2S-5 共有二分決定図を用いた組合せ論理回路のテスト生成

井置　一哉　　石浦　菜岐佐　　矢島脩三
(京都大学工学部)

1．はじめに

組合せ論理回路のテストパタン（故障検査入力）を高速に求めるアルゴリズムは種々研究されており[1][2]、多くの故障に対するテストパタンは現実的な時間で求めることができる。しかしテスト生成問題はＮＰ完全であり、バックトラック回数の多くなる故障が存在する。このような故障に対するテスト生成法として二分決定図を用いる方法[3]が提案されているが、本稿ではさらに効率の良い共有二分決定図を用いた単一故障伝播によるテスト生成法を提案する。

2．共有二分決定図

論理関数の表現法の１つに二分決定図（BDD:Binary Decision Diagram）と呼ばれるグラフ表現がある。このグラフ表現において複数の論理関数でサブグラフを共有したもの（図１）を共有二分決定図と呼ぶ[4]。

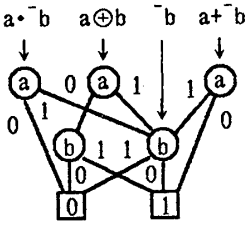

図１　共有二分決定図

共有二分決定図の特長は主に次の３点である。
(1)複数の関数間でもサブグラフを共有するので多くの論理関数が同時にコンパクトに表現できる。
(2)論理関数の等価性判定がグラフへのポインタの一致判定だけで行える。
(3)演算結果テーブルの利用により同じ関数に対する演算の重複を避けることができるため演算効率が良い。

また、複数の関数を同時に表す際に、不要となった関数に対するグラフだけを解放することも可能である。

3．共有二分決定図を用いたテスト生成

3.1　記号故障シミュレーション

外部出力の中に正常時の論理関数と故障発生時の論理関数が異なる信号線があればその故障は検出可能であり、なければ冗長故障である。我々は外部出力の論理関数を求める操作を共有二分決定図を用いて行う。この操作は全入力パタンに対するシミュレーションを行うのと等価であるが、サブグラフの共有により同じ演算の重複が避けられるので非常に効率が良い。従って従来のテスト生成法でバックトラックが頻発する故障に対しても効率良くテスト生成が行える可能性がある。本稿では記号故障シミュレーションの一手法として、共有二分決定図の特長を生かした単一故障伝播に基づく方法を提案する。

Test Generation for Combinational Logic Circuits Using Shared Binary Decision Diagrams
kazuya IOKI, Nagisa ISHIURA, Shuzo YAJIMA
Faculty Engineering, Kyoto Univercity

3.2　単一故障伝播

通常の故障シミュレーションにおける単一故障伝播[8]の手続きは次のとおりである。

step1 正常時の回路のシミュレーションを行い、全信号線の論理値を記憶する。
step2 各々の故障について故障位置から故障時の信号値を伝播させ、故障の影響が外部出力まで到達するかを調べる。

この考え方を記号故障シミュレーションに用いると次のようになる。

step1 正常時の回路の記号シミュレーションを行い、全信号線の論理関数を記憶する。
step2 各々の故障について故障位置から故障時の論理関数（故障位置では縮退値をとる定数関数）を伝播させ故障の影響が外部出力に到達するかを調べる。

step1ですべてのネットに対する論理関数を記憶する必要があるが、共有二分決定図を用いているため比較的コンパクトな表現になる。またstep2での正常時と故障時の論理関数の等価性判定はグラフへのポインタの一致判定だけで行える。多くの場合故障の影響が伝わる範囲は狭く、また故障時の関数は正常時と大きく異ならないので、故障時の関数を表すグラフを含めてもグラフはあまり大きくならないと考えられる。

step2で故障伝播が終了するのは、
(1)外部出力に故障の影響が伝わった場合、
(2)正常時と故障時の論理関数が回路内のすべての信号線で等しくなった場合、
のいずれかである。(1)の場合は検出可能としてテストパタンを求め、(2)の場合は冗長故障として終了する。

実際のプログラムでは効率を良くするため次のような手続きを用いている。

step1 各故障についてstep2からstep6の操作を行う。
step2 故障位置の正常時の論理関数が求まっていなければこれまでに求まっている点から記号シミュレーションを行い、これを求める。
step3 故障位置の故障時の関数を縮退値の定数関数とする。
step4 正常時と故障時の論理関数が等価でない限り、ゲートの出力線の正常時と故障時の論理関数を並行して求めていく。
step5 故障の影響が外部出力に一カ所でも到達するか、回路内のすべての信号線で正常時と故障時の論理関数が等価になれば終了する。
step6 故障時の論理関数のグラフを解放する。

