完全故障検出効率を保証するデータパスの非スキャンテスト容易化設計法

和田　弘樹[†]　増澤　利光[†]　Kewal K. Saluja[††]　藤原　秀雄[†]

A Non-Scan DFT Method for Data Paths to Provide Complete Fault Efficiency

Hiroki WADA[†], Toshimitsu MASUZAWA[†], Kewal K. SALUJA[††], and Hideo FUJIWARA[†]

あらまし　本論文では，レジスタ転送レベルデータパスを対象とした完全故障検出効率を保証するテスト容易化設計法，及び，対応するテストプラン生成法を提案する．提案するテスト容易化設計法は，階層テスト生成法に基づいており，組合せ回路用テスト生成ツールによるテスト生成を可能としている．組合せ回路用テスト生成を利用する従来法の完全スキャン設計法に比べ，提案手法は，ハードウェアオーバヘッドが小さく，テスト生成時間，テスト実行時間が短い．提案するテスト容易化設計法の時間計算量は $O(n)$（n はデータパス中の回路要素数）であり，テストプラン生成法は各回路要素のテストプランを $O(n^2)$ 時間で生成する．

キーワード　テスト容易化設計，データパス，階層テスト，完全故障検出効率

1. まえがき

近年における論理回路の大規模化に伴い，論理回路の故障の有無を調べるテストが重要となり，故障の有無で論理回路の出力が異なる入力系列（テスト系列）を印加するテストの要求も高まっている．特に単一縮退故障集合に対して完全故障検出効率(注1)をもつテスト系列でのテストは，回路の信頼性を保証する観点からも極めて有用だが，一般の順序回路に対して故障検出効率が高いテスト系列は生成困難である．そこで，単一縮退故障集合に対して高い故障検出効率をもつテスト系列が容易に（組合せ回路に対するテスト生成と同程度の計算量で）生成できるように論理回路の設計を変更するテスト容易化設計（以下 DFT）が実用化されている．

完全スキャン設計 [1] は現在一般的な DFT であり，ゲートレベル回路中の全フリップフロップ（以下 FF）をスキャンフリップフロップに置き換えることで，FF の値を外部から制御/観測可能としている．したがって，FF を外部入出力端子で置き換えて得られる組合せ回路（以下，核回路）に対してテスト生成を行えば，完全故障検出率をもつテスト系列が容易に得られるが，以下の問題点がある．

1. 論理合成後の回路を変更するので，タイミング等の合成時の制約を損なう．
2. ハードウェアオーバヘッドが大きい．
3. テスト系列長が大きい．
4. at-speed テストができない．

上記の問題点を解消するために，ゲートレベル回路に変換される前の設計を対象とした DFT が提案されている [3], [7]．ゲートレベルに変換される前の回路はデータを処理するデータパスとデータパスの動作を制御するコントローラという二つの部分回路で構成され，データパスはレジスタ転送レベル（以下 RTL）回路として設計される．ここで RTL 回路とはレジスタやマルチプレクサ，演算器などの回路要素を相互に接続したものを指す．一般に回路のほとんどはデータパスなので，RTL データパスを対象とした DFT は上記の問

[†] 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 Science, Nara Institute of Science and Technology, 8916–5 Takayama-cho, Ikoma-shi, 630–0101 Japan
[††] ウィスコンシン大学，米国
Department of Electrical & Computer Engineering, University of Wisconsin-Madison 1415 Engineering Drive, Madison, WI 53706–1691 USA

(注1)：テスト生成アルゴリズムは故障に対してテスト系列の生成を試みることで，故障の冗長/非冗長を判定するが計算時間の問題から実用上は判定を打ち切ることがある．ここで全故障数を N，判定を打ち切った故障数を Na，とすると故障検出効率 FE は $\frac{N-Na}{N}$ で与えられ，$FE = 100\%$ の場合を完全故障検出効率という．

題点の解消に大きく貢献することが期待されており，これまでにいくつかの提案が行われている [3], [7]．本論文でも RTL データパスに対する DFT 手法を提案する．以下に RTL データパスに対する既知の DFT 手法について述べる．

まず RTL 回路に対して DFT を行うことで，論理合成後の回路への DFT が不用となるため，問題点 1 は解消される．問題点 2.〜4. を解消する目的で，データパスが通常使用するデータ転送経路を通じてレジスタの値の制御/観測を行う DFT として直交スキャン設計法 [3], [4] や Genesis [5]〜[7] が提案されている．一般に完全スキャン設計の回路をテストする場合，通常動作時とは別系統のクロック系に通常動作時よりも周波数の低いクロックを与える必要がある．つまり，通常動作時に用いられるクロック系に通常動作時と同じ周波数のクロックを与えるテスト（at-speed テスト）ができない．直交スキャンや Genesis ではテスト実行時のテストパターンや応答の伝達にデータパスの通常動作時のデータ転送経路を利用するので，通常動作時のクロック系に通常動作時と同じ周波数のクロックを与えるテストが可能である．よって問題点 4. を解消している．

