7-7

並列含意操作を用いたテスト生成の効率化

日大生産工　○小松　正樹　日大生産工　細川　利典
九大　吉村　正義　明大　山崎　浩二

1. はじめに

近年，半導体微細化技術の進歩に伴い，大規模集積回路（Large Scale Integration: LSI）が大規模化し，テスト生成時間の増加が問題となっている．

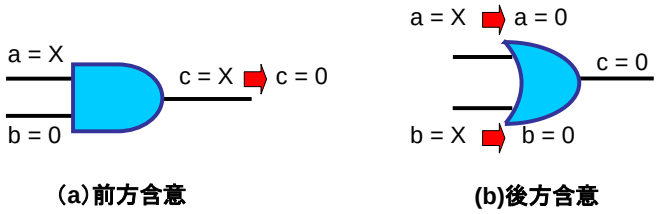
テスト生成の高速化のため，一意に値を決定できる，できるだけ多くの信号線の値を決定する必要がある．そのための手法として間接含意[2]がある．間接含意は SOCRATES[1]で提案された静的学習[2]で求める．テスト生成および静的学習の処理で，回路内の信号値を一意に決定していく含意操作[3]を行っている．含意操作は直接含意と間接含意に分けられ，間接含意は直接含意の結果から求められる．静的学習によりテスト生成の前処理の段階で間接含意を求め，テスト生成中の含意操作で利用することで，テスト生成時間を高速化する．

また含意操作を高速化する技術の一つとして並列含意操作[4][5]がある．並列含意操作は信号線の値を計算機の各ビットに対応付けて表す．もし計算機の整数型 1 語が 32 ビットなら最大で 32 本の信号線の値を同時に保持できる．各ゲートの演算は同じ位置のビット毎に論理演算を行なうビット演算で一度に行うことができる．こうして最大 32 本の含意操作を並列処理することができ，処理の高速化につながる．

静的学習はテスト生成の高速化に必要不可欠であるが，近年の LSI の大規模化に伴い，静的学習内の含意操作の処理時間の増加が問題となっている．本論文では文献[4]で提案された並列含意操作を静的学習で用いて，静的学習を高速化することを目的としている．

2. 含意操作

本章では含意操作について説明する．含意操作はゲートの入出力の接続関係から値を求めることのできる直接含意[2]と，



(a)前方含意　(b)後方含意

図 1　直接含意例

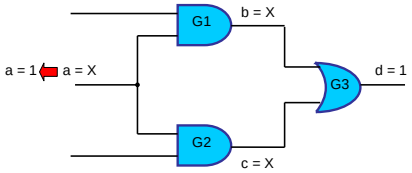


図 2　間接含意例

ゲートの入出力の接続関係だけでは値を求めることのできない間接含意[2]がある．また直接含意は前方含意と後方含意に分けられる．

図 1 に直接含意例を示す．図 1(a)は前方含意の例である．信号線 b の信号値が 0 である場合，AND ゲートの接続関係から信号線 c の信号値は 0 と決定できる．このようにゲートの入力から出力側に値を決定していくのが前方含意である．図 1 (b)は後方含意の例である．信号線 c の信号値が 0 である場合，OR ゲートの接続関係から，信号線 a,および b の信号値は 0 と決定できる．このように出力側から入力側に値を決定していくのが後方含意である．

直接含意は論理ゲートの入出力関係と推移律を利用して容易に得ることができる．

図 2 に間接含意例を示す．信号線 d の信号値 1 とした場合，直接含意では他の信号線は決定できないが，b=1 または c=1 のいずれかが成り立たなければならないため，いずれの場合も a=1 が成り立つ．図 2 のように“d=1 ならば a=1”となるのが間接含意である．

