ISSN 2186-5647

2-3

レジスタ転送レベルにおけるテスト並列化のための制御信号のドントケア割当て及びテストスケジューリング法

日大生産工(院)　○徳田　晴太　　日大生産工　細川　利典
京産大　　吉村　正義　　日大生産工　新井　雅之

1. まえがき

近年，超大規模集積回路(Very Large Scale Integrated Circuits：VLSI)のテストコスト増大に伴い，テストパターン数の削減が重要視されている．テストパターン数削減手法にはテスト圧縮法[1-2]やテストパターン数削減のためのテスト容易化設計手法(Design-for- Testability：DFT)[3-8]が提案されており，多数の故障を並列にテストするテスト並列化によってテストパターン数の削減を行っている．

しかしながら，テスト圧縮法[1-2]において回路構造が原因となり，テスト圧縮の効果が低い場合がある．また，ゲートレベルでDFT[6-8]を適用すると論理合成後の回路の変更により，論理合成で実行したタイミングの最適性を損失する可能性がある．以上の理由から，論理合成適用前の抽象度の高いレジスタ転送レベル(Register Transfer Level：RTL)でテスト並列化を考慮することが重要である．

RTLにおけるテストパターン数削減のためのDFT手法として文献[3,4]が提案されている．文献[4]では，ハードウェア要素のテスト並列度を高くするための制御信号の生成をコントローラの無効状態の状態遷移に設計している．そのため，テスト圧縮の効率が向上し，テストパターン数を平均33%削減できることが報告されている[4]．しかしながら，無効状態の状態遷移の設計によるコントローラ拡大のため，面積オーバヘッドが平均7.11%，最大30%になることが報告されている[4]．

文献[5]では，面積オーバヘッドを抑制するために，コントローラの有効状態の状態遷移における制御信号値のドントケアに着目して，データパス中のハードウェア要素のテストの並列度を向上させるためのドントケア割当て法が提案された．文献[5]で提案された制御信号のドントケア割当ては，擬似ブール最適化（Pseudo Boolean Optimization：PBO)問題として定式され，PBOソルバを用いて各状態遷移のドントケア割当てを求めた．小規模な高位合成ベンチマーク回路では，論理合成によりドントケア割当てが行われた回路と比較して，テストパターン数が平均8.1%削減されたことが報告されている．

しかしながら，中規模以上の高位合成ベンチマーク回路は，制御信号のドントケア数がPBOでは処理が困難になるほど増大する．したがって，ヒューリスティックドントケア割当てアルゴリズムが必要となる[10]．

文献[10]のアルゴリズムは，できる限り多数の演算器の並列テストが実現可能な状態遷移を選択し，実際にテストが可能となるように，制御信号の X 割当てを実行する．アルゴリズムは，できる限り少ない状態遷移ですべての演算器がテスト可能となるような X 割当てを実現している．しかしながら，データパスのテストパターン数を支配する演算器を少数の状態遷移を用いてテストするとは限らない．本論文では，できる限り多数の状態遷移で演算器のテストを可能とすることが，テストパターン数の削減に効果的であると考える．

本論文では，コントローラの状態遷移においてデータパスに供給される制御信号にドントケアを含んだまま，楽観構造的記号シミュレーション[10]及び悲観構造的記号シミュレーション[10]を実行し，各ハードウェア要素の楽観テスト回数と悲観テスト回数を求める．ここで，テスト可能と推定されたハードウェア要素は制御信号のドントケアに論理値が割当てられなければ，実際にテスト可能か否かが決定しないものも含まれる．楽観テスト回数は各ハードウェア要素のテスト回数の上界を見積り，悲観テスト回数は下界を見積る．楽観テスト回数が少ない演算器からテスト可能となるようドントケア割当てを実行する．ドントケア割当てアルゴリズムは，楽観/悲観構造的記号シミュレーションから得られる全演算器の楽観テスト回数と悲観テスト回数が一致するまで繰り返す．一致したら，すべてのハードウェア要素がテスト可能と推定されるための最小個の推定テストパターン数を PBO によって求め，各状態遷移で生成するテストパターン数を決定する．

