ISSN 2186-5647

2-31

テスト並列化のためのテストスケジューリングにおける制御信号のドントケア割当て手法

日大生産工　○徳田　晴太　　日大生産工(院)　豊岡　雄大
日大生産工　細川　利典　　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[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では，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なドントケア割当てアルゴリズムが必要となる．

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，楽観的な構造的記号シミュレーション[10]を実行し，テスト可能と推定されるハードウェア要素を特定する．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．すべてのハードウェア要素が完全にテスト可能と推定されるための最小個の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，RTLデータパスのハードウェア要素の並列テストの諸定義について述べ，第3章では構造的記号シミュレーションについて述べ，第4章でPBOを用いた見積りテストパターン数算出法について述べ，第5章でテスト並列化のためのテストスケジューリングにおける制御信号のドントケア割当て法について述べ，第6章で予備実験結果を示す．最後に結論と今後の課題について述べる．

2. RTL データパスのハードウェア要素の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 やレジスタ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* はレジスタ，＋と －はそれぞれ加算器と減算器，*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．m1 は M1 に制御信号値を供給する制御信号線である．R1 はホールド機能付きレジスタである．r1 は R1 に制御信号値を供給する制御信号線である．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．

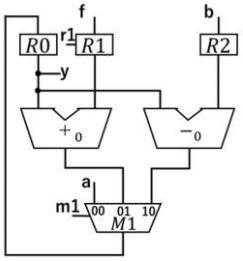

図 1. RTL データパスの部分回路

A Don’t Care Filling Method of Control Signals on Test Scheduling for Concurrent Testing

Haruta TOKUTA, Yudai TOYOOKA, Toshinori HOSOKAWA and Masayuki ARAI

表1. ハードウェアテスト可能表

ADD_in0	ADD_in0	ADD_out	SUB_in0	SUB_in1	SUB_out	M1_in0	M1_in1
○	○	○	×	×	×	×	○
M1_in2	M1_out	M1_1'b_sa0	M1_1'b_sa1	M1_2'b_sa0	M1_2'b_sa1	R0_in	R0_out
×	○	×	○	○	×	○	○
R1_in	R1_out	R2_in	R2_out	a	b	f	y
○	○	○	×	×	○	○	○

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 やレジスタに適切な制御信号を供給する必要がある．本論文ではテスト経路の活性化処理について，構造的記号シミュレーション[10]を使用する．

3. 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．また文献[5]で用いた構造的記号シミュレーションは制御信号値が 0 又は 1 の 2 値であったが，本論文では，制御信号値を 0, 1, X の 3 値に拡張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：楽観的な構造的記号シミュレーション）

楽観的な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り，あるハードウェア要素がテスト可能となるように X の論理値が割当てされていると仮定する．各状態遷移における X の論理値割当てにより，テスト可能なハードウェア要素を最大限に見積る．

3.1 ハードウェアテスト可能表

表1にハードウェアテスト可能表の例を示す．ハードウェアテスト可能表とは，ある状態遷移で楽観的な構造的記号シミュレーションを実行した後，テスト可能なハードウェア要素を示した表である．表1において1, 3, 5行目に各ハードウェア要素の名前を示し，2, 4, 6行目に各ハードウェア要素がテスト可能であるか否かを示す．テスト可能であれば「○」，テスト不能であれば「×」とする．例えば，1行目1列目の「ADD_in0」は2行目1列目が「○」であることから，加算器の入力信号線0がテスト可能であることを示している．

4. PBO を用いた見積りテストパターン数算出法

4.1 擬似ブール最適化問題

PBO とは，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．

各状態遷移のハードウェアテスト可能表を 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の算出を行い，データパスの見積りテストパターン数最小化問題を解く．

4.2 見積りテストパターン数

見積りテストパターン数とはデータパスのハードウェア要素を単体で完全にテストするために必要なテストパターン数である．表1の各ハードウェア要素の見積りテストパターン数を事前に計算する必要があるため，各ハードウェア要素に対してATPGツールを用いて単体でテスト生成を実行し，得られた必要なテストパターン数を見積りテストパターン数とする．

PBOソルバの入力は3.1章で説明したデータパスのハードウェア要素の楽観的なテスト可能表の情報である．出力は各状態遷移の見積り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の見積りテストパターン数の総和である．以下にPB0ソルバへ入力とする制約式及び最適化関数について説明する．

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

式(1)は全ハードウェア要素が少なくとも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}とは導入する変数である．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．1の時は並列テスト可能(テスト可能表の「○」)を表す，0の時は並列テスト不能(テスト可能表の「×」)である．N_mはハードウェア要素数，N_sは状態遷移数である．

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

式(2)はハードウェア要素jに対して，状態遷移sで生成する必要最小限のテストパターン数の総和の制約式である．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．