直交スキャン設計では，データパス上の各レジスタの値を外部入出力端子から一度に制御/観測できるように，レジスタに対するホールド機能と演算器に対するマスク素子を追加する．マスク素子は演算器の入出力間での値の伝達に必要な定数を発生する．直交スキャンでは，DFT とレジスタの制御/観測のためのデータ転送経路発見にかかる時間計算量が $O(n^2)$（ただし n はデータパス上の回路要素数）となり，ハードウェアオーバヘッドも完全スキャンより小さい．しかし核回路全体に対して一括してテスト生成を行う必要があり，データパスの大規模化に伴うテスト生成時間の増加は避けられない．

またテスト生成時間を短縮する手法として階層テスト生成法 [2] がある．階層テスト生成法ではデータパス上の故障 f に対して 2 段階のテスト生成を行う．第 1 段階では f が発生する回路要素 M のゲートレベル回路を用いてテストベクタ v を生成する．第 2 段階では M に対して RTL 回路を用いたテストプラン生成を行う．テストプランとは外部入力から M の入力へ値を伝達し，また M の出力の値を外部出力まで伝達するためのデータパスへの制御入力の時系列である．第 2 段階で v のためのテストプラン生成に失敗した場合，第 1 段階にバックトラックして f に対する新たなテストベクタ v' を生成し，最終的にテストプラン生成が失敗した場合は f を冗長とみなす．

また階層テスト生成を志向した RTL データパスの DFT として Genesis [5]〜[7] がある．Genesis では第 1 段階として回路要素ごとにゲートレベル回路を用いたテスト生成を行う．次に第 2 段階として各回路要素に対して，外部入力から回路要素の入力へ任意の値を伝達し，また回路要素の任意の値を外部出力まで伝達できるテストプランの生成を試みる．テストプランが存在しない場合には DFT として外部入力から直接値を代入したり，外部出力で直接値を観測するためのマルチプレクサ（テストマルチプレクサ）と配線を RTL データパス上の適切な回路要素の前後に挿入する．このような手法によって従来手法 [2] で生じる第 2 段階から第 1 段階へのバックトラックがなくなる．しかし，テストプラン探索時のバックトラックは発生するので，データパスの大規模化とともにテストプラン生成時間が急速に増大する．また，テストマルチプレクサと外部入出力への配線は大域的で配線長が増加しやすく，回路面積も増大しやすい．

そこで本論文では完全故障検出効率を保証するテストが実行可能であり，かつ完全スキャン設計の問題点 1.〜4. を解決する RTL データパスに対する DFT 手法とテストプラン生成法を提案する．提案する手法では RTL データパス上のすべての回路要素に対してテストプランが高速に（バックトラックせずに）求められるようにマスク素子とレジスタのホールド機能を追加する．また生成されるテストプランは Genesis のテストプランと同様にデータパスの外部入力から回路要素への任意の値の伝達と，回路要素が出力し得る任意の値のデータパスの外部出力への伝達を保証する．提案する DFT 手法を適用した RTL データパス上の各回路要素に対してテスト生成の第 1 段階で完全故障検出効率をもつテストベクタを求め，第 2 段階で求めるテストプランを用いてテストベクタの入力と応答の観測を行えばデータパス全体に対して完全故障検出効率を達成できる．

直交スキャン設計では，データパス上のすべてのレジスタの値を同時に制御/観測することを要求する．一方，提案手法ではすべてのレジスタを同時に制御/観測する必要はなく，一つの回路要素に接続する入力信号線に，外部入力から任意の値の組を設定し，出力信号線の値を外部出力で観測できれば十分である．した

がって提案方式は，直交スキャン設計よりも DFT で実現する回路の能力が真に小さいため，直交スキャン設計よりハードウェアオーバヘッドが小さい．また，マスク素子，ホールド機能の追加は局所的な設計変更であり，Genesis のテストマルチプレクサに比べても提案手法はハードウェアオーバヘッドが小さい．よって提案手法は従来の RTL データパスを対象とした DFT に比べてハードウェアオーバヘッドが小さい．

2. 対象とするデータパス

本論文で扱う RTL データパスについて述べる．

2.1 データパス

データパスは信号線と回路要素で構成される．信号線は制御信号線，状態信号線，データ信号線に分類される．制御信号線はコントローラからデータパスの回路要素へ制御信号を伝達し，状態信号線はデータパスの回路要素からコントローラへ状態信号を伝達する．本論文では，テスト実行時にすべての制御信号線に対して任意の時点で任意の値が回路外部から設定可能であり，かつ，すべての状態信号線に対して任意の時点で任意の値が回路外部から観測可能であると仮定する．

データ信号線（以下，信号線）はデータ出力端子とデータ入力端子を接続する．データ出力端子には複数の信号線が接続できる（ファンアウト可能）が，データ入力端子に接続できる信号線は一つとする．

回路要素は外部入力，外部出力，レジスタ，マルチプレクサ，演算モジュール，観測モジュールに分類され，各種信号線が接続される端子をもつ．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の入力であるデータ入力端子（以下，入力端子）と，出力であるデータ出力端子（以下，出力端子）に分類される．すべてのデータ端子のビット幅は等しいとする．また，各入力端子は少なくとも一つの外部入力から到達可能であり，各出力端子からは少なくとも一つの外部出力へ到達可能とする．

便宜上，外部入力は一つの出力端子のみを，外部出力は一つの入力端子のみをもつ回路要素として扱う．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ち，制御端子の値によって，いずれかの入力端子の値を出力する．

