設計自動化 59－8
（1991. 10. 9）

順序回路のＡＴＰＧにおける検出率向上手法

太田 光保　　細川 利典　　本原 章

松下電器産業株式会社　　半導体研究センター

あらまし　順序回路のテスト生成は一般的に非常に困難であるために，処理の打ち切りが発生することが多い．ＲＴＰ法[1-3]を基本とした順序回路のテスト生成においては，ほとんどの場合処理の打ち切りは，故障伝搬または状態初期化で発生していることがわかっている．本報告では，この状態初期化の打ち切りに着目し，状態初期化が打ち切られている原因について考察を行ない，それらの問題を解消する方法として，（1）状態遷移の枝刈り，（2）初期状態を変えてテスト生成を行なう，の２つを提案する．また，この方法についてＩＳＣＡＳ８９ベンチマーク回路を用いて実験を行ない，従来の方法と比較して高い検出率を得ることができたことを報告する．

AN APPROACH TO FAULT COVERAGE IMPROVEMENT OF ATPG

Mitsuyasu Ohta　Toshinori Hosokawa　Akira Motohara

Semiconductor Research Center　Matsushita Electric Industrial Co.,Ltd

3-15, YAGUMO-NAKAMACHI, MORIGUCHI, OSAKA, 570 JAPAN

Abstract　An approch to fault coverage improvement in sequential ATPG is described. In general, ATPG of sequential circuits is very difficult and test generation procedure is often aborted. It is known that most of the abortion occured at fault propagation or state initialization of ATPG which adopts an algorithm based on reverse time processing(RTP)[1-3]. In this papaer, we propse two methods to reduce the abortion at state initialization. According to the preliminary experiments on ISCAS89 benchmark circuits, our approach shows that high fault coverage is obtained than the previous methods.

1. はじめに

論理回路が大規模化するに伴って ＬＳＩ開発におけるテスト設計コストの削減はますます重要な課題となっており，自動化の要求が高まっている．このテスト設計の技術については２つの大きな柱がある．一つは与えられた論理回路について，いかに高い検出率を持った系列を生成するかというテスト生成技術，更にもう一つはパーシャルスキャン等でテストパターンの生成が容易となるように回路を変更する検査容易化設計に関する技術である．本報告では，このテスト生成技術に関していくつかのアイディアを提案する．

現在の自動テスト生成技術では，組合せ回路についてはほぼテスト生成可能である．しかし，依然として順序回路については困難な問題を多く残している．本報告で用いた実験システムは，故障信号を外部出力にまで到達させる故障伝搬，および故障を励起した状態まで初期状態を遷移させる状態初期化を，時間軸と逆向きに行なうＲＴＰアルゴリズムを基本としている．状態初期化の成功／不成功には，初期状態と，その初期状態からの状態遷移を行なう経路の選択方法がかかわってくる．今回は，このＲＴＰ法を基本に，テスト生成を行うときの初期状態と状態初期化における状態遷移の経路の選択方法に着目したアイディアを加えたテスト生成システムを開発しＩＳＣＡＳ８９ベンチマーク回路を用いて実験を行ったので報告する．

ここで，対象とする論理回路は，組合せ回路中にフィードバックループを含まないものとし，故障はゲートレベルの単一縮退故障とする．

2. テスト生成アルゴリズム

図１は本アルゴリズムが対象とする順序回路の回路モデルである．この回路モデルは，外部入力と疑似外部入力と呼ぶフリップフロップの出力を入力とし，外部出力と疑似外部出力と呼ぶフリップフロップの入力を出力とするする組合せ回路部と，フリップフロップ（記憶素子）から構成されている．

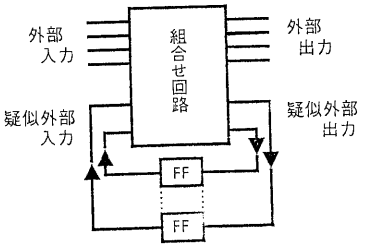

図1. 順序回路モデル

ＲＴＰ法[1-3]は，拡張Ｄアルゴリズム[4]等，従来のアルゴリズムと比較して，計算時間・メモリ量等において優位性が得られていることから第２世代の順序回路のテスト生成アルゴリズムと言われている．我々が今回実験を行なったテスト生成システムもこのＲＴＰ法を基本としている．以下，この実験システムで用いたテスト生成アルゴリズムの基本部分について，処理の順を追いながら説明する．図２に，このアルゴリズムのフローチャートを示す．