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

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にする最適化関数である． PB0ソルバは各変数に対して1，0しか割当てることができないので，前処理として，A_sのバイナリーエンコーディングが必要である．式(4)はエンコーディングの式である．A_sは8ビットの2進数でエンコーディング可能である．その変数をZ_{s7}～Z_{s0}（0または1）とする．前提条件は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いう前提であるので，各状態遷移で生成されるテストパターン数の上限値は，テスト可能なハードウェア要素の中で最大のTP_jとなる．

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

制約式(1)，(2)，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，PB0ソルバに入力して解を求めると，各状態遷移で生成されるテストパターン数が求まり，必要なテストパターン数が推定される．

4.3 PBO ソルバの解から推定されるテストパターン数

前提条件としては，各状態遷移でテスト可能なハードウェア要素はすべて並列テストが可能である．図 2 にコントローラ例を示す．表 2 にハードウェア要素をテストするのに必要な見積りテストパターン数とその状態遷移についてまとめた表を示す．これは，各ハードウェア要素が完全にテストされるために必要なテ

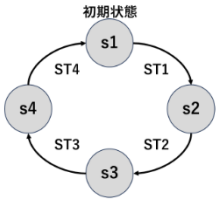

図 2. コントローラ例

表 2. ハードウェア要素をテストしている状態遷移とテストするのに必要な見積りテストパターン数

ハードウェア要素	見積りテストパターン数	テストされている状態遷移
H1	30	ST1, ST2
H2	20	ST2, ST3
H3	10	ST2
H4	5	ST2, ST4
H5	15	ST1, ST4

表 3. PBO ソルバが出力した各状態遷移で生成するテストパターン数

状態遷移	生成するテストパターン数
ST1	20
ST2	10
ST3	10
ST4	0

ストパターン数と，どの状態遷移でテスト可能であるかを示す情報である．例えば，ハードウェア要素 H1 を完全にテストするために必要なテストパターン数は 30 個である．H1 は状態遷移 ST1, ST2 でテストされているため，状態遷移 ST1 と ST2 のテストパターン数の総和は必ず 30 以上となる．

全ハードウェア要素(H1~H5)の見積りテストパターン数から PBO ソルバは，各状態遷移でのテストパターン数の総和が最小となるような各状態遷移で生成するテストパターン数を出力する．表 3 に PBO ソルバが出力した各状態遷移で生成するテストパターン数を示す．表 2 の情報から，PBO ソルバは状態遷移 ST1 のテストパターン数が 20，ST2 は 10，ST3 は 10 と，ST4 は 0 と出力する．ここで ST4 は，他の状態遷移 ST1~ST3 までで全ハードウェア要素をテストすることができたため，「0」を出力している．

5. テスト並列化のためのテストスケジューリングにおける制御信号のドントケア割当て手法

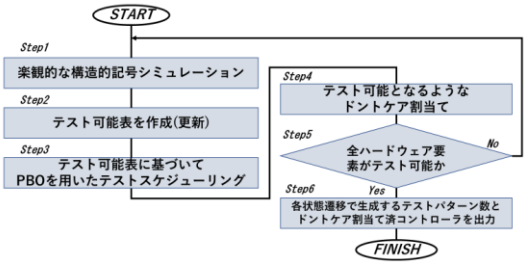

図 3. 本提案手法のフローチャート

図 3 に本提案手法のフローチャートを示す．次に，各ステップで行う処理について説明する．

(STEP1)
楽観的な構造的記号シミュレーションを行い，テスト可能となるハードウェア要素を最大限に見積る．

(STEP2)
テスト可能であるハードウェア要素を表として示す．

(STEP3)
STEP2 で示したテスト可能表を用いて 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の算出を行う．

(STEP4)
テスト可能であると見積ったハードウェア要素を実際にテストするよう制御信号 X に適切な論理値 0，1 のどちらかを割当てる．

(STEP5)
STEP4 の結果，すべてのハードウェア要素がテスト可能であれば，STEP6 へ処理を進める．不能であれば，STEP4 で行ったドントケア割当てをしたコントローラを用いて STEP1 へ処理を進める．

(STEP6)
各状態遷移で生成するテストパターン数とドントケア割当て済コントローラを出力する．

6. 予備実験結果

本章では，実験結果について説明する．本実験では，図2のフローチャートSTEP3までの処理で実験を行う．まず，コントローラの制御信号のドントケアにランダムに論理値を割当てたものを10,000個作成する．各コントローラに対応する構造的記号シミュレーションを実行し，各状態遷移において，正確にテスト可能なハードウェア要素を求め，PBOで生成したテストパターン数で各ハードウェア要素が実際にどの程度テストされるかを評価する．

