列を印加できず，テスト実行時間も長い．

一方，test per clock 方式では，回路中の（一部の）レジスタを TPG，RA に変更する．このようなテストレジスタとしては，BILBO（Built-In Logic Block Observer）[3]，CBILBO（Concurrent BILBO）が用いられる．test per clock 方式の BIST として，Wunderlich ら [4] は，回路中のすべての閉路が少なくとも二つの BILBO か一つの CBILBO を含むようにするテスト容易化設計法を提案している．この手法では，内部レジスタを含む多くのレジスタを BILBO や CBILBO に設計変更する必要があるため，ハードウェアオーバヘッドが大きくなる．

[†] 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information of Science, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma-shi, 630–0101 Japan
[††] 株式会社日立製作所中央研究所システム LSI 研究部，国分寺市
Central Research Laboratory, Hitachi, Ltd, 1-280 Higashi-Koinokubo, Kokubunji-shi, 185–8601 Japan
[†††] 大阪大学大学院基礎工学研究科，豊中市
Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka-shi, 560–8531 Japan

そこで，井筒ら [2] は，単一制御可検査性というテスト容易性を提案し，test per clock 方式の BIST として，単一制御可検査性に基づく方法とそのテスト容易化設計法を提案している．この手法では，内部レジスタを BILBO や CBILBO に変更せず，TPG，RA は，それぞれテスト対象回路の PI，PO のみに付加する．そして，階層テストに基づいてデータパス中の組合せ回路要素（演算器，マルチプレクサなど）ごとにテストを行う．つまり，テスト系列を TPG から各組合せ回路要素までデータパス内の経路を通って伝搬させ，応答をその組合せ回路要素から RA までデータパス内の経路を通って伝搬させる．また，連続クロックでテスト系列の生成/印加，応答の解析を可能にするため，これらの経路が共通部分をもたないようにする．そのため，単一制御可検査性は（単一縮退故障に対して）高い故障検出率が保証され，更にシステムクロックでのテストが必要となる他の故障モデル（トランジション故障，遅延故障など）も検出することが可能である．井筒らの手法 [2] では，内部レジスタを TPG や RA に変更せず，応答の伝搬はデータパス中の経路を利用するため，Wunderlich ら [4] の手法と同等の故障検出率を得るために必要な，ハードウェアオーバヘッドは小さい．しかし，組合せ回路要素を一つずつテストするために，テストセッション数がテスト対象回路要素分だけ必要になるためテスト実行時間が大きくなる．

そこで，本論文では，井筒らの手法 [2] の長所である高い故障検出率と低いハードウェアオーバヘッドを実現し，欠点であったテスト実行時間を短縮するために，単一制御並行可検査性を提案する．単一制御並行可検査性は，複数の組合せ回路要素を同時にテストすることによりテスト実行時間を短縮するように単一制御可検査性を一般化したものである．単一制御並行可検査性は単一制御可検査性と同様にデータパス内のレジスタを TPG や RA に変更せず，PI のみに付加した TPG からの値を伝搬し，PO のみに付加した RA への応答の伝搬を行うため，井筒らの手法 [2] と同様に低いハードウェアオーバヘッドで高い故障検出率を得ることができる．また，低いハードウェアオーバヘッドで与えられたデータパスを単一制御並行可検査になるように設計変更するテスト容易化設計法を提案する．

本論文で提案する BIST の特徴は以下のとおりである．

- 高い故障検出率：提案手法は階層テスト [5] に基づき，テストはデータパス中の回路要素ごとに行われる．実際のデータパスで使用されるほとんどの組合せ回路要素（加算器，減算器，乗算器，マルチプレクサなど）は，ランダムパターンを用いることにより，高い故障検出率を得られることが知られており [6]，提案手法で高い故障検出率が得ることが期待できる．
- 低いハードウェアオーバヘッド：提案手法は井筒らの手法 [2] と同様に TPG，RA を PI，PO のみに付加するので，Wunderlich ら [4] の手法に比べ，ハードウェアオーバヘッドが低いことが期待できる．
- 短いテスト実行時間：提案手法はデータパス中の複数の組合せ回路要素を同時にテストする．そのため，一つずつテストする井筒らの手法 [2] に比べて，テスト実行時間が短くなることが期待できる．
- test per clock 方式：井筒らの手法 [2] と同様，連続クロックで，テストパターンの生成，応答の解析が可能であり，システムクロックでのテストが可能である．ただし，システムクロックは，通常動作時の最大遅延だけから決定するのではなく，テスト動作時の最大遅延も考慮して決定するものとする．

以下，**2.** ではデータパスを定義し，データパスグラフについて述べる．**3.** では単一制御並行可検査性を定義する．**4.** では単一制御並行可検査性に基づくテスト容易化設計法を述べる．**5.** ではベンチマーク回路を用いた実験により，提案手法を評価し，**6.** にてまとめる．

2. データパス

2.1 データパス

レジスタ転送レベル回路はデータパスとコントローラから構成されるが，本論文ではデータパスのみを対象と考える．データパスは以下の構成要素からなる．

▷ 回路要素
▷ データ信号線：回路要素を相互に接続
▷ 制御信号線：コントローラから回路要素へ制御信号を伝達
▷ 状態信号線：回路要素からコントローラへ状態信号を伝達

以下に各構成要素について説明する．

- 回路要素：回路要素は PI，PO，ラッチ，レジスタ，マルチプレクサ，演算モジュールに分類される．このうちマルチプレクサ，演算モジュールを組合せ回路要素と呼ぶ．