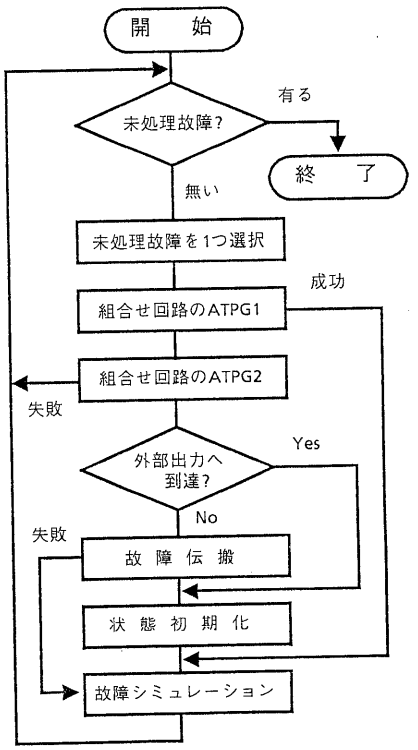

図2. 従来のテスト生成アルゴリズム

2. 1 組合せ回路のテスト生成（モード1）

疑似外部入力に初期状態を割り当てて，外部入力を自由に制御できる点，疑似外部出力を観測点として組合せ回路のテスト生成を行なう．ここで系列を生成することができたならば，故障伝搬・状態初期化の処理は必要なく，テスト生成は終了である．

2. 2 組合せ回路のテスト生成（モード2）

モード1で失敗したときに，外部入力およびフリップフロップの出力である疑似外部入力を制御点とし，また，外部出力およびフリップフロップの入力である疑似外部出力を観測点として，組合せ回路のテスト生成を行なう．このフェーズで，組合せ回路的に冗長な故障を同定することができる．また，故障が外部出力に伝搬した場合には，次の故障伝搬のフェーズをスキップすることができる．

2. 3 故障伝搬

このフェーズでは，ＲＴＰ法を用いて故障信号を外部出力に伝搬させる．図3に故障伝搬の概念を示す．

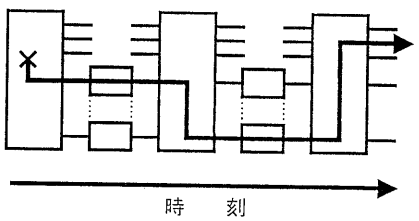

図3. 故障伝搬

図3では図1で説明した回路モデルを時間軸上に展開している．各タイムフレームでは，故障箇所または疑似外部入力から，外部出力または次のタイムフレームで故障経路となる疑似外部出力へ故障信号を伝搬させる．このタイムフレーム毎の処理を時刻を遡りながら行うことで，故障を検出する外部出力から順に経路を活性化することを実現する．

2. 4 状態初期化

故障伝搬と同様に故障信号を発生させたタイムフレームの状態を，初期状態から導かれるように時刻を遡りながら正当化する．

2. 5 故障シミュレーション

生成したテスト系列により故障シミュレーションを行ない，故障が実際に検出されるかどうかの検証を行なう．また，同時に検出される他の故障の確認も行ない，検出された故障については未処理の故障の集合から削除する．

この時，故障伝搬・状態初期化で打ち切られた系列も，回路の状態を変化させるために，そのまま用いている．

3. 打ち切り故障

順序回路のテスト生成は一般的に非常に困難であるために，処理の打ち切りが発生する．これまでの実験によりほとんどの場合，故障伝搬または状態初期化で発生していることが分かっている．

以下，故障伝搬・状態初期化における打ち切りの原因について検討する．

3. 1 故障伝搬での打ち切り故障

故障伝搬での打ち切り故障は以下の2つに分類することができる．

（1）故障を伝搬させる場合の経路を決定するヒューリスティクスが貧弱であるために，タイムフレームを遡っても故障を励起するタイムフレームに到達できず打ち切られる．

（2）タイムフレーム間で故障を伝搬させる際

に，同時に２つ以上のフリップフロップを活性化することが，現在のインプリメンテーションでは不可能なために，最終的に打ち切られるものがある．

（１）を解決するためには，ヒューリスティクスの強化が必要であるが，これは非常に困難な問題である．また，（２）については，複数のフリップフロップを活性化するアルゴリズムを採用することで解決は可能である．

3．２　状態初期化での打ち切り故障

アルゴリズムにより回路の状態遷移を自由に制御することは困難であるために，状態初期化で打ち切りが発生することが多い．特に状態遷移が複雑な回路においては，回路を初期状態まで遷移させるのが困難であり，打ち切りが発生する確率が高い．この状態初期化での打ち切りは次の２つに分類することができる．