6.1 テスト可能率

実験で使用するテスト可能率について説明する．テスト可能率とは，制御信号のドントケアに論理値を割当てたコントローラでそのハードウェア要素の見積りテストパターン数に対して，制御信号の論理値割当てによって，そのハードウェア要素を実際にテストできたテストパターン数の割合のことである．テスト可能率の算出式を式(5)に示す．

$$\textit{ハードウェア要素H のテスト可能率}(\%) = \frac{\textit{実際にH をテストしたテストパターン数}}{\textit{見積りテストパターン数}} \times 100 \quad (5)$$

ただし(5)を計算した結果，100%を超える場合は，(5)の値を 100%とする．

表 4 に，テスト可能率について示す．表 4 の 2 行目から 4 行目のハードウェア要素 H1~H3 がコントローラ 1 でのハードウェア要素である．コントローラ 1 のハードウェア要素 H1 をテストするのに必要なテストパターン数は 15，実際にテストしたテストパターン数は 30 であり，実際にテストしたテストパターン数がテストするのに必要なテストパターン数以上となるため 100%と評価する．

次に，表 4 の 4 行目のハードウェア要素 H3 において，ハードウェア要素 H3 をテストするのに必要なテ

表 4. テスト可能率

コントローラ名	ハードウェア要素	見積りテストパターン数	テストしたテストパターン数	テスト可能率
コントローラ1	H1	15	30	100%
	H2	65	65	100%
	H3	65	52	80%
コントローラ2	H1	15	15	100%
	H2	65	65	100%
	H3	65	65	100%

ストパターン数は 65，実際にテストしたテストパターン数は 52 であり，実際にテストしたテストパターン数がテストするのに必要なテストパターン数未満となり，評価は 80%となる．

6.2 予備実験結果考察

テスト可能率は，各ハードウェア要素で生成されたテストパターン数の平均とする．表5に，ex2回路のハードウェア要素すべてのテスト可能率を示す．

表 5 から，乗算器 0 以外すべてテスト可能率が 100%となり，どのコントローラにランダムに X 割当てしても乗算器 0 以外のハードウェア要素はテスト可能となるような割当てがされていることがわかる．したがって乗算器 0 をテストするために適切な論理値を割当てることができないコントローラが一部テ存在していたことがわかる．

7. まとめ

本論文では，PBOソルバを用いてex2回路のハードウェア要素のテスト可能率を算出した．楽観的な構造的記号シミュレーションではテストできていた乗算器が実際に0か1にランダムで論理値割当てするとテストできなくなることがわかった．

今後の課題として，図3のSTEP3以降の処理を実装することが挙げられる．まず，STEP4で実際にハードウェア要素がテストできるようドントケア割当てを行い，STEP5で全ハードウェア要素をテストしたかどうかの判定を行い，STEP6で各状態遷移で生成するテストパターン数とドントケア割当て済コントローラを出力するよう実装する．

表 5.　ex2 テスト可能率

−0	*1	*0	m9_in0	m9_in1	m9_in2	m9_out	m9 1b_0_sa	m9 1b_1_sa	m9 2b_0_sa
100%	100%	86%	100%	100%	100%	100%	100%	100%	100%
m9 2b_1_sa	m8_in0	m8_in1	m8_in2	m8_out	m8 1b_0_sa	m8 1b_1_sa	m8 2b_0_sa	m8 2b_1_sa	m7_in0
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
m7_in1	m7_out	m7 1b_0_sa	m7 1b_1_sa	m6_in0	m6_in1	m6_out	m6 1b_0_sa	m6 1b_1_sa	m5_in0
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
m5_in1	m5_out	m5 1b_0_sa	m5 1b_1_sa	m4_in0	m4_in1	m4_out	m4 1b_0_sa	m4 1b_1_sa	m3_in0
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
m3_in1	m3_out	m3 1b_0_sa	m3 1b_1_sa	m2_in0	m2_in1	m2_in2	m2_out	m2 1b_0_sa	m2 1b_1_sa
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
m2 2b_0_sa	m2 2b_1_sa	m1_in0	m1_in1	m1_out	m1 1b_0_sa	m1 1b_1_sa	REG2 mux_in0	REG2 mux_in1	REG2 mux_out
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
R2_mux 1b_0_sa	R2_mux 1b_1_sa	REG1 mux_in0	REG1 mux_in1	REG1 mux_out	R1_mux 1b_0_sa	R1_mux 1b_1_sa	REG0 mux_in0	REG0 mux_in1	REG0 mux_out
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
R0_mux 1b_0_sa	R0_mux 1b_1_sa	REG4 in	REG4 out	REG3 in	REG3 out	REG2 in	REG2 out	REG1 in	REG1 out
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
REG0 in	REG0 out	dz	u	y	z	u1			
100%	100%	100%	100%	100%	100%	100%			

参 考 文 献

[1] S.Kajihara, I.Pomeranz, K.Kinoshita：”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 Remov al 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.Yoshimu ra, “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.Internatio nal, 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.