演算モジュールは一つ又は二つの入力端子，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．演算モジュールは，制御端子と（2 入力演算モジュールの場合は）他方の入力端子に対して適切な定数を与えることで，各入力端子と出力端子の間が全単射になると仮定する．つまり，演算モジュール M が実現する関数族 $\mathcal{F}_M$ に対して，M が 1 入力（入力端子を x，出力端子を z とする）ならば，$z = f(x)$ が全単射となる関数 $f \in \mathcal{F}_M$ の存在を仮定する．また M が 2 入力（入力端子を x, y，出力端子を z とする）ならば，$z = f(x, a)$，$z = g(b, y)$ がそれぞれ全単射となる関数 $f, g \in \mathcal{F}_M$ と定数 a, b の存在を仮定する．以降，$f(x, a)$ を $f_{y=a}(x)$ と表し，$g(b, y)$ を $g_{x=b}(y)$ と表す．

観測モジュールは一つ又は二つの入力端子，一つの状態端子，たかだか一つの制御端子をもち，出力端子をもたない．観測モジュールは比較器等のモデルである．

データパス上の回路要素 e_1, e_k に対して e_1 を始点，e_k を終点とする経路 p を $p = (e_1, l_1, e_2, l_2, \cdots, l_{k-1}, e_k)$ と表し，$e_1 - e_k$ 経路と呼ぶ．ここで l_i は信号線で，回路要素 e_i の出力端子と回路要素 e_{i+1} の入力端子の一つを接続する．特にすべての e_i が相異なる場合，p を単純経路と呼ぶ．経路 p の始点と終点が同じ回路要素 e であり，e が始点と終点以外に現れず，e 以外の回路要素が p にたかだか 1 回だけ現れる場合，p をサイクルと呼ぶ．また，経路 p に現れるレジスタ数を p の順序深度と呼ぶ．任意の相異なる回路要素 e_1, e_2 に対して，e_1 を始点，e_2 を終点とする任意の相異なる単純経路の組を再収束経路の組と呼ぶ．

本論文で提案する DFT 手法が適用できるためにデータパス上の任意の経路が満たすべき条件は次のとおり．

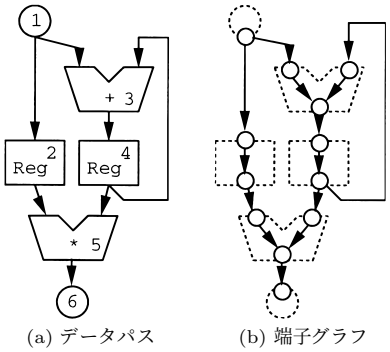
P1: 任意の演算/観測モジュール e_1, e_2 に対して，任意の $e_1 - e_2$ 経路はレジスタを含む．

P2: 任意の再収束経路の組に対して，一方の経路にのみ含まれるレジスタが存在する．

P3: 任意のサイクルにレジスタが存在する．

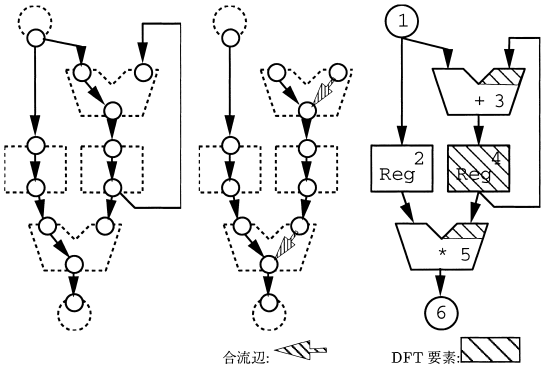
2.2 端子グラフ

以下では，データパスを端子グラフ $G = (\mathbf{V}, \mathbf{E})$ として表す．頂点集合 $\mathbf{V}$ はデータパス上のすべてのデータ端子からなる．また，有向辺集合 $\mathbf{E} = \mathbf{E}_1 \cup \mathbf{E}_2$ は信号線に対応する有向辺集合 $\mathbf{E}_1$ と，回路要素でのデータ端子間の入出力関係を表す有向辺集合 $\mathbf{E}_2$ からなる．つまり，入力端子 v_1 と出力端子 v_2 が同じ回路要素に



(a) データパス　(b) 端子グラフ

図 1　データパスと端子グラフ
Fig. 1　Data Path and corresponding port graph.