回路要素に対してデータ信号線が接続される端子をデータ端子，制御信号線が接続される端子を制御端子，状態信号線が接続される端子を状態端子と呼ぶ．デー

タ端子は回路要素に信号を入力するためのデータ入力端子と回路要素からデータを出力するためのデータ出力端子に分類される（以下，データ入力端子を入力端子，データ出力端子を出力端子と呼ぶ）．データパス上のすべての回路要素のデータ端子は等しいビット幅をもつものとする．

PI，PO： PI はデータパス外部からデータパスにデータを入力するための回路要素，PO はデータパスから外部にデータを出力するための回路要素である．便宜上，PI は一つの出力端子のみをもち，PO は一つの入力端子のみをもつものとする．

マルチプレクサ： マルチプレクサは二つの入力端子と一つの出力端子，1 ビットの制御端子をもつ．制御端子の値に従って，対応する入力端子の値をそのまま出力端子に出力する．

演算モジュール： 演算モジュールは一つまたは二つの入力端子，一つの出力端子，たかだか一つの制御端子，たかだか一つの状態端子をもつ．入力端子に与えられた値に対して演算を行い，その結果を出力端子に出力する．

ラッチ，レジスタ： ラッチ，レジスタはいずれも記憶素子である．ラッチは一つの入力端子と一つの出力端子をもつ．入力端子に与えられた値を記憶し，その値を次のクロックサイクルで出力端子に出力する．レジスタは一つの入力端子と一つの出力端子，1 ビットの制御端子をもつ．制御端子の値によって，入力端子の値を新たに記憶するか（ロード），既に記憶している値を保持する（ホールド）．記憶している値は次のクロックサイクルで出力端子に出力する．

- データ信号線：データ信号線（以降，単に信号線と呼ぶ）は相異なる回路要素の出力端子と入力端子を接続する．複数の信号線を同一の出力端子に接続できる（ファンアウト可能）が，入力端子に接続する信号線は 1 本のみとする．
- 制御信号線，状態信号線：制御信号線はコントローラからデータパス上の回路要素に対して制御信号を伝達し，状態信号線は回路要素からコントローラに対して状態信号を伝達する．

2.2 データパスグラフ

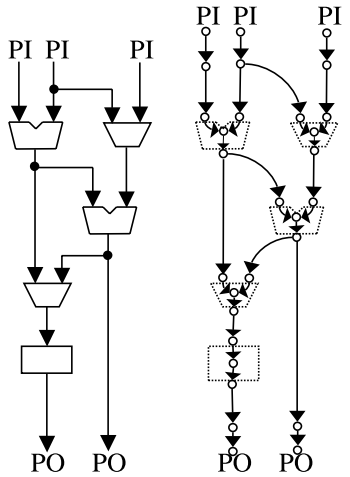
データパスに対してデータパスグラフ $G = (V, A)$ を次の有向グラフとして定義する．

- $V = V_1 \cup V_2$

ここで V_1 はデータパス中のすべての回路要素の集合，V_2 はデータパス中のすべてのデータ端子の集合とする．以下では，データパスグラフ G の頂点集合 V_1, V_2 をそれぞれ $G.V_1$, $G.V_2$, と表す．また，すべての組合せ回路要素の頂点集合を $V_C(\subseteqq V_1)$ として表す．

- $A = A_1 \cup A_2 \cup A_3$

ここで A_1 はデータ信号線を表し，$A_1 = \{(x, y) \in V_2 \times V_2 |$ 出力端子 x と入力端子 y がデータ信号線で接続 $\}$ とする．また，A_2，A_3 はそれぞれ，入力端子と回路要素を接続する信号線，回路要素と出力端子を接続する信号線を表す．すなわち，$A_2 = \{(x, u) \in V_2 \times V_1 | x$ は u の入力端子 $\}$，$A_3 = \{(u, x) \in V_1 \times V_2 | x$ は u の出力端子 $\}$ とする．以下では，データパスグラフ G の辺集合 A_1, A_2, A_3 をそれぞれ $G.A_1$, $G.A_2$, $G.A_3$ として表す．



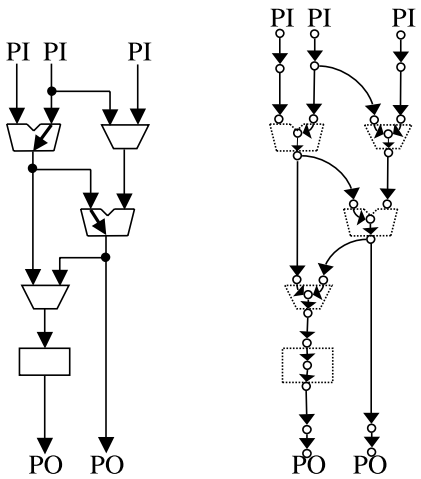
(a) Data path (b) Data path digraph

図 1 データパスとデータパスグラフ
Fig. 1 Data path and data path digraph.

図 1 (a) のデータパスに対するデータパスグラフを図 1 (b) に示す．本論文で対象とするデータパスは，そのデータパスグラフにおいてすべての入力端子は，少なくとも一つの PI から到達可能であり，すべての出力端子は少なくとも一つの PO に到達可能であるものとする．また，データパスグラフはその対応するデータパスと同一視する．

またデータパスグラフの拡張として，演算器にスルー機能が実現されている場合のデータパスグラフについても考える．スルー機能は，演算モジュールにおいて入力端子と出力端子の間での任意の値の伝搬を保証する機能である．

そこで，データパスグラフ $G = (V, A)$ の部分



(a) Data path with thru (b) Data path digraph with thru

図 2 スルー機能付データパスとスルー制約付データパス部分グラフ
Fig. 2 A data path and data path diagraph with thru function.