第2章では，ハードウェア要素並列テストの諸定義について述べる．第3章では，テスト可能なハードウェア要素について述べる．第4章では，ドントケア割当てアルゴリズムについて述べる．第5章では，各状態遷移で生成するテストパターン数の算出法について述べる．第6章では，実験結果について述べ，最後に結論と今後の課題について述べる．

2. ハードウェア要素並列テストの諸定義

2.1 ハードウェア要素

データパス回路内の信号線，演算器，マルチプレクサ(MUX)，MUX の制御信号線，レジスタ(REG)，レジスタの制御信号線をハードウェア要素[4]と呼ぶ．特に MUX や REG の制御信号線に関しては，定義された故障もハードウェア要素と呼ぶ．

図 1 にデータパスの部分回路の例を示す．図 1 において R0~R2 は REG，+と*はそれぞれ加算器と乗算器，M1 は MUX，a，b，f は外部入力，y は外部出力を示す．MUX 内の数字は MUX の制御信号値に対応した入力番号を示す．R1 はホールド機能付き REG である．データパス内の MUX はコントローラが出力する制御信号値で制御される．同様に，データパス内のホールド機能付きレジスタはコントローラが出力する制御信号値によって，ロードモードもしくはホールドモードに制御される．

X-Filling of Control Signals and Test Scheduling Methods for Concurrent Testing at Register Transfer Level

Haruta TOKUTA, Toshinori HOSOKAWA, Masayoshi YOSHIMURA and Masayuki ARAI

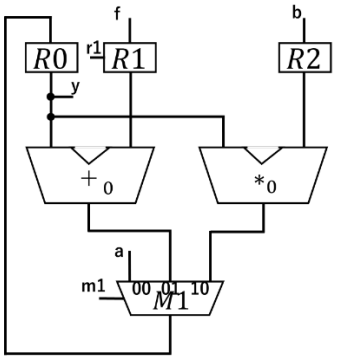

図 1. RTL データパスの部分回路

2.2 ハードウェア要素の並列テストの実現

データパスのテストでは，ある状態遷移においてテスト対象ハードウェア要素に対するテスト経路を活性化する必要があり，それを実現するにはコントローラから MUX や REG に適切な制御信号を供給する必要がある[4]．本論文では，テスト経路の活性処理について，構造的記号シミュレーション[5, 10]を使用する．

3. テスト可能なハードウェア要素

3.1 構造的記号シミュレーション

本論文では，回路はフルスキャン設計されていることを前提とする．構造的記号シミュレーションとは，各状態遷移で可制御性，可観測性からデータパスのハードウェア要素のテスト可能性を求める手法である[5, 10]．データパスとコントローラを入力とし，テスト可能表を出力とする．テスト可能表については，3.1 節で詳細に説明する．

構造的記号シミュレーションには，0 と 1 の 2 値[5]，0 と 1 と X の 3 値[10]に対応したものがある．2 値に対応した構造的記号シミュレーションは，制御信号の X 値に論理値 0 または 1 を割当てることを仮定しなければならない．3 値に対応した構造的記号シミュレーションは，制御信号 X に論理値 0 または 1 を割当てずにテスト可能性を求めることができる．しかしながら，制御信号値 X にどのような論理値を割当てることを仮定するかによって求められるテスト可能なハードウェア要素が変化する．本論文では，3 値の制御信号値に対応した構造的記号シミュレーション[10]を用いる．データパスにおける状態遷移 ST で並列テスト可能なハードウェア要素について，以下のように定義する．

（定義 1：楽観構造的記号シミュレーション）

楽観構造的記号シミュレーションは，構造的記号シミュレーション[5]の処理とほぼ同じである．ただし，制御信号値 X をあるハードウェア要素がテスト可能となるような論理値割当てがされていることを仮定する．楽観構造的記号シミュレーションは，テスト可能なハードウェア要素を最大限に見積る．

