ISSN 2186-5647

2-23

並列テストのためのコントローラの状態遷移出力のドントケア割当てアルゴリズム

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

1. はじめに

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，テストパターン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論理の変更により，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[3]．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[3].

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，疑似ブール最適化(Pseudo Boolean Optimizat-ion：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，テストパターン数が支配的である演算器の並列テストに着目し，できる限り多数の演算器の並列テストを実行可能にし，かつできる限りテストに使用する状態遷移数を削減するヒューリスティックアルゴリズムを提案する．

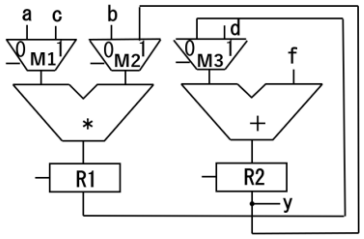

図 1.RTL データパスの部分回路

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 *R1~R2* はレジスタ，*と+は演算器，*M1~M3* は MUX，*a,b,c,d,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 と R2 はホールド機能付きレジスタである. 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．

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[9]を使用する．シミュレーションの結果からデータパスのハードウェア要素の並列テスト可能性をハードウェアテスト可能表[5]でまとめて，ドントケア割当てを実施する．

A Don't Care Filling Algorithm of Control Signals for Concurrent Testing
Haofeng XU Toshinori HOSOKAWA Masayuki ARAI Masayoshi YOSHIMURA

表 1. 悲観的な構造的記号シミュレーションによるハードウェアテスト可能表

ADD -in0	ADD -in1	ADD -out	MUL -in0	MUL -in1	MUL -out	M1-0	M1-1	M2-0
X	X	X	○	○	○	○	X	○
M2-1	M3-0	M3-1	m1-s0	m1-s1	m2-s0	m2-s1	m3-s0	m3-s1
X	X	X	X	○	X	○	X	X
R1in	R2in	R1out	R2out	r1-s0	r1-s1	r2-s0	r2-s1	a
○	X	X	○	○	X	X	X	○
b	c	d	f	y				
○	X	X	X	○				

2-3. 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を用いる．拡張した構造的記号シミュレーションは下のように定義する．

（定義 1 ：悲観的な構造的記号シミュレーション）

悲観的な構造的記号シミュレーショは構造的記号シミュレーショ[5]の処理とほぼ同じであり，あるハードウェア要素のテスト不可能となるように X の論理値割当てされていることを仮定する．各状態遷移で必ずテスト可能なハードウェア要素を示す．テスト可能なハードウェア要素を最小限に見積る．

（定義 2：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は構造的記号シミュレーショ[5]の処理とほぼ同じであり，あるハードウェア要素のテスト可能となるように X の論理値割当てされていることを仮定する．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示す．テスト可能なハードウェア要素を最大限に見積る．

3. 制御信号線のドントケア割当て

コントローラの状態遷移における制御信号にはドントケアが存在する場合がある．本提案手法はある状態遷移で並列テスト可能な演算器数ができるだけ多くなり，かつできる限り少ない状態遷移数で全ハードウェア要素をテスト可能になるようにドントケア割当てを行う．

3-1. ドントケア割当て

演算器の見積もりテストパターン数[5]は他のハードウェア要素と比較して支配的であるため，本提案手法は制御信号のドントケア割当てを行う時，演算器の並列テストについて着目する．