Promotion of Test Pattern Generation used by parallel implication

Masaki KOMATSU, Toshinori HOSOKAWA, Masayosi YOSHIMURA
and Kouji YAMAZAKI

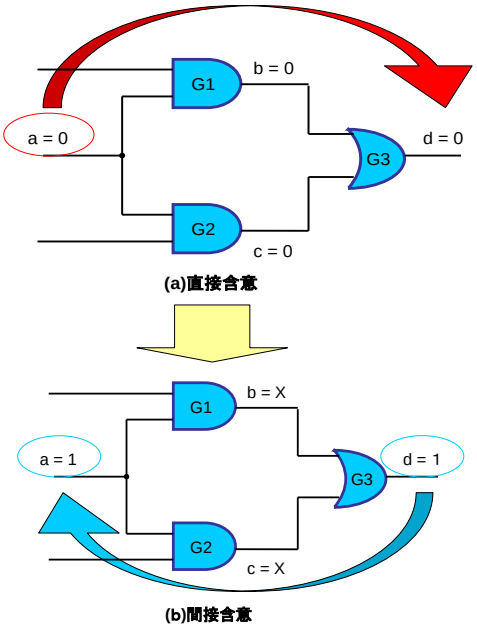


図 3　静的学習の例

3. 静的学習

静的学習では直接含意の結果から間接含意を求めている．そのため，命題が真ならばその対偶も真であるという対偶規則を用いることができる．本論の静的学習では次の学習規則 A を満たす直接含意が存在するとき間接含意を保存する．

[学習規則 A]

信号線 s の値を v(v∈{0,1})とする前方含意操作により，ある 2 入力以上のゲートの出力信号線 t が値 w(w∈{0,1})をとり，かつ，w がそのゲートの非制御値であるとき，間接含意として”t = ¬w ならば s = ¬v”を保存する．ここで非制御値は AND ゲートに対する論理値 1 のようにゲートの入力値がすべて決まったときに出力も決まる値である．

図 3 に静的学習の例を示す．図 3(a)は，まず信号線 a の信号値を 0 とし，前方含意操作を行なう直接含意の例を示す．その結果，“a=0 ならば d=0”とわかる．この対偶“d=1 ならば a=1”も真となる．信号線 d は 2 入力以上のゲートの出力信号線であり，値 0 はゲートの非制御値のため，図 3(b)のように“d=1 ならば a=1”という間接含意を学習する．

学習を行なう信号線の順序は文献[6]で学習順序として効果が高いとされる，回路の論理段数を基にした前方への幅優先探索による学習順序を用いる．

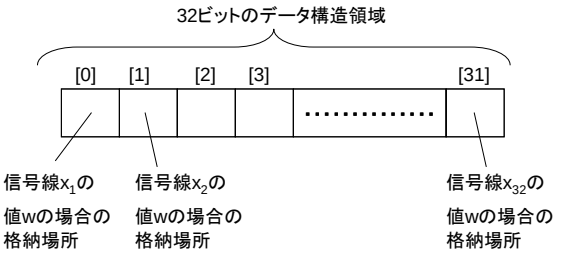


図 4　並列含意操作で使用する信号線 1 本のデータ保存領域

4. 並列含意操作を用いた静的学習

本章では並列含意操作およびそれを用いた静的学習について説明する．図 4 に並列含意操作で使用される信号線のデータ構造を示す．ビット数は 32 ビット計算機での使用を仮定する．

32 ビットあるデータ構造の領域の 1 ビット目には，ある信号線 x_1 を信号値 w (w∈{0, 1}) と設定した場合の信号線 s の値が保存される．同様に n ビット目には別の信号線 x_n を値 w と設定した場合の信号線 s の値が保存される．最大 32 本の信号線を並列含意可能である．このデータ構造を用いた場合，含意操作はビット演算を用いることによって 32 ビット一度に計算可能である．よって従来手法では信号線 1 本を対象に行なっていた静的学習を，最大 32 本の信号線をグループ化し，グループの単位で含意操作を行なうことが可能であり，処理の高速化につながる．