グラフとして，スルー制約付データパス部分グラフ $G' = (V, A'(\subseteqq A))$ を定義する．$A' = A_1 \cup A'_2 \cup A_3$ のように定義され，A'_2 は，スルー機能を有する組合せ回路要素の入力端子と回路要素，若しくはレジスタ及びラッチの入力端子と回路要素への対応を表す．

図 2 にスルー機能を有するデータパスに対するスルー制約付データパスグラフを示す．データパスの制御信号線に適切な制御ベクトルを与えることでスルー制約付データパス部分グラフ上の任意の単純経路に沿って任意の値を連続して伝搬することができる．

図 2 (a) のデータパスに対するデータパスグラフを図 2 (b) に示す．

3. 単一制御並行可検査性

3.1 単一制御可検査性

井筒らの手法 [2] は，BIST のためにレジスタ転送レベルデータパスの単一制御可検査性を提案し，与えられたデータパスを単一制御可検査にするための DFT 手法を提案した．データパスの単一制御可検査性を次のように定義している．

[定義 1] データパス中の任意の組合せ回路要素 M において，以下の条件を満たす三つの共通部分をもたない経路 P_1, P_2, P_3 が存在するなら，データパスは単一制御可検査であるという．

- 任意の値がそれぞれ P_1, P_2, P_3 を通って伝搬可能．
- P_1, P_2 は TPG を始点とし，M(注 1)の入力端子を終点とする．
- P_3 は，M の出力端子を始点とし，RA を終点とする． □

P_1, P_2 を M の制御経路，P_3 を M の観測経路と呼ぶ．

単一制御可検査データパスにおいて TPG と RA をそれぞれ PI と PO に置くことにより，組合せ回路要素 M に対して，制御経路を用いて PI から連続したテスト系列を印加し，観測経路を用いて M の応答を連続して PO で観測できる．ほとんどの組合せ回路要素（加算器，減算器，乗算器，シフタ，マルチプレクサなど）はランダムパターンでテスト可能である．また，比較器などランダムパターンでテスト可能でない組合せ回路要素に関しては，テストポイント挿入手法によってランダムパターンでテスト可能にできる [6]．したがって，単一制御可検査データパスに対しては，BIST によって高い故障検出率を得ることができる．

制御経路，観測経路上の，演算モジュールやマルチプレクサは，この経路上の入力端子の値が出力端子に伝搬するように制御する(注 2)．これらを制御する信号は，M のテストの間固定しておけばよい．このように，各組合せ回路要素に対して，一つの制御パターンで十分なので，この性質のことを単一制御可検査性と呼ぶ．

3.2 単一制御並行可検査性

ここでは，複数の組合せ回路要素を同時にテスト可能なように単一制御可検査性を拡張した単一制御並行可検査性を定義する．

[定義 2] $\mathcal{M}$ をデータパス中の組合せ回路要素の集合とする．スルー制約付データパス部分グラフ G' において組合せ回路要素の集合 $\mathcal{M}$ が以下の条件を満たすとき，$\mathcal{M}$ は単一制御並行可検査であるという．

- 以下の条件を満たす，互いに共通部分をもたない木 $T_1, T_2, \ldots, T_m$ が存在．(各 T_i を制御木と呼ぶ．)
  - それぞれの木は PI を根とする．
  - 各組合せ回路要素 $M_i (\in \mathcal{M})$ の各入力端子は木

(注 1)：P_1 と P_2 はそれぞれ異なる TPG を始点とする．
(注 2)：演算モジュールが入力端子の値を出力端子に伝搬する機能（スルー機能）をもたない場合は，テスト容易化設計でスルー機能を付加し，それを利用する．

の葉である．

– 各組合せ回路要素 $M_i(\in \mathcal{M})$ の二つの入力端子は異なる木に属する．

- $G'-(T_1 \cup T_2 \cup \ldots \cup T_m)$ において，以下の条件を満たす，互いに共通部分をもたない経路 $P_1, P_2, \ldots, P_n$ が存在.（各 P_i を観測経路と呼ぶ.）

– 各組合せ回路要素 $M_i(\in \mathcal{M})$ の出力端子が始点である．

– 各経路の終点は PO である． □

$\mathcal{M}$ が単一制御並行可検査なら，TPG，RA をそれぞれ PI，PO に配置し，テスト系列，応答を制御木，観測経路を用いて伝搬することにより，$\mathcal{M}$ に属するすべての組合せ回路要素を同時にテストできる．更に，このテストの間，制御木及び観測経路に現れる演算モジュール及びマルチプレクサの制御信号を固定しておけばよい．つまり，$\mathcal{M}$ に対して，一つの制御パターンで十分である．

データパスのすべての組合せ回路要素を，単一制御並行可検査な組合せ回路要素の集合 $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_l$ に分割できるとする．このとき，各 $\mathcal{M}_i$ に属する組合せ回路要素を同時にテストすれば，データパス全体のテスト実行時間は，全体の組合せ回路要素数ではなく，l に比例すると考えられる．このとき，各 $\mathcal{M}_i$ に属する組合せ回路要素のテストをテストセッションと呼ぶ．

データパス全体のテスト実行時間を短縮するには，テストセッション数が小さくなるように，すべての組合せ回路要素を単一制御並行可検査な集合に分割すればよい．一方，同時にテストされる組合せ回路は，相異なる応答解析器を利用するので，応答解析器が k 個しかなければ，たかだか k 個の組合せ回路要素が単一並行可検査となり得る．これらのことを考慮して，データパスの並行可検査性を以下のように定義する．