（定義 2：悲観構造的記号シミュレーション）

悲観構造的記号シミュレーションは，構造的記号シミュレーション[5]の処理とほぼ同じである．ただし，制御信号値 X をあるハードウェア要素がテスト不能となるような論理値割当てがされていることを仮定する．悲観構造的記号シミュレーションは，テスト可能なハードウェア要素を最小限に見積る．

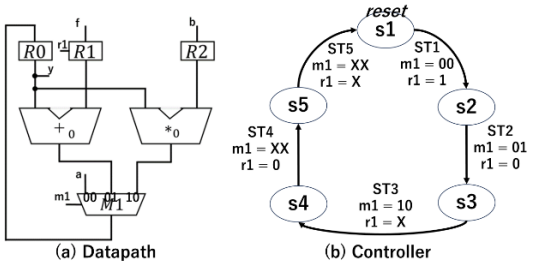

図 2. RTL 回路例

表 1. テスト可能表

(a) 楽観テスト可能表

	$+_0$	$*_0$	M1in0	M1in1	M1in2	M1out	M1_1b_sa0	M1_1b_sa1	M1_2b_sa0	M1_2b_sa1	R0	R1	R1_1b_sa0	R1_1b_sa1	R2
ST1	×	×	○	×	×	○	×	○	×	○	○	×	○	×	×
ST2	○	×	×	○	×	○	○	×	×	×	○	○	×	○	×
ST3	×	○	×	×	○	○	×	×	○	×	○	○	○	○	○
ST4	○	○	○	○	○	○	○	○	○	○	○	○	×	○	○
ST5	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

(b) 悲観テスト可能表

	$+_0$	$*_0$	M1in0	M1in1	M1in2	M1out	M1_1b_sa0	M1_1b_sa1	M1_2b_sa0	M1_2b_sa1	R0	R1	R1_1b_sa0	R1_1b_sa1	R2
ST1	×	×	○	×	×	○	×	○	×	○	○	×	○	×	×
ST2	○	×	×	○	×	○	○	×	×	×	○	○	×	○	×
ST3	×	○	×	×	○	○	×	×	○	×	○	×	×	×	○
ST4	×	×	×	×	×	○	×	×	×	×	○	○	×	○	×
ST5	×	×	×	×	×	○	×	×	×	×	○	×	×	×	×

3.2 テスト可能表

テスト可能表とは，構造的記号シミュレーションを実行後，各状態遷移での各ハードウェア要素のテスト可能性を示す表である．楽観構造的記号シミュレーション後は楽観テスト可能表，悲観構造的記号シミュレーション後は悲観テスト可能表が生成される．図 2 に RTL 回路例を示す．図 2(a)は，図 1 のデータパスを示しており，図 2(b)はコントローラを示している．表 1 に図 2 の回路に対して楽観/悲観構造的記号シミュレーションを実行して生成した楽観/悲観テスト可能表を示す．

表 1(a)は，図 2 で示した回路に対して楽観構造的記号シミュレーションを実行した後に得られる楽観テスト可能表である．表 1(b)は，図 2 で示した回路に対して悲観構造的記号シミュレーションを実行した後に得られる悲観テスト可能表である．

表 1(a), (b)において，1 行目に図 2(a)で示したデータパスの各ハードウェア要素を示し，1 列目に状態遷移を示している．2 行 2 列目から，各ハードウェア要素がテスト可能であるかどうかを示す．並列テストが可能であれば「○」，そうでなければ「×」とする．例えば，表 1(a)における 2 行目 2 列目は，演算器$+_0$がテスト不能であることを示している．ここで，表 1(a), (b)の 6 行目の状態遷移 ST5 に着目すると，楽観テスト可能表は全ハードウェア要素がテスト可能であるのに対し，悲観テスト可能表は M1out，R0 のみテスト可能である．楽観テスト可能表ではテスト可能であるのに対し，悲観テスト可能表ではテスト不能となるハードウェア要素があることから，楽観構造的記号シミュレーションと悲観構造的記号シミュレーションで求めるテスト可能なハードウェア要素に差異があることがわかる．また，楽観テスト可能表からドントケア割当てによってはテスト可能となるかもしれないハードウェア要素を特定することができ，悲観テスト可能表からはドントケア割当てをしなくともテスト可能なハードウェア要素を特定することができる．

```
Algorithm Xfill
Input    :C , DP, TPL
Output   :MC
1.     MC = C;
2.     flag = false;
3.       While ( flag ≠ true )
4.             OTT = Opt_SSS(MC, DP) ;
5.             PTT = Pes_SSS(MC, DP) ;
6.             OPT = SORT(OTT, PTT, TPL) ;
7.             EOP = OP_selection(OPT) ;
8.             if (EOP ! = ∅ ) then
9.                   MC = Xfill(MC, DP, EOP) ;
10.            else
11.                  flag = true;
12.            endif
13.      endwhile
14.       return MC ;
15.    End
```