並列含意操作によって同時に含意操作される信号線群を信号線グループとする．信号線グループに属する信号線は，樹枝状回路 (Fanout Free Region :FFR) に属する信号線を優先的にグループ化する．FFR は分岐元信号線(Fanout stem)から入力側に回路をたどり，外部入力か分岐先信号線(branch)までに存在した信号線の集合である．FFR 内の信号線でグループ化すれば，含意操作の範囲となる信号線が，分岐元信号線に収束していくため，含意操作を並列化しても，含意すべき信号線は少なくなり，並列含意操作の効率が上がる．本論文では並列含意操作を行う際の信号線グループは同一の FFR に属する信号線から信号線グループを作成する．

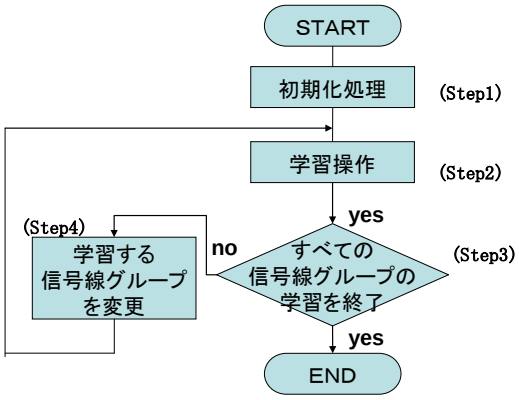


図 5　並列含意操作を用いた静的学習のアルゴリズム

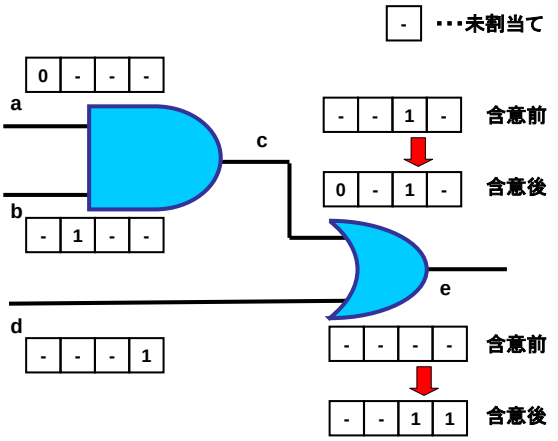


図 6　並列含意操作の例

図 5 に並列含意操作を用いた静的学習のアルゴリズムを示す．以降で各ステップの処理内容を説明する．

(Step1)
初期化処理として間接含意の保存領域を確保する．次に並列含意操作を行う各信号線グループの信号線と，それらの信号線グループの静的学習を行う順番を決定する．

(Step2)
信号線グループをもとに学習操作を行う．信号線グループ内の各信号線に初期設定で値を割当て，含意操作を行う．含意操作の終了後，学習規則 A を満たす間接含意がある場合，間接含意の情報を保存する．

(Step3)
すべての信号線グループに対して学習操作を行ったかの判定を行う．行なっている場合，静的学習を終了する．行なっていない場合，他の信号線グループで学習操作を行うために(Step4)に進む．

(Step4)
学習操作を行う信号線グループを変更し，再び(Step2)の学習操作を行う．

図 6 に並列含意操作の例を示す．計算機の 1 語のビット数は 4 ビットと仮定する．並列含意操作で使用されるデータ構造の初期設定として，1 ビット目は信号線 a を値 0，2 ビット目は信号線 b を値 1，3 ビット目は信号線 c を値 1，4 ビット目は信号線 d を値 1 とする．これ以外のビットは値が未割当ての状態とする．

この初期設定の下で含意操作を開始する．そのため，例として信号線 a のデータ構造では含意前の状態が 1 ビット目は 0，他のビットは未割当ての状態となっている．まず信号線 a と信号線 b から信号線 c の内容を決定する．信号線 c の 1 ビット目は信号線 a に制御値の 0 が割当てられているため，値が伝搬し 0 となる．2 ビット目は信号線 b に割当てられた値 1 が非制御値のため，信号線の値が一意に決まらず未割当て状態となる．3 ビット目は信号線 a と信号線 b ともに未割当ての状態であるが，信号線 c で 3 ビット目を 1 と定義しているので 3 ビット目は値を変更せず 1 とする．4 ビット目は信号線 a と信号線 b ともに未割当ての状態のため，信号線 c も未割当ての状態とする．次に信号線 e の含意操作を行う．1 ビット目は信号線 c の値 0 が非制御値であり，信号線 d は未割当てのため，値が一意に決定せず未割当てとなる．2 ビット目も信号線 c と信号線 d がともに未割当てのため，同様に未割当てとなる．3 ビット目と 4 ビット目は入力信号線である信号線 c か信号線 d のどちらかに制御値である 1 が割当てられているため 1 と決定できる．他に含意可能な信号線がないため，含意操作を終了する．