[定義 3]（データパスの単一制御 k–並行可検査性） スルー制約付データパス部分グラフ G' において，以下の条件を満たす組合せ回路要素 V_C 上の分割 $P(V_C)$ が存在するなら，データパス DP が単一制御 k–並行可検査であると定義する．

- $\forall S \in P(V_C)$ に対して，S（組合せ回路要素の集合）が単一制御並行可検査
- $\forall S \in P(V_C)$ に対して，$|S| \leq k$
- $|P(V_C)| = \lceil |V_C|/k \rceil$ □

単一制御 k–並行可検査データパスでは，組合せ回路要素集合 V_C を，$\lceil |V_C|/k \rceil$ 個の部分集合（ただし，各部分集合はたかだか k 個の組合せ回路要素を含む）に分割し，各部分集合に属する組合せ回路要素を同時にテストできる．したがって，テストセッション数が単一制御可検査データパスの場合に比べてほぼ $1/k$ に減少する．単一制御 k–並行可検査性は，単一制御可検査性の一般化であり，$k = 1$ の場合，単一制御 k–並行可検査性は単一制御可検査性となる．以下では，k の値を特に指定しないときは，単一制御 k–並行可検査性のことを単一制御並行可検査性と呼ぶ．

4. 単一制御並行可検査テスト容易化設計法

本章では，与えられたデータパスを単一制御並行可検査データパスに設計変更するための DFT 手法を示す．

4.1 問題の定式化

与えられたデータパスにおいて，同時にテストする組合せ回路要素に対して，共通部分をもたない制御木及び観測経路が存在しない場合，単一制御並行可検査にするためには，データパスに新しい経路を付加しなければならない．提案する DFT では，この経路付加は，マルチプレクサ（テスト MUX と呼ぶ）を用いて実現する．また，制御経路，観測経路に演算モジュールが現れる場合，任意の値を伝搬可能とするために，この演算モジュールにスルー機能を付加する．そこで，単一制御並行可検査のための DFT を，次の最適化問題として定式化する．

[定義 4]（単一制御並行可検査 DFT） 単一制御並行可検査のための DFT を次の最適化問題として定義する．

- 入力：データパス，並行度 k
- 出力：単一制御 k–並行可検査なデータパス
- 最適化目標：付加する DFT 要素（スルー機能，テスト MUX）のハードウェア量最小化 □

4.2 DFT アルゴリズムの概要

単一制御並行可検査 DFT のための発見的アルゴリズムを示す．本アルゴリズムは，以下の 2 段階からなる．

ステージ 1： 与えられたデータパスを単一制御可検査に設計変更する．この設計変更には，井筒ら [2] の手法を改良したものを用いる．詳細は **4.3** に示す．

ステージ 2： ステージ 1 で得られたデータパスを単一制御 k–並行可検査なデータパスに設計変更する．ここでは，すべての組合せ回路要素に対して制御経路と観測経路を決定するまで，以下の三つのステップを繰り返す．

（1） スケジューリング：同時にテストする組合せ

回路要素として，未スケジューリング組合せ回路要素から k 個選択する．**1.** で詳細を示す．

（2） 観測経路に関する DFT：(1) で選ばれた k 個の組合せ回路要素に対して，それぞれ観測経路を決定する．ここで，テスト MUX を付加した場合，付加したテスト MUX は未スケジューリング組合せ回路要素となり，後のスケジューリングの対象となる．**1.** で詳細を示す．

（3） 制御経路に関する DFT：(1) で選ばれた k 個の組合せ回路要素に対して，それぞれ制御経路を決定する．(2) と同様に，付加したテスト MUX は未スケジューリング組合せ回路要素であり，後にスケジューリング対象となる．**4. 5** で詳細を示す．

4. 3 単一制御可検査 DFT（ステージ 1）

このステージの目的は，与えられたデータパスを最小のハードウェアオーバヘッドで単一制御可検査にすることであり，井筒ら [2] の手法を改良した手法を用いる．文献 [2] の手法は，組合せ回路要素一つずつを順に処理し，互いに共通部分をもたない二つの制御経路と一つの観測経路をもつように DFT 要素を付加する．先に加えた DFT 要素は後の組合せ回路要素で再利用できるので，全体のハードウェアオーバヘッドを削減するには，必要性の高い DFT 要素を早い段階で付加することが重要である．そこで，本論文の手法では，文献 [2] の手法を適用する前に，以下に定義するカットエッジに基づいた処理を行う．

[定義 5] データパスグラフ G に対して，$e \in G.A_3$ を取り除いて得られるグラフを $G'(e)$ と表す．つまり，$G'(e).V = G.V$ かつ $G'(e).A = G.A - \{e\}$ である．$G'(e)$ においてどの PI からも到達不能な組合せ回路要素 M が存在する場合，$e(\in G.A_3)$ をカットエッジと呼び，M は e によって支配されるという． □

データパスグラフがカットエッジ e をもつなら，e によって支配される組合せ回路要素は共通部分をもたない二つの制御経路をもつことができず，MUX を付加することにより，新たな経路を追加する．一つの MUX を加えることにより，複数のカットエッジを解消できる可能性がある．そこで，カットエッジに対して次の半順序関係を定義し，その半順序に従ってカットエッジを処理する．

[定義 6] データパスグラフ G におけるカットエッジ e に対して，e によって支配されるすべての組合せ回路要素を $D(e)$ と表す．ここで，G におけるカットエッジの半順序を次のように定義する．

```
while (there exists a cut edge in G){
 e :=minimal cut edge in G;
 V(e) :=the set of all vertices unreachable from any PI
        in G(e);
 M(e) :=the maximal subgraph of G(e) with vertex set
        V(e);
 construct a spanning tree T of M(e)
 with the root u (u is the end vertex of e);
 V' := {v ∈ V(e) | v has an outgoing edge
 in M(e).E − T.E};
 choose any v from V' and insert a test MUX M
 in front of the v's input port;
 if (there exists a PI i unreachable to e)
  connect i to the other input port of M;
 else
  choose any PI i and connect i to
  the other input port of M;
 update G by adding M and the lines;
}
apply the DFT method in [2];
```

図 3 単一制御可検査 DFT
Fig. 3 DFT for single-control testability.