図 3. ドントケア割当てアルゴリズム

4. ドントケア割当てアルゴリズム

本アルゴリズムは，各ハードウェア要素のテスト並列度を向上させるためのドントケア割当てを目的とする．図 3 に，本論文で提案するヒューリスティックなドントケア割当てアルゴリズムを示す．

本アルゴリズムは，オリジナルコントローラ C，データパス DP，各ハードウェア要素の単体テスト結果情報を持つテストパターンライブラリ TPL を入力とし，ドントケア割当てが適用されたコントローラ MC を出力とする．

初めに，MC を C で初期化する(1 行目)．ドントケア割当て終了判定フラグ flag を false で初期化する(2 行目)．flag が true となるまで 4~12 行目の処理を繰り返す(3 行目)．ドントケア割当て対象コントローラとデータパスを入力とし，楽観テスト可能表 OTT と悲観テスト可能表 PTT を生成する(4, 5 行目)．楽観テスト可能表，悲観テスト可能表，テストパターンライブラリを入力とし，各演算器の楽観テスト回数，悲観テスト回数，見積りテストパターン数を格納した演算器テーブル OPT を生成する(6 行目)．演算器テーブル OPT を入力とし，EOP には楽観テスト回数が最も少ない演算器とドントケア割当てを実行する状態遷移が格納される．状態遷移の選択は，処理の対象とする演算器が楽観でテスト可能であるが悲観でテスト不能と判定される状態遷移を 1 つ選択する．ここで，全演算器の楽観テスト回数と悲観テスト回数が一致したら EOP には∅が格納される(7 行目)．7 行目で，処理対象の演算器が選択されたら 9 行目へ進み，それ以外の場合，11 行目に進む(8 行目)．ドントケア割当て対象コントローラ MC，データパス DP，処理対象の演算器 EOP を入力とし，EOP に格納された状態遷移で処理対象の演算器を実際にテスト可能とするようにドントケア割当てを行い，ドントケア割当てされたコントローラ MC を出力する(9 行目)．全演算器の楽観テスト回数と悲観テスト回数が一致したため，ドントケア割当てアルゴリズム終了判定フラグ flag を true にする(11 行目)．全演算器を可能な限りテスト可能とすることでテスト並列度を向上させるようなドントケア割当てが適用されたコントローラ MC を返却する．ただし，演算器以外のハードウェア要素に対して楽観テスト回数に悲観テスト回数を近づけるためのドントケア割当てを実行していない．したがって，MC の各状態遷移には X が残ることもある(14 行目)．

5. 各状態遷移で生成するテストパターン数

本章では，各状態遷移で生成するテストパターン数の算出法について述べる．前提条件として，各状態遷移でテスト可能なハードウェア要素は全て並列テストが可能であるものとする．したがって，各状態遷移で生成するテストパターン数は，各状態遷移でテスト可能な全ハードウェア要素を並列にテストすることができるものと仮定する．

5.1 擬似ブール最適化問題(PBO)

PBO とは，制約式を充足し，最適化関数を最小化するように，制約式で使用されている変数の論理値割当てを求める問題である．PBO は，一般的に 0-1 整数計画問題と同一の問題を解くことができる．