(a) 制御林　(b) 観測林　(c) データパス（DFT 後）

図 2　制御/観測林と DFT 後のデータパス
Fig. 2　Control/observe forest and datapath. (after DFT)

属する場合のみ，有向辺 (v_1, v_2) が $\mathbf{E}_2$ の要素となる．

図 1 にデータパスと対応する端子グラフの例を示す．以下では特に混乱のない限り，$\mathbf{V}$ の要素を入出力端子として扱い，$\mathbf{E}_1$ の要素を信号線として扱う．

3. テスト容易化設計アルゴリズム

3.1 基本方針

本論文ではデータパスに対して階層テスト生成を行った場合に完全故障検出効率が得られるための十分条件としてデータパスの強可検査性を考える．すべての回路要素に対して，1) 外部入力から任意の値を入力端子に伝達可能（強可制御性）かつ，2) 出力端子のとり得る任意の値が外部出力まで伝達可能（強可観測性）ならば，データパスが強可検査であるという．データパスが強可検査ならば，全回路要素に対してテストプランが存在することから，回路要素ごとに完全故障検出効率をもつテストベクタ集合を用いることで完全故障検出効率を実現する階層テストが可能となる．

提案するテスト容易化設計法はデータパスを強可検査とする．まず強可制御性を確保するために，入力端子 a ごとに外部入力から a へ任意の値を伝達する経路（制御経路）を決定する．次に強可観測性を確保するために，出力端子 b ごとに b から外部出力まで任意の値を伝達する経路（観測経路）を決定する．そして，これらの経路上で値が伝達できるように DFT 要素を追加する．DFT 要素は 2 入力演算モジュールでの全単射の実現に必要な定数を発生するマスク素子と経路間のタイミングに関する問題を解決するレジスタのホールド機能からなる．

3.2 制御林の生成

各入力端子に対し，制御経路を求める．一般に，外部入力からある入力端子への経路は複数存在するが，テスト実行時間を短縮するために，順序深度が最小の経路を制御経路とする．また，追加される DFT 要素を削減する目的で，制御経路の集合が外部入力を根とする林を構成するように制御経路を選ぶ．制御林は，データパス上で外部入力を始点として順序深度に基づいた最短経路林を構成することで $O(n)$ 時間で求められる．図 2 (a) に制御林の例を示す．

3.3 観測林の生成

各出力端子に対し，観測経路を求める．一般に，出力端子から外部出力への経路は複数存在するが，観測経路と制御経路の共有で DFT 要素が削減されるので，観測経路と制御経路の共有部分の極大化を目的として，制御林に含まれない観測経路上の辺が最小となるように選ぶ（制御林に含まれない観測林上の辺を合流辺と呼ぶ）．観測林は，データパス上で外部出力を始点として制御林に含まれない辺の数に基づいた最短経路林を構成することで $O(n)$ 時間で求められる．図 2 (b) に観測林の例を示す．

3.4 DFT 要素の追加

任意の制御/観測経路上を任意の値が伝達できるように，以下のように DFT 要素を追加する．

［マスク素子の挿入］ 回路要素に関する条件から，すべての回路要素で入出力間に全単射が実現できる．したがって，制御林上の任意の経路 p において，p 上のすべての辺 $l \in \mathbf{E}_2$ を全単射とする制御入力を回路要素に与えれば p の始点–終点間で任意の値が伝達できる．ただし，2 入力演算モジュール M（入力を x, y,

出力を z とする）に含まれる p 上の辺 $e^{(注2)}$を全単射とする場合，M が全単射 $z = f_{x=a}(y)$ を実現するように，M の制御入力に対して M の関数 $f \in \mathcal{F}_M$ を実現するための定数を与え，x に対して定数 a を与える必要がある．そこで定数 a を発生することが可能なマスク素子を x の直前に追加する．マスク素子は 1 ビットの制御入力をもつ 1 入力 1 出力の回路要素で，制御入力に従って定数又は入力値を出力する．マスク素子で定数 a を発生し，M に制御信号として f を与えれば $y - z$ 間が全単射になる．以下ではマスク素子と M を併せて一つの演算モジュール $M^\star$ とみなし，マスク素子と M の制御信号線を併せて $M^\star$ の制御信号線とみなす（$M^\star$ の入力端子を x', y'，出力端子を z' とすると，y' は y と，x' はマスク素子の入力と，z' は z と同一である）．また以下では $M^\star$ の関数で，$y' - z'$ 間を全単射とするものを $THRU$ と呼び，$x' - z'$ 間で全単射が実現できる関数を $THRU^\star$ と呼ぶ．

観測林上の合流辺を含まない部分経路 q は制御林に含まれるので，q 上でも値の伝達が可能である．また任意の合流辺 d（d を含む回路要素の入力端子を x, y とし，d の始点を x とする）に対し，d を全単射とするためには，y へある定数を入力しなければならないが，この定数は y への制御経路（以下では，d の補助経路と呼ぶ）から設定できる．よって制御林上の経路において任意の値の伝達が可能であれば，任意の観測経路においても任意の値が伝達される．

各経路上で値を伝達する方法を以下にまとめる．制御経路上ではレジスタに値をロードし，マルチプレクサには経路上の入力端子の値を出力する制御信号を与える．また演算モジュールには関数 $THRU$ を実現するために必要な定数を制御入力に与える．観測経路上ではレジスタに値ををロードし，マルチプレクサには経路上の入力端子の値を出力する制御信号を与える．また合流辺でない辺を経路上にもつ演算モジュールにはその関数 $THRU$ を実現するために必要な定数を制御入力に与える．一方で合流辺を経路上にもつ演算モジュールには関数 $THRU^\star$ を実現するために必要な定数を制御入力と他方の（経路上にない）データ入力に与える．このときデータ入力に与える定数は合流辺の補助経路を用いて外部入力から供給する．