カットエッジ e 及び e' において，$D(e) \subset D(e')$ のとき，またそのときに限り，$e < e'$ とする． □

もし，二つのカットエッジ e_1 と e_2 が $e_1 < e_2$ を満たすなら，e_1 のために加えられる MUX によって，e_2 がカットエッジでなくなることがある．このように提案手法では，最初に e_1 に対してテスト MUX を付加しておき，必要であれば e_2 に対してもテスト MUX を付加する．カットエッジがなくなるまでテスト MUX を付加し，その後は [2] で提案した手法を適用して DFT を行う．ただし，提案手法は最初のステージでは制御経路及び観測経路の決定は行わない．図 3 に我々の DFT 手法の最初のステージを示す．

4. 4 スケジューリング及び観測経路に関する DFT（ステージ 2）

スケジューリングは同時にテストする組合せ回路要素の集合 $\mathcal{M}$ を決定することである．提案する DFT アルゴリズムは，最小のハードウェアオーバヘッドで観測経路を実現できるように $\mathcal{M}$ を決定する．観測経路を制御経路より先に決定するのは，制御経路は異なる組合せ回路要素で共有可能だが，観測経路は共有不能であり，可観測性のための DFT ハードウェアオーバヘッドが支配的（並行度 k が大きいとき，特に顕著）になるためである．

スケジューリングにおいて，組合せ回路要素が既に決定された集合 $\mathcal{M}$ に含まれるならスケジューリング済み，そうでないなら未スケジューリング組合せ回路

要素という．ただし，すべての PO はスケジューリング済みとする．未スケジューリング組合せ回路要素から同時にテストする組合せ回路要素を決定する 2 ステップを以下に示す．

（1） 候補組合せ回路要素の選択

- k 個以上未スケジューリング組合せ回路要素が存在する場合：スケジューリング済み組合せ回路要素に隣接するすべての未スケジューリング組合せ回路要素を候補とする．候補数 ℓ が k 未満ならば，候補が k 個になるように，未スケジューリング組合せ回路要素を適当に候補に加える
- $m(< k)$ しか未スケジューリング組合せ回路要素が存在しない場合：すべての未スケジューリング組合せ回路要素を候補とする．

（2） 組合せ回路要素集合 $\mathcal{M}$ 及び $\mathcal{M}$ に含まれる組合せ回路要素の観測経路の決定

次に示すようにデータパスグラフを構築し，流量 k（ただし，候補数が $m(< k)$ 個なら流量 m）の最小費用流問題を解く．図 4 に $k = 3$ の例を示す．

- 各候補組合せ回路要素に対する辺をもつダミー頂点を付加し始点とする．
- すべての PO からの辺をもつダミー頂点を付加し終点とする．
- 各候補組合せ回路要素から PO へテスト MUX に対応する辺を加える．
- 得られたグラフにおける各辺 (u, v) に対して，容量を $c(u, v) = 1$ と定義し，コスト $p(u, v)$ を以下のように定義する．

$$\begin{aligned} p(u,v) &= cost\_thru(u) && u \text{ が } v \text{ の入力端子} \\ &= cost\_MUX && (u,v) \text{ が付加された MUX} \\ &= 0 && \text{上記以外の場合} \end{aligned}$$

（$cost\_thru(u)$ は u から v の出力端子へのスルー機能のハードウェアコスト．スルー機能が既に付加されている場合は，$cost\_thru(u) = 0$．$cost\_MUX$ は MUX のハードウェアコスト）

グラフの最小費用流は $\mathcal{M}$ を構成する組合せ回路要素数であり，最小費用流で用いる経路は，最小のハードウェアオーバヘッドで実現できる観測経路を表す．観測経路のために，スルー機能とテスト MUX が付加される．

4.5 制御経路に関する DFT（ステージ 2）

与えられた組合せ回路要素集合 $\mathcal{M}$ に対して，$\mathcal{M}$ に含まれるすべての組合せ回路要素の制御経路を図 5 の手続きによって決定する．この手続きでは，$\mathcal{M}$ の要素は一つずつ考慮する．組合せ回路要素の制御経路は観測経路と同様に決定される：G から $\mathcal{M}$ に含まれる組合せ回路要素のすべての観測経路を除去して構成されるグラフ G' において，流量 2 の最小費用流問題を解く．G の代わりに G' において扱うのは，決定する制御経路が既に決定されている観測経路と共通部分をもたないようにするためである．すべての制御経路が手続きによって互いに共通部分をもたない木を決定するので，各繰返しステップの最後に G' を更新する．

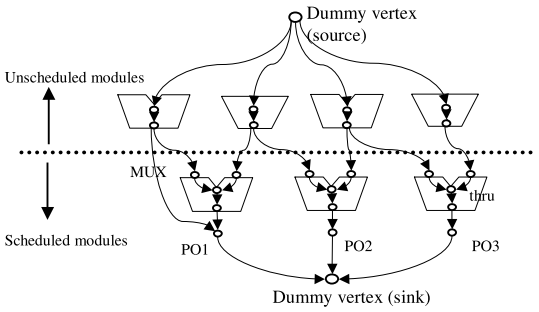


図 4 $(k = 3)$ スケジューリング例
Fig. 4 Example of scheduling $(k = 3)$.

```
G' := a digraph obtained from G by removing
  the observation paths of all modules in M
while (M ≠ ∅){
  choose any module v ∈ M;
  determine control paths P1 and P2 of v
  (by finding the minimum cost flow of two);
  add DFT elements for P1 and P2;
  for each module u on Pi (i = 1, 2)
    update G' by removing the outgoing edge
    from the input port of u that is not on Pi;
  M:=M-{v}
}
```

図 5 制御経路に関する DFT
Fig. 5 DFT for control paths.