本論文では，ドントケア割当てが適用されたコントローラの悲観テスト可能表を PBO の変数として制約式と最適化関数を定式化することで，各状態遷移で生成するテストパターン数を求め，データパスの推定テストパターン数最小化問題を解く．

5.2 制約式と最適化関数

PBO ソルバの入力は 3.1 節の表 1(b)で説明したデータパスのハードウェア要素の悲観テスト可能表の情報である．出力は，各状態遷移で生成するテストパターン数である．最適化関数は，選択された状態遷移で生成するテストパターン数の総和である．以下に，PBO ソルバへの入力とする制約式及び最適化関数について説明する．

$$\prod_{j=1}^{N_m}\sum_{s=1}^{N_s} X_{sj} \geq 1 \quad (1)$$

式(1)は，全ハードウェアが少なくとも 1 つの状態遷移で並列テスト可能であるという制約式である．X_{sj}は，状態遷移sにおいてハードウェア要素jがテスト可能か否かを判断する変数である．X_{sj}が 1 の時は並列テスト可能(テスト可能表の「○」)を表し，0 の時は並列テスト不能(テスト可能表の「×」)を表す．N_mはハードウェア要素数，N_sは状態遷移数である．

$$\sum_{s=1}^{N_s} X_{sj} \times A_s \geq TP_j \quad (\forall_j) \quad (2)$$

式(2)は，ハードウェア要素jに対して状態遷移sで生成するテストパターン数の総和が見積りテストパターン数以上となる制約式である．TP_jはハードウェア要素jの見積りテストパターン数である．A_sは状態遷移sで生成するテストパターン数である．

$$min \sum_{s=1}^{N_s} A_s \quad (3)$$

式(3)は，各状態遷移で生成するテストパターン数の総和を最小化する最適化関数である．

PBO ソルバは，各変数に対して 0, 1 のみ割当てるため，前処理としてA_sのバイナリエンコーディングが必要となる．式(4)は，エンコーディングの式である．A_sは，8bit の 2 進数でエンコーディング可能である．また，比較器の見積りテストパターン数は 68 パターンであり，2 進数で表現するためには 8bit の変数が必要であるため，エンコーディングに用いる変数を$Z_{s7} \sim Z_{s0}(\in \{0,1\})$とする．前提として，ある状態遷移におけるテスト可能なハードウェア要素は並列にテストされると仮定しているため，各状態遷移で生成するテストパターン数の上限値は，その

状態遷移でテスト可能なハードウェア要素の中で最も多い見積りテストパターン数である．

$$A_s=Z_{s7}\times 2^7+Z_{s6}\times 2^6+Z_{s5}\times 2^5+Z_{s4}\times 2^4+Z_{s3}\times 2^3+Z_{s2}\times 2^2+Z_{s1}\times 2^1+Z_{s0}\times 2^0 \quad (4)$$

制約式(1)，(2)と最適化関数(3)，及びエンコーディング式(4)と変数X_{sj}をまとめて PBO ソルバに入力し解を求めると，PBO によるテストスケジューリングを行うことによって各状態遷移で生成するテストパターン数を決定する．

6. 実験結果

本実験では，信号線のビット幅が 32bit の RTL ベンチマーク回路 ex2,ex4,kim,maha,sehwa を対象とする．オリジナルコントローラ org，提案手法適用コントローラ pro，従来手法[10]cnv を適用したコントローラをそれぞれデータパスと論理合成し，生成された論理回路に対して評価を行う．

論理合成ツールは Synopsys の Design Compiler，テスト生成ツールは，有効状態のみ使用してテスト生成を行う内製の ATPG ツールを使用する．また，本実験ではコントローラの故障は対象故障から除外し，データパスの故障のみ仮定する．