図 3 は本論文の制御信号のドントケア割当てのヒューリスティックアルゴリズムを示す．

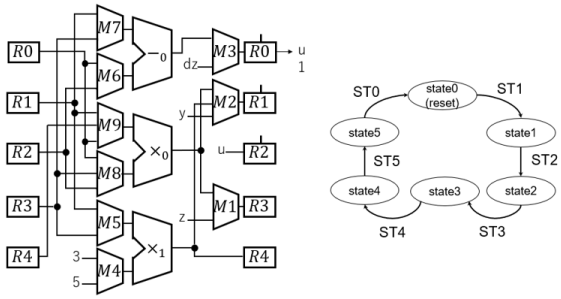

図2. ベンチマーク回路ex2

```
1.Heuristic_Algorithm(DP, CTL){
2.    TH = ∅; TST = ∅;
3.    ST =enumerate_state_transition(CTL);
4.    (HE, OP) = enumerate_hardware_element(DP);
5.    for(each st_i ∈ ST){
6.        th_i =optimistic_symbolic_simulation(st_i , DP);
7.        TH = TH ∪ th_i;
8.    }
9.    while(OP! = ∅){
10.       st_j =select_maxop_test_state_transition(TH);
11.       st_j =X-filling_control_signal(st_j , DP);
12.       th_j =pessimistic_symbolic_simulation(st_j , DP);
13.       TH = TH − th_j;
14.       TST = TST ∪ st_j;
15.       OP = OP − th_j;
16.       HE = HE − th_j;
17.   }
18.   while(HE! = ∅){
19.       st_k =select_maxhe_test_state_transition(TH);
20.       st_k =X-filling_control_signal(st_k , DP);
21.       th_k =pessimistic_symbolic_simulation(st_k , DP);
22.       TH = TH − th_k;
23.       TST = TST ∪ st_k;
24.       HE = HE − th_k;
25.   }
26.   return(TST);
27.}
```

図 3.ヒューリスティックアルゴリズム

（行 1）アルゴリズムに入力する情報は *RTL* 回路のデータパス *DP* とコントローラ *CTL* である．

（行 2）ある状態遷移でテスト可能なハードウェア要素集合の集合 *TH* とテストで使用する状態遷移集合 *TST* を空集合に初期化する．

（行 3）コントローラ *CTL* から状態遷移集合 *ST* を生成する．

（行 4）データパス *DP* からハードウェア要素集合 *HE* と演算器集合 *OP* を生成する．

（行 5）*ST* 中の各状態遷移st_i に対して，行 6,7 の処理を繰返し実行する．

（行 6）*DP* と状態遷移st_i の制御信号から楽観的な構造的記号シミュレーションを行い，テスト可能なハードウェア要素集合th_iを求める．

（行 7）求められた集合th_iを *TH* に追加する．

楽観的な構造的記号シミュレーションの結果により，各ハードウェア要素がテスト可能な状態遷移がわかる．表 2 は図 2 のベンチマーク回路 ex2[11]において楽観的な構造的記号シミュレーションの結果から解析された各ハードウェア要素のテスト可能な状

態遷移を示す.

(行 9) OPが空集合になるまで行 10~16 の処理を繰返し実行する.

(行 10) 並列テスト可能な演算器数が最大な状態遷移st_jを選択する.

表 2 の結果から ST4 でテスト可能な演算器数が最大であるため，ST4 を選択する.

(行 11) 選択された状態遷移st_jで並列テスト可能なハードウェア要素数が最大になるようにドントケアに論理値割当てを行う.

(行 12) 選択された状態遷移st_jと DPから悲観的な構造的記号シミュレーションを行い，テスト可能なハードウェア要素集合th_jを求める.

(行 13) THからth_jの要素を削除する.

(行 14) 状態遷移st_jを TSTに追加する.

(行 15) OPからth_jの要素を削除する.

ST4 を選択すると OPが空集合になる.

(行 16) HEからth_jの要素を削除する.

表 3 に示されるハードウェア要素は状態遷移 ST4 を選択した後の HEの要素を示す.

(行 18) ハードウェア要素集合 HEが空集合になるまで行 19~24 の処理を繰返し実行する.

(行 19) 並列テスト可能なハードウェア要素数が最大である状態遷移st_kを選択する.

表 3 の結果から ST3 でテスト可能なハードウェア要素数が最大であるため，ST3 を選択する.

(行 20) 選択された状態遷移st_kで並列テスト可能なハードウェア要素数が最大になるようにドントケアに論理値割当てを行う.