（１）初期状態まで状態遷移を行なう経路を決定するヒューリスティクスが貧弱であるために，初期状態に到達できず打ち切られる．

例えば図４に示すように，初期状態Ｓ０から状態Ｓ４に遷移する系列の生成を考える．

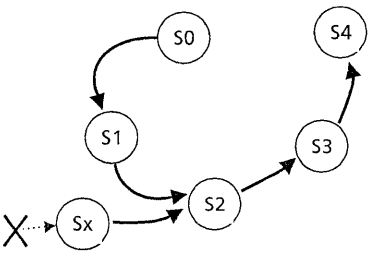

図4．状態初期化での打切り(1)

この時，Ｓ０からＳ４に遷移する経路は，Ｓ０→Ｓ１→Ｓ２→Ｓ３→Ｓ４が存在していたとする．ＲＴＰアルゴリズムは時刻を遡りながら系列を生成するので，［Ｓ３→Ｓ４］のパターン，［Ｓ２→Ｓ３］のパターンの順で系列が生成される．このときヒューリスティクスが貧弱であるために［Ｓｘ→Ｓ２］の経路を選択しそのパターンを生成した場合，初期状態Ｓ０まで到達できずに，いずれ打ち切りが発生する．

（２）初期状態が故障信号を発生させたタイムフレームの状態まで遷移させるのが非常に困難な状態にある．

例えば図５に示すような，初期状態Ｓ０から状態Ｓｎに遷移する系列の生成において，Ｓ０からＳｎに遷移する経路が非常に複雑な場合にも打ち切りが発生する確率が高い．仮にこの時の初期状態がＳｙであれば，状態遷移が単純であることから，簡単に状態初期化が成功することが予想される．

次章より，この問題を解消するための方法について述べる．

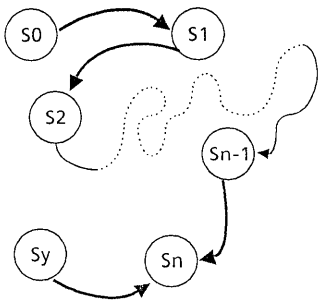

図5．状態初期化での打切り(2)

4．状態初期化のアルゴリズム強化

今回考案し実験を行なった，3．２で考察した状態初期化での打ち切りの原因を解消する手法について説明する．

4．１　状態遷移の枝刈り

3．２で考察した状態初期化における打ち切りの原因（１）は，ヒューリスティクスが弱いというものであった．ヒューリスティクスを強化すればこの問題は解消されるが，これは非常

に困難な問題である.

そこで, この原因（１）を解消する方法として, 一つ前の状態から現在の状態に遷移させる系列が生成できずに打ち切りが発生した場合に, 現在の状態を禁止状態として記憶し, 以後の状態初期化において, この禁止状態を通る経路を選択しないようにする. つまり, 状態初期化の失敗から学習し, 経路の枝刈りを行なうアルゴリズムを提案する.

例えば, 図４においては, ［Ｓ１→Ｓ２］の経路を選ぶべき所を, ［Ｓｘ→Ｓ２］の経路を選択していたため, 打ち切りが発生していた. この時, 回路がＳｘの一つ前の状態からＳｘへと状態遷移する系列の生成で打ち切られたならば, Ｓｘを禁止状態として記憶し, その経路の枝を刈る. これにより, ［Ｓｘ→Ｓ２］の経路は選択されなくなり, ［Ｓ１→Ｓ２］の経路が選択され, 状態初期化が正しい方向に進む.

4. 2 初期状態を変えての生成

3. 2で考察した状態初期化における打ち切りの原因（２）は, 初期状態から故障信号を発生させたタイムフレームの状態までの状態遷移が複雑であるというものであった. これも, もちろんヒューリスティクスを強化すれば解消されるが, 同様に困難である. そこで, この原因（２）を解消する方法を以下に提案する.

失敗の原因は図５のＳ０のような初期状態で, Ｓｎへの状態遷移を考えた点にある. 状態Ｓｙのような初期状態でＳｎへの遷移を考えれば初期化が容易に成功することが予想される.

そこで, テスト系列を生成すると回路の状態が変化することに着目し, 状態初期化に失敗した故障については, 図６のフローチャートに示すように, 他の故障のテスト系列を生成した後に, 再度未処理故障とすることで, 異なった初期状態からのテスト生成を試みる.