図 5 において，組合せ回路要素に二つの共通部分をもたない制御経路が見つからない場合，テスト MUX M を制御経路を構成するために付加する．テスト MUX M は，付加されたときには未スケジューリング組合せ回路要素とみなされ，M の制御経路と観測経路は後で決定する．データパスの構造によっては，M の入力端子に新たにもう一つのテスト MUX が制御経路を決定するために加えられる可能性があり，更にテスト MUX のためのテスト MUX が付加され続ける可能性もある．連続したテスト MUX の付加を避けるた

めに，必要に応じて図 5 の手続きに例外処理を行う．この場合，グラフ G において M の制御経路の決定を妨げる観測経路を有する組合せ回路要素 $M' \in \mathcal{M}$ を検出し，M' から（テスト MUX を加えることで）PO に直接観測経路を生成することによって M' の観測経路を変更する．

4.6 テスト MUX 挿入に関する考察

提案した DFT 手法では，ある組合せ回路要素 M の制御経路若しくは観測経路を新たに生成するために，テスト MUX（M' とする）を付加することがある．このテスト MUX M' もテストの対象であり，追加時には，M' は未スケジューリング組合せ回路要素とみなされる．つまり，M のスケジュールのために M' を加えることは，未スケジューリング組合せ回路要素数を減らすことには寄与しない．

そこで，テスト MUX M' を付加する代わりに，M を以降のテストセッションに移すことによって，全体として同じテストセッション数を小さいハードウェアオーバヘッドで実現できる可能性がある．ただし，M のテストのために M' を追加すると，M' を他の組合せ回路要素の制御経路及び観測経路のために利用可能であり，他のテスト MUX の追加を防ぐことがある．このため，M' を付加しないことで，同じテストセッション数を小さいハードウェアオーバヘッドで実現できるとは限らない．そこで，上記の手法の有効性を実験的に調査する．そのために，図 5 において，M が前節で述べた例外処理を必要とする組合せ回路要素である場合，$M(\in \mathcal{M})$ を $\mathcal{M}$ から取り除き，未スケジューリング組合せ回路要素とするように変更した場合についても，実験結果を次節に示す．

ただし，上記の変更によって得られるデータパスは，変更しない場合と同じテストセッション数をより小さ

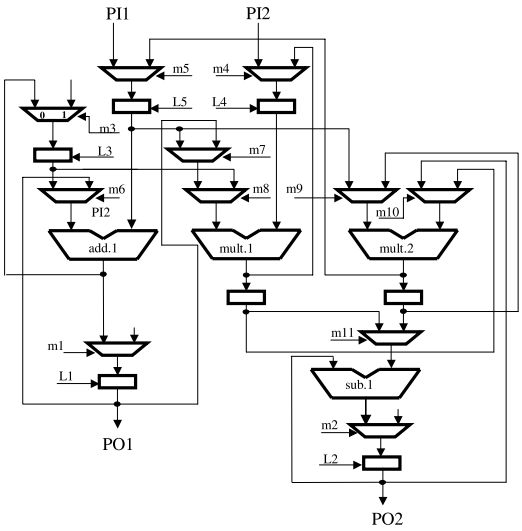


図 6　Paulin 元回路
Fig. 6　Paulin original circuit.

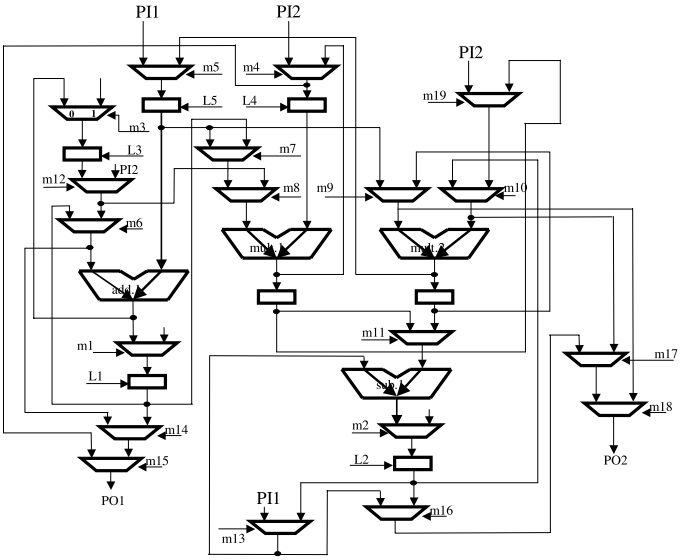


図 7　DFT 適用後 ($k = 2$)
Fig. 7　Paulin DFT circuit($k = 2$).