図 4 に，各ベンチマーク回路を使った各手法のテストパターン数の棒グラフを示す．提案手法 pro と org を比較すると全回路においてテストパターン数を削減しており，平均 10.59%削減している．次に，pro と従来手法[10]cnv を比較すると，全回路においてテストパターン数を削減しており，平均約 10.39%削減している．しかしながら，面積オーバヘッドに関しては，org と pro を比較すると平均約 0.66%増加し，cnv と pro を比較すると平均約 0.57%の増加であった．テストパターン数を削減するためにテスト可能な演算器が増加させたことで，面積を最小化とするドントケア割当てがなされていないことがわかる．

7. まとめ

本論文では，レジスタ転送レベルにおけるテスト並列化のための制御信号のドントケア割当て及びテストスケジューリング法を提案した．

文献[10]と比較し，テストパターン数削減率は平均約 10.39%削減し，面積オーバヘッドは平均約 0.57%の増加であった．

今後の課題として，テストスケジューリング法に基づくテスト生成手法を提案し，更にテストパターン数を削減することが挙げられる．

文　献

[1] S. Kajihara, I. Pomeranz, K. Kinoshi “Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits,” IEEE Transaction s on Computer Aided Design of Integrated Circuits and Systems, Vol.14, Issue12, pp.1496-1504, Dec.1995.

[2] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy “On Compaction Test Sets by Addition and Removal of Test Vectors,” VLSI Test Symposium, 1994. Proceedings.12th IEEE, pp.202-207, Cherry Hill, NJ, The USA, Apr.1994.

[3] T. Hosokawa, S. Takeda, H. Yamazaki, M. Yoshimura, “Controller Augmentation and Test Point Insertion at RTL for Concurrent Operational Unit Testing,” IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS), pp.17-20, Thessaloniki, Greece, July.2017.

[4] T. Hosokawa, S. Takeda, H. Yamazaki and M. Yoshimura, “A Test Register Assignment Method Based on Controller Augmentation to Reduce the Number of Test Patterns,” Proc. Int. Symp. on On-Line Testing and Robust System Design, pp.228-231, 2018.

[5] 徐浩豊, 細川利典, 山崎紘史, 新井雅之, 吉村正義 “論理故障テスト並列テスト化のための制御信号のドントケア割当て法” 信学技報, CPSY2021-56, DC2021-90, pp.67-72, 2022 年 3 月.

[6] M. J. Geuzebroek, J. Th. van der Linden, and A. J. van de Goor, “Test Point Insertion for Compact Test Set s,” Test Conference, 2000. Proceedings. International, pp.292-301, Atlantic City NJ, The USA, Oct 2000.

[7] S. Remersaro, J. Rajski, T. Rinderknecht, Sudhakar M. Reddy, I. Pomeranz, “ATPG Heuristics Dependant Observation Point Insertion for Enhanced Compaction and Data Volume Reduction,” IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, pp.385-393, Oct.2008.

[8] M. Yoshimura, T. Hosokawa, and M. Ohta, “A Test Point Insertion Method to Reduce the Number of Test Patterns,” IEEE the 11th Asian Test Symposium (ATS 2002), pp.298-304, Nov. 2002.

[9] 土渕航平, 細川利典, 山崎浩二, “レジスタ転送レベル回路における故障診断容易化のためのコントローラの制御信号のドントケア割当て法,” CPSY2020-62, pp.73-78, Mar.2021.

[10] 徐浩豊, 細川利典, 吉村正義, 新井雅之 “並列テストのためのコントローラの制御信号のドントケア割当てアルゴリズム” 信学技報, vol. 122, no. 205, DC2022-24, pp. 37-42, 2022 年 10 月.

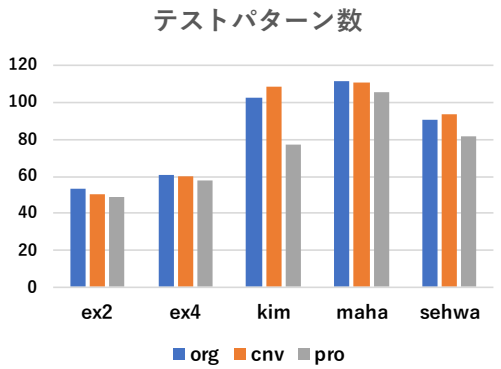

図 4. 実験結果