5. 実験結果

ＩＳＣＡＳ８９ベンチマーク回路について, 状態初期化の打ち切りに対する２つの提案について, それぞれ実験を行なった. 但し, 4. 2で提案した初期状態を変えて生成を行なう方法では, 図６に示すループは１度限りに制限している.

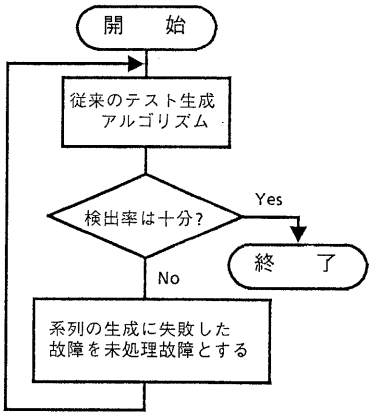

図6. 初期状態を変えての生成

表１は従来のテスト生成アルゴリズムに基づく結果, 表２は従来のテスト生成アルゴリズムに4. 1で行なった枝刈りの提案を加えた結果, 表３は従来のテスト生成アルゴリズムに4. 2で行なった初期状態を変えて生成を行なう提案を加えた結果, 表４は２つの提案共に加えた結果である. プラットホームとしては, ソルボーン シリーズ５／６００を用いている. また, 検出率はポテンシャルディテクトを含まないハードディテクトな検出率である.

初期状態変更を加えた場合（表３）は, 従来の方法と比較して, 検出率が上がった回路が１５, 検出率が下がった回路が１という効果が得られた. 枝刈と初期状態変更を加えた場合（表４）は, 枝刈のみを加えた場合（表２）と比較して, 検出率が上がった回路が１１, 下がった回路が２であり, また初期状態変更を加えた場合（表３）と比較して, 検出率が上がった回路が１２, 下がった回路が５という効果が得られている. しかし, 枝刈りのみを加えた場合（表２）を従来の方法と比較すると, 検出率が上がった回路が１０, 下がった回路が７とあまり効果が得られていない. これは, 禁止状態により無駄な検査系列の生成が阻害されたために, 系列生成の対象以外で同時に検出される故障が減

回路名	検出率 (%)	系列長	CPU時間(秒)	
			ATPG	Fsim
S27	100.00	27	0.20	0.20
S208	60.19	276	8.00	6.62
S298	80.84	437	15.37	6.80
S344	91.23	307	490.20	6.37
S349	90.57	267	732.28	6.92
S382	86.22	926	45.78	29.80
S386	66.41	351	2050.83	8.32
S400	83.25	974	69.42	35.88
S420	39.77	575	48.68	31.03
S444	78.69	783	70.57	31.73
S510	0.00	1	5.27	5.93
S526	77.30	4198	662.22	259.20
S641	83.73	193	190.23	11.43
S713	77.80	211	323.97	15.47
S820	46.47	1166	396.87	66.45
S832	46.32	1004	9259.23	59.72
S838	29.02	723	688.27	119.55
S953	7.78	34	16.05	32.63
S1196	99.03	419	20.38	30.50
S1238	93.73	413	30.13	31.22
S1423	59.14	1077	5023.15	255.78
S1488	73.15	1469	2515.80	110.63
S1494	75.23	1072	2179.23	89.45
S5378	67.17	2610	1356.42	553.88

表1. 従来の方法の結果

回路名	検出率 (%)	系列長	CPU時間(秒)	
			ATPG	Fsim
S27	100.00	27	0.22	0.20
S208	59.26	231	8.38	5.65
S298	82.79	398	15.63	6.57
S344	92.98	284	489.75	6.55
S349	91.43	264	732.73	6.58
S382	86.22	926	54.43	31.67
S386	70.31	271	1509.08	7.35
S400	83.73	1001	83.43	35.93
S420	39.53	487	55.82	29.37
S444	78.69	783	76.20	32.38
S510	0.00	1	5.35	6.02
S526	57.12	4623	896.82	314.42
S641	84.37	172	309.88	11.58
S713	79.86	212	319.63	16.18
S820	46.94	695	298.88	40.35
S832	45.52	613	5940.00	42.77
S838	28.90	724	750.67	117.23
S953	7.78	34	16.88	33.22
S1196	99.03	419	20.40	29.85
S1238	93.73	413	30.63	31.08
S1423	57.36	1189	6418.42	273.78
S1488	76.04	738	1282.75	68.22
S1494	51.59	261	677.42	46.58
S5378	70.13	2494	1922.78	565.13