［ホールド機能の追加］ テストベクタを印加する際やテストベクタに対する回路要素の応答を伝達する際にタイミング衝突が発生する場合がある．ここでタイミング衝突とは，ある時点で同一の頂点に異なる値を与えようとすることを指し，テストに必要な経路の集合が，順序深度の等しい経路からなる再収束経路の組を含む場合に再収束経路の始点で発生する．経路に関する条件（P2）から再収束経路の組にはどちらか一方にしか含まれないレジスタが存在する．このレジスタにホールド機能を与えることで再収束経路の組を構成する経路間でテスト実行時の順序深度が実質的に異なるようにする．タイミング衝突が発生するのは次のような場合である．

［場合 1］ 2 入力の回路要素 M（入力を x, y とする）のテスト時に，x の制御経路と y の制御経路が同一の始点と順序深度をもつ場合（図 3 (a)）．

［場合 2］ 2 入力の回路要素 M（入力を u, v，出力を w とし (v, w) を合流辺とする）の合流辺 (v, w) を含む観測経路 s を用いてテストベクタに対する応答を観測する場合を考える．制御経路（又は補助経路）の終点に v を終点とする s 上の部分経路を接続して得られる経路のうち，始点と順序深度が (v, w) の補助経路と等しいものが存在する場合（図 3 (b)）．

回路要素 M の入力端子 x に対して $depth(x)$ は x への制御経路の順序深度を，$ctrlPI(x)$ は x への制御経路の始点となる外部入力を表すこととする．また合流辺 (v, w) のうち $depth(v)$ と $depth(w)$ が異なるものを観測林から除いてできた林を T' とする．以下のいずれかの条件を満たすレジスタにホールド機能を与える．

- 任意の 2 入力の回路要素 M（入力端子を x, y，出力端子を z とし，(x, z) が制御林に含まれない

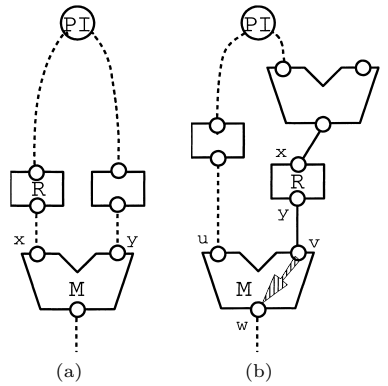


図 3　ホールド機能の追加
Fig. 3　Templates for hold-addition.

(注 2)：2 入力モジュール上の 2 本の辺のうち 1 本のみが制御林に含まれるので，一般性を失うことなく $e = (y, z)$ と仮定する．

ものとする）において，$depth(x) = depth(y)$ かつ $ctrlPI(x) = ctrlPI(y)$ である場合，制御林上で x に最も近い先祖のレジスタ R にホールド機能を与える（図 3 (a)）．

- 任意のレジスタ R（入力を x，出力を y とする）に対して以下の条件を満たす回路要素 M（入力を u, v，出力を w とし，(v, w) を合流辺とする）が存在する場合，R にホールド機能を与える（図 3 (b)）．
  - $depth(u) = depth(v)$
  - T' 上で順序深度が 0 の $y - v$ 経路が存在
  - $ctrlPI(u) = ctrlPI(x)$

このテスト容易化設計により，図 1 (a) のデータパスから図 2 (c) のデータパスが得られる．図 2 (c) の斜線部にホールド機能又はマスク素子が挿入される．

4. テストプラン生成アルゴリズム

本章では前章のテスト容易化設計で得られたデータパス上の任意の回路要素 M に対し，そのテストプランを生成するアルゴリズムを示す．ただし，アルゴリズムの入力として，制御林，観測林も与える．

M が 2 入力の回路要素で，M を通るサイクルが存在する場合，M の入力端子に対する制御経路（cp とする）の中間に M が現れることがある．M が故障している場合，cp を用いてテストベクタが正しく伝達できず，M の故障が検出されない可能性がある．そこで，このような cp が存在する場合，テストベクタごとに副テストを行ってから主テストを行う．ここで副テストとは外部入力から cp を通して外部出力までテストベクタを伝達することで cp 上をテストベクタが正しく伝達されるかを確認するテストであり，主テストとは cp を用いた M のテストを指す．副テストのためのテストプランは主テストのためのテストプランと同様に生成できるので，以下では主テストのためのテストプラン生成についてのみ述べる．

（1） 主経路集合の生成　まず，M の入力端子ごとに制御林から制御経路を求め，（もしあれば）M の出力端子に対して観測林から観測経路を求める．更に観測経路上の合流辺ごとに制御林から補助経路を求める．これらの経路集合を主経路集合と呼ぶ．

図 1 (a) の加算器に対する主経路集合を図 4 に示す．Cpath1，Cpath2 が制御経路，Opath が観測経路，Spath1 が補助経路を表す．

（2） 経路集合のタイミング調整　**3.4** で示したように，回路要素 M にテストベクタを外部入力から伝達するとき，あるいは，その応答を外部出力まで伝達するときに，タイミング衝突が発生する可能性がある（例えば図 4 において $Cpath1$ と $Spath1$ が同じタイミングで同じ外部入力を利用しようとしている）．このようなタイミング衝突を解消する手続きを以下に示す．

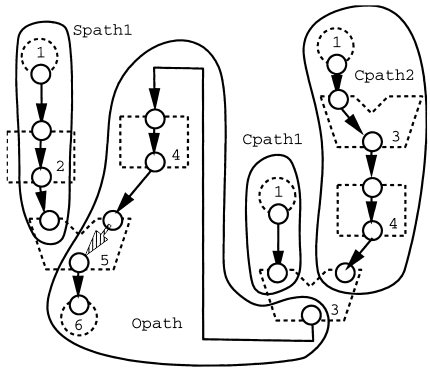


図 4　加算器 (3) の主経路集合
Fig. 4　Main path set for adder (3).