表 1　間接含意数と実行時間の比較

	辞書数		学習時間	
回路名	従来法	提案手法	従来法	提案手法
s27.v	1	1	0.03	0.05
s298.v	106	5	0.09	0.11
s510.v	2555	28	0.47	0.22
s953.v	12260	490	1.33	0.61
s1494.v	9764	0	2.08	1.47
s5378.v	5140	500	3.06	15.5

表 2　故障検出率と実行時間の比較

	故障検出率(%)		CPUタイム(SEC)	
回路名	従来法	提案手法	従来法	提案手法
s27.v	100	100	0.03	0.03
s298.v	100	100	0.89	0.3
s510.v	91.67	91.67	2.66	1
s953.v	96.28	96.28	9.31	3.13
s1494.v	99.2	99.2	22	7.19
s5378.v	99.08	99.08	227.05	28.06

5. 予備実験結果

テスト生成ソフトウェア STAGY[7]の静的学習部に並列含意操作を実装し ISCAS89 ベンチマーク回路の組合せ回路部分に対して実験を行った．信号線を一本ずつ含意操作して静的学習を行う方法を従来手法とし，信号線グループごとに並列含意操作して静的学習を行う方法を提案手法とする．表 1 は静的学習を実行した時に得られた間接含意数と実行時間の比較である．表 2 は静的学習後にテスト生成まで実行した時の故障検出率と実行時間の比較である．実験の結果，提案手法は従来手法と同等の間接含意数を獲得でき，s5378 では従来手法より１２倍ほど実行時間を削減できたため，並列含意操作が有効であると分かる．

6. おわりに

本論文では並列含意操作を静的学習部に組み込み評価を行った．その結果，得られる間接含意数は同様のまま実行時間を最大１２倍ほど削減することができた．今後の課題として学習基準 B に関しても並列含意操作を実装し，静的学習部に組み込んで評価を行うことが挙げられる．

参考文献

[1] M.Schulz “SOCRATES：A Highly Efficient Automatic Test Generation System” IEE TRANSACTIONS ON COMPUTER – AIDED DESIGN,VOL 7 NO 1,JANUARY (1988) p130

[2]梶原誠司　猿渡慶太郎“静的学習における効率的な間接含意の発見と保存について” 電子情報通信学会　TECHNICAL REPORT OF IEICE (2002) p26

[3]藤原秀雄,「ディジタルシステム設計とテスト」工学図書株式会社 (2004) p163

[4]南山哲郎　高松雄三“組合せ回路における冗長故障の判定法に関する考察” 電子情報通信学会論文誌 D-Ⅰ Vol.J81-D-Ⅰ No.10 pp1149-1156(1998) p1149~1156

[5]Mahesh A, Iyer and Miron Abramovici “FIRE:A Fault-Independent Combinational Redundancy Identification Algorithm” IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS , VOL 4 NO 2 JUNE 1996 (1996) pp295~301

[6] Hedeyuki ICIHARA Seiji KAJIHARA Kozo KINOSITA “On Processing Order for Obtaining Implication Relations in Static Learning” IEICE TRANS INF . &SYST., VOL.E-83D,NO10 OCTOBER 2000 (2000) p1908~1909

[7]大森悠翔,“静的学習による組合せテスト生成の高速化アルゴリズムに関する研究” 平成17年度日本大学生産工学部数理情報工学科卒業研究概要集,2005