(行 21) 選択された状態遷移st_kと DPから悲観的な構造的記号シミュレーションを行い，テスト可能なハードウェア要素集合th_kを求める.

(行 22) THからth_kの要素を削除する

(行 23) 状態遷移st_kを TSTに追加する.

(行 24) HEからth_kの要素を削除する.

(行 26) TSTを出力する.

表 2. 楽観的な構造的記号シミュレーションによる各ハードウェア要素の状態遷移

MUL1	MUL0	SUB0	M1-in0	M1-in1	M2-in0	M2-in1	M2-in2
ST2 ST3 ST5 ST4	ST2 ST3 ST4	ST4 ST5	ST1	ST2	ST1	ST3	ST4
M3-in0	M3-in1	M4-in0	M4-in1	M5-in0	M5-in1	M6-in0	M6-in1
ST1	ST4 ST5	ST2	ST3	ST3	ST3	ST4	ST5
M7-in0	M7-in1	M8-in0	M8-in1	M8-in2	M9-in0	M9-in1	M9-in2
ST4	ST5	ST2	ST3	ST4	ST2	ST3	ST4
m1-sa0	m1-sa1	m2-1'b-sa0	m2-1'b-sa1	m2-2'b-sa0	m2-2'b-sa1	m3-sa0	m3-sa1
ST2	ST1	ST4	ST1 ST3	ST3	ST1 ST4	ST4 ST5	ST1
m4-sa0	m4-sa1	m5-sa0	m5-sa1	m6-sa0	m6-sa1	m7-sa0	m7-sa1
ST3	ST2	ST3	ST2	ST5	ST4	ST5	ST4
m8-1'b-sa0	m8-1'b-sa1	m8-2'b-sa0	m8-2'b-sa1	m9-1'b-sa0	m9-1'b-sa1	m9-2'b-sa0	m9-2'b-sa1
ST4	ST2 ST3	ST3	ST2 ST4	ST4	ST2 ST3	ST3	ST2 ST4

表 3. ST4 を選択したハードウェア要素表

			M1-in0	M1-in1	M2-in0	M2-in1	
			ST1	ST2	ST1	ST3	
M3-in0		M4-in0	M4-in1	M5-in0	M5-in1		M6-in1
ST1		ST2	ST3	ST2	ST3		ST5
	M7-in1	M8-in0	M8-in1		M9-in0	M9-in1	
	ST5	ST2	ST3		ST2	ST3	
m1-sa0	m1-sa1		m2-1'b-sa1	m2-2'b-sa0			m3-sa1
ST2	ST1		ST1 ST3	ST3			ST1
m4-sa0	m4-sa1	m5-sa0	m5-sa1	m6-sa0		m7-sa0	
ST3	ST2	ST3	ST2	ST5		ST5	
	m8-1'b-sa1	m8-2'b-sa0			m9-1'b-sa1	m9-2'b-sa0	
	ST2 ST3	ST3			ST2 ST3	ST3	

ex2 回路でテストに使用する状態は ST4,ST3,ST2, ST1 である.

3-3. PBO ソルバを用いて見積もりテストパターン数計算

PBO とは Partial Max SAT の一般化で, 最適化関数を最小化するように，制約式への論理変数割当てを求める問題である.

選択された状態遷移の構造的記号シミュレーションの結果を 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う．PBO ソルバに与える式は文献[5]で提案されたものである.

4. 実験結果

本提案手法ではフルスキャン設計が適用されたベンチマーク回路である ex2,ex4,maha,kim,sehwa[11]の 5 つの RTL 回路に対して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を適用して実験を行った．評価項目として，故障検出率，冗長故障数，面積オーバヘッド，テストに利用した状態遷移，ドントケア割当て率，テストパターン数について評価した.