1. 制御経路の調整　　M が 2 入力の回路要素（入力を x, y，出力を z とし，一般性を失うことなく (x, z) が制御林に含まれないものと仮定する）である場合，x の制御経路を cp_x，y の制御経路を cp_y とする．cp_x と cp_y の順序深度と始点がともに等しい場合タイミング衝突が生じる．この場合，cp_x 上の終点に最も近いレジスタ r にホールド機能が追加されているので，r を 1 回ホールドすることでタイミング衝突を解消する．以降，追加されたホールドの回数を経路のレジスタ数に加えたものをその経路の順序深度とする．

2. 補助経路の調整　　制御経路の調整が終了した後に行う．観測経路上の合流辺を始点に近いものから順に $j_1, \cdots, j_k$ とし，各 j_i の補助経路を sp_i とする．また，j_0 を制御経路と観測経路を結ぶ M 上の辺で制御林に含まれるものとする．$sp_1, \cdots, sp_k$ を順に次の方法で調整する．これまでに調整が終了した制御/補助経路 l ごとに，観測経路の始点から j_i の始点までの観測経路上の部分経路を l の終点に連結して得られる経路の集合を $\mathbf{L}$ とし，$\mathbf{L}$ の要素のうち sp_i と始点が等しい経路の集合を $\mathbf{L}^\star$ とする．$\mathbf{L}^\star$ の中に sp_i と順序深度が等しい経路が存在する場合，j_i の始点から観測経路を逆にたどって最初に現れるレジスタ r_i を探す．1) r_i は j_{i-1}–j_i 間に存在すること，2) テスト容易化設計でホールド機能が r_i に追加されること，3) 制御経路は順序深度が最小であることより，r_i を 1 回ホールドする（これによって $\mathbf{L}$ に含まれるすべての

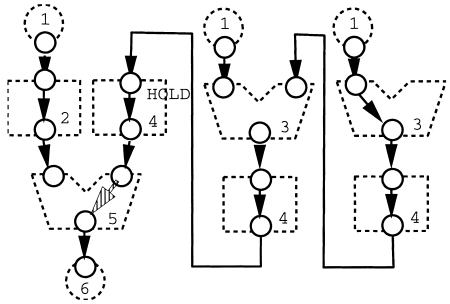


図 5　加算器 (3) の主経路集合（タイミング調整済み）
Fig. 5　Main path set for adder (3). (after timing regulation)

経路の順序深度が 1 増える）ことで sp_i とタイミング調整済である経路間のタイミング衝突が解消される．

ただし sp_1 の調整時に観測経路の先頭まで探索してもレジスタが存在しないことがある．この場合，経路集合に含まれる制御経路が 1 本であれば，制御経路を終点から逆方向に探索して最初に現れるレジスタを r とする．経路集合に含まれる制御経路が 2 本であれば，制御林に含まれない M 上の辺の始点を終点にもつ制御経路 cl が存在するので，そちらを終点から逆方向に探索し，最初に現れるレジスタを r とする．この場合，r のホールドで順序深度が増える経路は cl のみであるので，cl の順序深度が $\mathbf{L}^\star$ に含まれるどの経路とも異なるような最小の回数だけホールドする．

タイミング調整手続きは経路集合上の辺集合 $\mathbf{J} = (j_0, \cdots, j_i)$ の各要素ごとに，対応する補助経路と既にタイミング調整済みの経路との間でタイミング衝突が起きるかどうかを判定し，もし衝突が起こるのであれば，それを解消する．$\mathbf{J}$ の要素数はたかだか n であり，各要素について衝突判定の時間計算量は $O(n)$ である．したがってタイミング調整手続きの時間計算量は $O(n^2)$ となる．図 5 に図 4 をタイミング調整したものを示す．Spath1 と Cpath1 によって生じるタイミング衝突が Opath 上のレジスタ 4 をホールドすることによって解消されている．

（3） テストプランの生成　タイミング調整済みの主経路集合と（もしあれば）副経路集合からテストプランを作成する．経路集合において，観測経路の終点（M が観測モジュールの場合は M）から経路を逆にたどった際のレジスタ数を各回路要素に与える．与えられた順序深度が大きいものから順に **3.4** で述べた制御信号を与えることでテストプランが生成できる．表 1 に図 5 に対応するテストプランを示す．

表 1　テストプラン
Table 1　Testplan.

時刻	1		
回路要素 制御入力	1 （ベクタ入力）	3 $THRU$	4 ロード
時刻	2		
回路要素 制御入力	1 （ベクタ入力）	3 （テストベクタ）	4 ロード
時刻	3		
回路要素 制御入力	1 （ベクタ入力）	2 ロード	4 ホールド
時刻	4		
回路要素 制御入力	5 $THRU^\star$	6 （ベクタ観測）	

表 2　データパスの特性
Table 2　Character of circuits.