いハードウェアオーバヘッドで実現する場合であっても，k 並行可検査の定義を満たすとは限らない．これは，得られたデータパスの組合せ回路要素が少なくなるために，テストセッション数が $\lceil |V_c|/k \rceil$ となることを保証できないからである．

4.7 適 用 例

提案した手法をレジスタ転送レベルベンチマーク回路 Paulin（図 6）に適用したデータパスを図 7 に示す．図 7 は，並行度 k を 2 として得られた単一制御 2 並行可検査なデータパスである．図 7 における $m12, m13, m14, m15, m16, m17, m18$ は，ステージ 1 の処理により加えられた DFT 要素である．$m13$ は，図 6 の $sub1$ と $m2$ を接続する信号線が，**4.3** で示したカットエッジであるために加えられた MUX であり，$m12$ もカットエッジ処理によって加えられた MUX である．また，$m14, m15, m16, m17, m18$ は井筒らの手法 [2] に基づいて挿入された MUX である．表 1 では，ステージ 2 の適用過程を示す：ID はテストセッションの ID を示し，test module は，セッションでテストされる組合せ回路要素を表す．DFTelements は，セッションを行うために付加された DFT 要素を示す．例えば，add.1-r は，加算器 add.1 の右入力から出力へのスルー機能の付加を表す．

表 1　Paulin に対するステージ 2 の適用過程 ($k = 2$)
Table 1　Application process of the stage2 in Paulin($k = 2$).

ID	test module		DFTelements
1	m15	m18	add.1-r,mult.1-r
2	m14	m17	
3	m1	m16	sub.1-r
4	m4	m2	mult.1-l
5	add.1	sub.1	
6	m6	m9	mult.2-r
7	m12	m11	mult.2-l
8	m5	m13	
9	m3	mult.1	
10	m8	m10	MUX(m19)
11	m7	mult.2	add.1-l
12	m19		

5. 実 験 結 果

この章では，提案手法の実験結果を示し，提案手法と井筒らの手法 [2] を比較する．

実験に使用したレジスタ転送レベルベンチマーク回路は，Tseng 及び Paulin と 3rd Lattice Wave Filter（LWF）である．なお，井筒らの手法では Tseng での適用結果について示されていないため，比較については Paulin と LWF で行う．表 2 にそれらのベンチマーク回路の特性を示す：#PI,#PO,#Reg,#MUX,#OP は それぞれ PI 数，PO 数，レジスタ数，マルチプレクサ数，演算モジュール数を表す．回路面積の単位は gate equivalent で，論理合成ツールとして Auto-LogicII（Mentor Graphics）を用い，ALTERA 社の論理合成ライブラリを使用して，計算機上で論理合成を行うことによって得た．

表 3 は，DFT によって付加されるテスト MUX とスルー機能数を示し，表 4 ではハードウェアオーバヘッド（HW/OH）を示す．これらの表において，Modification については **4.6** で示した変更によって得られた結果である．一般に BIST を実現する場合，PI 及び PO には TPG 及び RA を付加するので，表 4 にそれらは含まないものとした．ただし，$k = 3$ の場合は，新たに応答を観測するための LFSR を付加したため，

表 2　回路特性
Table 2　Circuit characteristics.

	#PI	#PO	#Reg	#MUX	#OP	Area(8 bit)	Area(16 bit)	Area(32 bit)
Tseng	3	2	6	7	7	858.6	2499.4	8085.0
Paulin	2	2	7	11	4	1142.9	3818.9	13778.9
LWF	2	2	5	5	3	355.8	735.0	1493.4

表 3　付加テスト MUX 及びスルー機能
Table 3　Added MUX and thru function.

	our previous method [2]		Our methods							
	($k = 1$)		$k = 1$		$k = 2$		$k = 3$		Modification ($k = 3$)	
Circuit	#MUX	#THRU	#MUX	#THRU	#MUX	#THRU	#MUX	#THRU	#MUX	#THRU
Tseng			8	8	10	11	15	8	14	8
Paulin	8	5	7	6	8	7	14	7	12	7
LWF	4	3	2	4	4	4	11	3	7	2

表4 ハードウェアオーバヘッド
Table 4 Hardware overhead.

		our previous method [2]	Our methods			
		($k=1$)	$k=1$	$k=2$	$k=3$	Modification ($k=3$)
Circuit	bit width	HW/OH(%)	HW/OH(%)	HW/OH(%)	HW/OH(%)	HW/OH(%)
Tseng	8		33.47	41.25	64.07	61.19
	16		22.61	27.83	42.43	40.48
	32		13.68	17.01	25.74	24.54
Paulin	8	22.50	19.15	26.17	46.38	41.98
	16	13.23	12.39	15.38	26.66	24.11
	32	7.27	6.80	8.66	14.48	11.00
LWF	8	35.10	23.66	37.55	109.13	78.92
	16	33.32	22.34	35.59	100.83	72.05
	32	32.47	21.71	34.66	96.87	68.78

表5 テストセッション
Table 5 Test session.

	our previous method [2]	Our methods			
	($k=1$)	$k=1$	$k=2$	$k=3$	Modification ($k=3$)
Circuit	session	session	session	session	session
Tseng		21	12	10	10
Paulin	23	22	12	10	10
LWF	12	10	6	7	6

そのハードウェアオーバヘッドを加えたものを示している．表 4 より，提案手法の $k=1$ の場合，井筒らの手法 [2] と比較してハードウェアオーバヘッドが改善されていることがわかった．これは，今回追加したカットエッジに基づく処理が効果的に働いたためである．表 5 にテストセッション数を示す．表 4 及び表 5 から，ハードウェアオーバヘッドとテスト実行時間のトレードオフがわかる．提案手法の $k=2$ の場合，以前に提案した手法 [2] よりもテストセッション数が削減できており，提案手法はテスト実行時間を改善している．しかしながら，テスト MUX のテストによってテストセッション数が増加したため，$k=3$ におけるテストセッション数は我々が予期していたよりも（組合せ回路要素数の約 1/3）より悪くなった．**4.6** で示した修正 DFT が今回実験した回路に関しては有効に働いたことが，表 4，表 5 から示される．故障検出率については示さないが，テスト対象組合せ回路要素に対して TPG からの値を伝搬し，その応答を RA へ伝搬する手法は井筒らの手法 [2] と同じであるため，井筒らの手法と同じ高い故障検出率が達成される．