表2. 枝刈りを加えた生成結果

回路名	検出率 (%)	系列長	CPU時間(秒)	
			ATPG	Fsim
S27	100.00	27	0.17	0.13
S208	60.65	386	10.30	8.75
S298	83.44	618	22.63	8.83
S344	91.52	487	495.82	8.07
S349	91.43	412	741.12	7.70
S382	86.22	1061	51.80	30.45
S386	66.67	505	2175.08	9.80
S400	83.25	1125	73.05	37.95
S420	40.00	909	59.70	52.38
S444	79.54	1023	78.07	36.63
S510	0.00	1	4.75	5.48
S526	78.20	6455	804.50	299.25
S641	85.22	213	191.08	12.23
S713	79.52	290	329.83	18.88
S820	49.65	1797	455.32	96.63
S832	46.67	1667	9218.98	88.18
S838	29.02	1093	706.30	190.27
S953	7.78	34	16.47	31.63
S1196	99.03	423	20.33	29.92
S1238	93.73	424	29.43	31.43
S1423	57.16	1365	5205.37	290.22
S1488	78.47	1832	2397.23	128.42
S1494	77.62	1652	2203.27	115.70
S5378	68.46	4092	1789.70	825.13

表3. 初期状態変更を加えた結果

回路名	検出率 (%)	系列長	CPU時間(秒)	
			ATPG	Fsim
S27	100.00	27	0.17	0.10
S208	59.26	249	9.60	5.58
S298	84.42	580	24.93	8.23
S344	92.98	360	493.50	9.03
S349	92.00	357	740.83	7.25
S382	86.22	1058	64.40	94.18
S386	74.48	308	1571.02	6.92
S400	83.73	1151	97.25	46.70
S420	39.53	711	78.27	41.93
S444	78.69	1045	99.30	48.48
S510	0.00	1	5.08	5.17
S526	66.13	7205	1308.60	583.50
S641	85.44	252	313.62	14.07
S713	79.69	217	446.22	73.32
S820	55.29	932	321.72	47.92
S832	57.24	940	6697.87	58.72
S838	29.02	1100	874.80	310.15
S953	7.78	34	17.42	31.87
S1196	99.03	423	21.37	28.30
S1238	93.73	424	33.65	31.28
S1423	74.26	1281	5937.63	623.62
S1488	89.30	1021	740.75	90.43
S1494	73.44	838	1262.52	96.42
S5378	69.30	4180	3974.32	1409.95

表4. 枝刈りと初期状態変更を加えた結果

ったためと考えられる.

6. まとめ

以上, ＲＴＰ法を用いた順序回路のテスト生成における問題点, 特に状態初期化の問題点とその解消方法について考察・提案を行なった. 実験結果より, 今回行なった提案は, 従来採用していたアルゴリズムより高い検出率を得られることがわかった. しかし, 枝刈りについては, 検出率の落ちる回路も少なくない.

今後の課題として, 3. 1で述べたような故障伝搬の改良を現在検討中である. これにより, 故障伝搬で打ち切られ, 他の故障を対象とした系列で同時に見つかっていた故障が検出されるために, 枝刈りによる負の効果が低減されると考える.

参考文献

[1] W.-T.Cheng,"Back Algorithm for Sequential Test Generation," Int. Conf. Computer Design, pp.66-69. Oct. 1988.

[2] D.R.Marlett,"An Effective Test Generation System for Sequential Circuits,"Proc. 23rd Design Automation Conf., pp.250-256, June 1986.

[3] M.H.Schulz and E.Auth,"ESSENTIAL : An Effective Self-Learning Test Pattern Generation Algorithm for Sequential Circuits," Proc.Int.Test Conf., pp.28-37, Aug.1989.

[4] G.R.Putzolu and J.P.Roth,"A Heuristic Algorithm for the Testing of Asynchronous Circuits," IEEE Trans. on Computers, Vol. C-20, No.6, pp.639-646,June 1971.

[5] F.Brglez,etal.,"Combinational Profiles of Sequential Benchmark Circuit,"Proc.Inc. Symp. Circuit and Systems, pp.1929-1934, May. 1989.

[6] H-K.T.Ma, S.Devadas, A.R.Newton,"Test Generation for Sequential Circuits,"IEEE Trans. on Computer-Aided Design, Vol.7,No. 10, pp.1081-1093, October 1988.

[7] E.Auth M.H.Schulz, "A Test-Pattern-Generation Algorithm for Sequential Circuits," IEEE DESIGN & TEST OF COMPUTERS, Vol.8, No.2, pp.72-85 June 1991.

[8] T.Niermann J.H.Patel,"HITEC : A TEST GENERATION PACKAGE FOR SEQUENTIAL CIRCUITS" EDAC'91.