	GCD	4thIIR	Paulin
入力数	2	1	2
出力数	1	1	2
レジスタ数	3	12	7
MUX 数	3	3	8
演算モジュール数	2	5	7
観測モジュール数	3	0	0

5. 実験結果

完全故障検出効率を保証するテスト容易化設計法として，提案方式と完全スキャンを，実験によって比較した．3 種類のデータパス「GCD」,「4thIIR」,「Paulin」に対して，テスト生成時間，テスト容易化設計に基づくオーバヘッド，テスト系列長の 3 項目をデータパスのビット幅を 8，16，32 ビットと変化させて評価した．表 2 に各データパスの特性を示す．

［テスト生成時間］ 完全スキャン（以下 FS）でははじめにデータパス上の全レジスタを外部入出力で置き換えた組合せ回路から論理合成ツールを用いてゲートレベル回路を得た．次にゲートレベル回路に対してテスト生成ツールを用いて組合せテスト生成を行いテストベクタを得た．テスト生成に要した時間を完全スキャンのテスト生成時間 (T_{FS}) とした．

提案方式（以下 ST）では，はじめにレジスタ転送レベル回路に対してテスト容易化設計とテストプランの生成を行い，要した時間を T_1 とした．ここでマルチプレクサ（以下 MUX）と演算/観測モジュールをテストプラン生成の対象とした．次に論理合成ツールを用いて MUX と演算モジュール（テスト容易化設計済みのもの），観測モジュールごとにゲートレベル回路を得た後，個々のゲートレベル回路に対してテスト生成ツールを用いて組合せテスト生成を行いテストベク

タを得た．各ゲートレベル回路のテスト生成に要した時間を T_2 とし，提案方式のテスト生成時間 T_{ST} を $T_1 + \sum T_2$ とした．

なお，論理合成ツールとして Autologic II（Mentor Graphics），テスト生成ツールとして TestGen（Sunrise）を用い，SUN SPARCstation 20（SuperSPARC 75 MHz）相当の計算機上で実行した（以下の実験も同じ）．結果を表 3 に示す．

大規模な回路において提案方式のテスト生成時間は完全スキャンよりも小さい．これはテスト生成を回路全体に対してではなく回路要素ごとに行うことで，テスト生成ツールが取り扱う回路の規模が小さくなることが原因であると思われる．よってこの傾向はデータパスの大規模化とともにより顕著になると思われる．

［テスト系列長］ テスト生成で得られるテストベクタとテストプランから以下の方法でテスト系列長を求めた．ここでテスト系列長とは全テストベクタの入力と，その応答の観測に必要なクロック数である．結果を表 4 に示す．

- 提案方式（ST）

（回路要素のテストベクタ数×回路要素のテストプラン長）の全テスト対象回路要素に対する和

- 完全スキャン（FS）

（テストベクタ数）×（スキャンパス長 + 1）+（スキャンパス長）

ただし，（スキャンパス長）=（フリップフロップ数）

完全スキャン方式と提案方式のテスト系列長の差はデータパスのビット幅の増加に伴って急速に広がっている．よって，提案方式が大規模なデータパスでのテスト実行時間を大幅に削減することがわかる．

表 3　テスト生成時間（単位：秒）
Table 3　Test generation time. (sec.)

	GCD		4thIIR		Paulin	
	$T_1 = 0.2$		$T_1 = 0.2$		$T_1 = 0.2$	
bit	T_{ST}	T_{FS}	T_{ST}	T_{FS}	T_{ST}	T_{FS}
8	1.1	0.5	0.8	0.7	1.2	1.4
16	1.6	1.7	1.3	2.2	2.8	4.5
32	5.8	7.5	14.2	33.7	11.3	18.2

表 4　テスト系列長（単位：クロック）
Table 4　Test sequence length. (clocks)

	GCD		4thIIR		Paulin	
bit	ST	FS	ST	FS	ST	FS
8	254	1299	852	2715	1137	2792
16	470	5732	1392	8105	1503	8587
32	760	22794	2600	71224	2512	24524

［ハードウェアオーバヘッド］ テスト容易化設計前（org.），完全スキャン設計（FS）後，提案方式によるテスト容易化設計（ST）後の回路を論理合成し，ゲート数を比較した．ここで，レジスタにホールド機能を与えるために 1 bit 当り 3 ゲート，スキャン機能を与えるために 3 ゲート，ホールド機能とスキャン機能を同時に与えるために 4 ゲートを要した．結果を表 5 に示す．ただし，表中の%OH の項は次のとおり．

$$\frac{\text{DFT 後のゲート数} - \text{DFT 前のゲート数}}{\text{DFT 前のゲート数}} \times 100(\%)$$

提案方式は完全スキャン方式に比べて小さいハードウェアオーバヘッドで完全故障検出効率を保証している．4thIIR に対する提案方式でのオーバヘッドが他の回路に対する提案手法のそれに比べて大きい．これは 4thIIR に含まれる定数乗算器（1 入力 1 出力の演算モジュール）の入出力間においてマスク素子と制御入力の制御だけでは全単射が実現できないために，乗ずる定数として設計値のほかに定数 1 を選択できるようにして入出力間で全単射が実現できるように機能を拡張したためである．

提案方式では演算モジュールの入出力間で全単射が実現できることを前提としているが，実際の回路では定数乗算器やシフタなどがこの前提を満たさない．この場合入出力間で全単射が成り立つように回路要素の機能を拡張することで提案手法が適用できる．4thIIR では 5 個の演算モジュールのうち 3 個が定数演算器であり，すべての定数乗算器に対して全単射が実現できるように機能の拡張を行った．実験結果は完全スキャ

表 5　ハードウェアオーバヘッド
Table 5　Hardware overheads.