6. むすび

本論文では，BIST のための RTL データパスの単一制御並行可検査性とその DFT について提案した．提案手法の BIST の利点は高い故障検出率，低いハードウェアオーバヘッド，短いテスト実行時間，システムクロックでテスト可能である．

提案手法は，test per clock 方式に基づき，テスト対象組合せ回路要素に対して，連続クロックでテストパターンの生成，応答の解析が可能であるのでシステムクロックでテスト可能であると考えている．ただし，システムクロックは，通常動作時の最大遅延だけから決定するのではなく，テスト動作時の最大遅延も考慮して決定するものとする．したがって，テスト動作時の最大遅延が通常動作時の最大遅延よりも大きい場合，通常動作時の回路性能を犠牲にしていると考えられる．この点を改良するには，テスト動作時の最大遅延が通常動作時の最大遅延より大きくならないように DFT 要素を付加することが考えられるが，これは今後の課題である．

また，提案手法では，MUX と演算モジュールを等価と扱っている．しかし，MUX のテスト系列長は演算モジュールのものより短い場合が多く，MUX と演算モジュールのテストセッションを分けることによってテスト実行時間を短くすることができると考えられる．したがって，MUX と演算モジュールを別のテストセッションでテストするようなスケジューリングの考案が必要である．更に，提案手法の有効性を明らかにするため，より大規模な回路への提案手法の適用も必要である．

提案手法は，RTL 回路のデータパスに対する BIST 手法である．したがって，コントローラも含めた RTL

回路全体の BIST 手法の考案が今後の課題である．

謝辞　本研究に際し，多くの貴重な意見を頂いた本学の井上美智子助教授，大竹哲史助手，情報論理学講座の皆様方に深く感謝致します．本研究は一部，日本学術振興会・科学研究費補助金・基盤研究 B(2)（課題番号 09480054）の研究助成，及び（株）半導体理工学研究センター（STARC）との共同研究による．

文　献

[1] M. Abramovici, M.A. Breuer, and A.D. Friedman, Digital System Testing and Testable Design, Computer Science Press, 1990.

[2] 井筒　稔，和田弘樹，増澤利光，藤原秀雄，“単一制御可検査性に基づくレジスタ転送レベルデータパスの組込み自己テスト容易化設計法，” 信学論 (D-I), vol.J84-D-I, no.1, pp.69–77, Jan. 2001.

[3] B. Koenemann, J. Mucha, and G. Zwiehoff, “Built-in logic block observation techniques,” Proc. 1979 IEEE Test Conf., pp.37–41, 1979.

[4] A.P. Stoele and H.J. Wunderlich, “Hardware-optimal test register insertion,” IEEE Trans. Comput. Aided Des. Intergrated Circuits & Syst., vol.17, no.6, pp.531-539, 1998.

[5] I. Ghosh, A. Raghunathan, and N.K. Jha, “Design for hierarchical testability of RTL circuits obtained by behavioral synthesis,” Proc. 1995 IEEE Int. Conf, on Computer Desing, pp.173–179, 1995.

[6] I. Ghosh, N.K. Jha, and S. Bhawmik, “A BIST scheme for RTL circuits based on symbolic testability analysis,” IEEE Trans. Comput. Aided Des. Intergrated Circuits & Syst., vol.19, no.1, pp.111–128, 2000.

（平成 13 年 7 月 11 日受付，10 月 18 日再受付）

山口　賢一　（学生員）

平 11 奈良高専専攻科・電子情報工学専攻了．平 13 奈良先端科学技術大学院大学博士前期課程了．現在同大博士後期課程に在学中．現在テスト容易化設計の研究に従事．

和田　弘樹　（正員）

平 8 阪大・工・通信卒．平 10 奈良先端大情報科学博士前期課程了．平 13 同大博士後期課程了．現在（株）日立製作所中央研究所に勤務．現在テスト容易化設計の研究に従事．博士（工学）．

増澤　利光　（正員）

昭 57 阪大・基礎工・情報卒．昭 62 同大大学院博士後期課程了．同年同大情報処理教育センター助手．同大基礎工助教授を経て，平 6 奈良先端科学技術大学院大学情報科学研究科助教授．平 12 阪大基礎工学研究科教授，現在に至る．平 5 コーネル大客員準教授（文部省在外研究員）．分散アルゴリズム，並列アルゴリズム，テスト容易化設計，テスト容易化高位合成に関する研究に従事．工博．ACM，IEEE，EATCS，情報処理学会各会員．

藤原　秀雄　（正員：フェロー）

昭 44 阪大・工・電子卒．昭 49 同大大学院博士課程了．同大・工・電子助手，明治大・工・電子通信助教授，情報科学教授を経て，現在奈良先端大・情報科学教授．昭 56 年ウォータールー大客員助教授．昭 59 年マッギル大客員準教授．論理設計論，フォールトトレランス，設計自動化，テスト容易化設計，テスト生成，並列処理，計算複雑度に関する研究に従事．著書「Logic Testing and Design for Testability」（MIT Press）など．大川出版賞，IEEE Computer Society Outstanding Contribution Award，IEEE Computer Society Meritorious Service Award など受賞．情報処理学会会員，IEEE Computer Society Golden Core Member，IEEE Fellow．