RTL 回路生成のための動作合成には内製の動作合成ツール PICTHY を使用し，RTL のデータパス信号線のビット幅は 32 ビットとした．論理合成およびフルスキャン設計ツールはそれぞれ Synopsys 社の Design Compiler と DFT Compiler を使用し，自動テストパターン生成ツール(Auto Test Pattern Generator:ATPG)は内製の PBO が出力したテストスケジューリング情報を用いた多重目標故障テスト生成ツール[10]を使用した．また，対象故障モデルは縮退故障とし．テスト生成の終了条件は故障検出効率 100%に達することとする.

提案アルゴリズムを適用後に，コントローラの制御信号にドントケアが存在する時は，論理合成時に面積を考慮してドントケアに論理値を割当てる.

表 4 は実験対象回路の特性である. 1 列目は回路名,

2 列目は外部入力数，3 列目は外部出力数，4 列目はレジスタ数，5 列目は演算器数，6 列目はデータパスのビット幅，7 列目は状態信号線数，8 列目は制御信号線数，9 列目は状態数，10 列目は状態遷移数である．

表 5 は実験結果である. 回路名_ori はオリジナル回路である．回路名_pro は提案手法を適用し回路である．1 列目は回路名，2 列目は総故障数，3 列目は検出故障数, 4 列目は冗長故障数, 5 列目は故障検出率, 6 列目は故障検出効率，7 列目はテストで使用する状態遷移数/総状態遷移数，8 列目はコントローラに残っているドントケアの数, 9 列目は生成されたテストパターン数である，10 列目はテストパターン数の削減率，11 列目は面積オーバヘッドである．実験回路の面積オーバヘッドは提案手法の適用により平均 0.18%，最大 3.67%削減することができた．テストパターン数は平均 13.96%, 最大 23.29%削減することができた.

5. むすび

本論文では，並列テストのためのコントローラの状態遷移のドントケア割当てのヒューリスティックアルゴリズムを提案した．面積オーバヘッドは平均0.18%削減し，テストパターン数を平均13.96%削減することができた.

今後の課題は，多数のベンチマーク回路で実験を行うことが挙げられる.

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 Transactions 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 Sets, ”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] 浅見竜輝,細川利典,山崎紘史,吉村正義,新井雅之 “RTL ハードウェア要素のテストスケジューリング情報を用いた多重目標故障テスト生成法” 信学技報, vol. 120, no. 358, DC2020-74, pp. 30-35, 2021 年 2 月.

[11] M.T.-C.Lee, “High-Level Test Synthesis of Digital VLSI Circuits”,Artech House Publishers, 1997.

表 4. RTL 回路特性

回路名	外部入力数	外部出力数	レジスタ数	演算器数	ビット幅	状態信号線数	制御信号線数	状態数	状態遷移数
ex2	6	1	5	3	32	0	12	6	6
ex4	6	2	6	4	32	0	7	5	5
maha	16	1	7	3	32	1	9	24	29
kim	21	1	7	5	32	1	16	24	26
sehwa	14	2	9	3	32	1	16	26	31

表 5. 実験結果

	総故障数	検出故障数	冗長故障数	故障検出率	故障検出効率	状態遷移	ドントケア数	テストパターン数	テストパターン削減率	面積オーバヘッド
ex2_ori	21385	21347	38	99.82%	100.00%	-	46	73	-	-
ex2_pro	21068	21025	43	99.80%	100.00%	4/6	33	56	-23.29%	-1.18%
ex4_ori	19500	19499	1	99.99%	100.00%	-	18	62	-	-
ex4_pro	19418	19417	1	99.99%	100.00%	4/5	15	60	-3.23%	-3.67%
maha_ori	7707	7700	7	99.91%	100.00%	-	505	129	-	-
maha_pro	7788	7772	16	99.79%	100.00%	18/29	193	101	-21.71%	+1.01%
kim_ori	9088	9077	11	99.88%	100.00%	-	405	116	-	-
kim_pro	9335	9288	47	99.50%	100.00%	14/26	215	104	-10.34%	+1.95%
sehwa_ori	8602	8598	4	99.95%	100.00%	-	523	98	-	-
sehwa_pro	8855	8842	13	99.85%	100.00%	13/31	320	87	-11.22%	+0.99%