	GCD					
	org.		ST		FS	
bit	#FF	#gate	#gate	%OH	#gate	%OH
8	24	259	267	3.1	283	9.3
16	48	529	545	3.0	677	28.0
32	96	873	905	3.7	1161	33.0
	4thIIR					
	org.		ST		FS	
bit	#FF	#gate	#gate	%OH	#gate	%OH
8	96	350	510	45.7	636	81.7
16	192	1192	1512	26.8	1768	48.3
32	384	4316	4641	7.5	5468	26.7
	Paulin					
	org.		ST		FS	
bit	#FF	#gate	#gate	%OH	#gate	%OH
8	56	1152	1184	2.8	1240	7.6
16	112	3848	3912	1.7	4042	5.0
32	224	13848	13976	0.9	14200	2.5

ンと比較した場合，提案方式のハードウェアオーバヘッドはこのような機能の拡張を行っても十分小さいことを示唆している．

6. むすび

本論文では，与えられたレジスタ転送レベルデータパスに対して，完全故障検出効率を保証するテスト容易化設計法，及び，テスト容易化設計法を適用したデータパス上の回路要素に対するテストプラン生成法を示した．また，完全故障検出効率を保証するテスト容易化設計法として完全スキャンと提案方式の比較実験を 3 種類の回路に対して行うことで，ハードウェアオーバヘッド，テスト生成時間，テスト系列長の 3 点で，特に大規模な回路において提案方式が完全スキャン方式より優れていることを示した．

今後の課題としては，データパスに対する条件の緩和が挙げられる．具体的には，1）異なるビット幅をもつデータ信号線の混在する場合，2）3 個以上のデータ入力端子や 2 個以上のデータ出力端子をもつ回路要素が存在する場合，にも適用できるようにアルゴリズムを拡張することを考えている．

謝辞　本研究に際し，多くの貴重な意見をいただいた本学の情報論理学講座の諸氏に深く感謝します．本研究は一部,（株）半導体理工学研究センター（STARC）との共同研究，及び，文部省科学技術研究費補助金・基盤研究 B(2)（課題番号 09480054）の研究助成による．

文　献

[1] H. Fujiwara, “Logic testing and design for testability,” The MIT press, Cambridge, 1985.

[2] J. Lee and J.H. Patel, “Hierarchical test generation under architectural level functional constraints,” IEEE Trans. on CAD, vol.15, no.9, pp.1144–1151, Sept. 1996.

[3] R.B. Norwood and E.J. McCluskey, “Orthogonal scan: Low overhead scan for data paths,” Proc. 1996 Int. Test Conf., pp.659–668, 1996.

[4] R.B. Norwood and E.J. McCluskey, “High-level synthesis for orthogonal scan,” Proc. 15th VLSI Test Symp., pp.370–375, 1997.

[5] S. Bhatia and N.K. Jha, “Genesis: A behavioral synthesis system for hierarchical testability,” Proc. European Design and Test Conference, pp.272–276, 1994.

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

[7] I. Ghosh, A. Raghunath, and N.K. Jha, “A design for testability techinique for RTL circuits using control/dataflow extraction,” Proc. 1996 IEEE/ACM Int. Conf. on CAD, pp.329–336, 1996.

（平成 10 年 10 月 14 日受付，11 年 2 月 5 日再受付）

和田　弘樹　（学生員）

平 8 阪大・工・通信卒．平 10 奈良先端科学技術大学院大学博士前期課程了．現在奈良先端科学技術大学院大学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．

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，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．

Kewal K. Saluja

ルーキー大学（インド）電気工学科卒．1973 アイオワ大電気計算機工学科博士後期課程了．現在，ウィスコンシン大（マディソン）電気計算機工学科教授．南カリフォルニア大，アイオワ大，ルーキー大，広大，奈良先端大などで客員教授．テスト生成，テスト容易化設計，フォールトトレラント計算，VLSI 設計，計算機アーキテクチャなどの研究に従事．工博．Journal of Electronic Testing: Theory and Applications (JETTA) Associate Editor.

藤原　秀雄　（正員）

昭 44 阪大・工・電子卒．昭 46 同大大学院博士後期課程了．阪大工学部助手，明治大理工学部教授を経て，現在奈良先端科学技術大学院大学情報科学研究科教授．昭 56 ウォータールー大客員助教授．昭 59 マッギル大客員準教授．論理設計，高信頼設計，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”（The MIT Press）など．大川出版賞．工博．IEEE，情報処理学会各会員，IEEE Fellow，IEEE Golden Core Member.