正常時の論理関数を表すグラフの解放は行わないため、正常時の論理関数は一度求めるだけでよい。したがって正常時の論理関数をすべて求めてしまうと後は高速にテスト生成が行える。また検査可能性の判定が

できるとすぐに処理を終了するため、すべての信号線の論理関数を求める必要のないことが多い。特に冗長故障の場合は故障の影響がすぐに消えることが多く、効率が良い。

3.3 テストパタンの生成

故障が検出可能と判定された場合にはテストパタンの生成を行う。その手続きは次のとおりである。

step1 故障の影響が現れた外部出力の正常時と故障時の論理関数の排他的論理和をとる

step2 step1 で作ったグラフを根から始めて直接論理値0の葉につながっていない枝を次々にたどる。

step2の操作の結果論理値1の葉に到達する。たどってきたパスに対応する変数の値の組合せが求めるテストパタンである。なお、step2の処理は変数の数に比例する時間で行える。

3.4 変数の順序づけ

二分決定図の大きさは変数の順序に依存する。この変数順序の決定はヒューリスティックを用いて行う。外部出力から深さ優先で回路をたどり到達した外部入力の順序を変数順序にすれば、ある程度良い順序になることが知られている[5]。そこで我々は observability measure の良い順に深さ優先で順序を決めた。observability measure を用いたのは外部出力の論理関数に大きな影響を与える変数を優先した順序をつけると、その影響力によってグラフが小さくなると考えられるからである。また複雑な論理関数を優先するという意味で controllability measure の悪い外部出力から深さ優先の処理を始めた。この変数順序によって、元のネットリスト順ではグラフが大きくなりすぎて最後まで論理関数を求めることができなかった回路に対してもテスト生成が行えるようになった。

4. 性能評価

本稿で述べた手法に基づくテスト生成プログラムを SUN3/60 上で c 言語を用いて実現した。ISCAS のベンチマーク回路[6]に対してテスト生成の実験を行った結果を表1に示す。[7]の重み付き乱数を用いたテスト生成の結果検査できなかった故障のみを対象とした。但し、c880については冗長故障が存在しないため全故障を対象とした。表中「使用ノード数」はテスト生成に必要としたノード数であり、「正常時のみのノード数」は全信号線の正常時の関数を記憶するために必要なノード数である。c499、c1355などでは故障の影響がすぐに消滅するため、3.2の効率化手法により正常時の記号シミュレーションを行う場合よりも格段に少ないノード数で処理が行えている。c5315に関してはノード数が20万になった時点で処理を打ち切ったため1つの故障について冗長性の判定が行えなかったが、その他の4回路についてはすべて冗長故障であることが判定できた。またc880に関しては全故障に対してテストパタンが生成できた。

5. おわりに

共有二分決定図を用いた単一故障伝播によるテスト生成法について述べた。正常時の論理関数さえ求められれば冗長故障の同定は比較的簡単に行える。しかし大規模な回路に対してはグラフが大きくなり過ぎて演算を行うことができなくなる。そこで今後は変数の順序づけの問題を考えるとともに、記憶量を減らすためにすべての故障を並行して処理し、不要なグラフを解放する手法を実現する予定である。

謝辞

本学、高木直史博士、伊藤雅樹氏、湊真一氏をはじめ、御助言、御討論頂いた矢島研究室の諸氏に感謝いたします。

参考文献

[1] H. Fujiwara: Proc.ISCAS-85,pp.671-675.(1985)
[2] M. Schulz, E. Auth: Proc. FTCS-18, pp. 30-35, (1988)
[3] R. Gaede, M. Mercer, K. Bulter, D. Ross: Proc. 25th DAC, pp. 597-600, (1988)
[4] 湊,石浦,矢島: "共有二分決定図を用いた記号シミュレーション", 本大会予稿集,(1989)
[5] 藤田,藤沢,川戸: 情処研報,88-D-43-2,(1987)
[6] F. Brglez and H. Fujiwara: Special Session on ATPG and Fault Simulation, ISCAS85, (1985)
[7] 伊藤他: "高速故障シミュレーターを用いた重みつき乱数によるテスト生成", 本大会予稿集, (1989)
[8] S. Hong: Proc. FTCS-8, pp.96-99,(1978)

回路名	故障数	冗長と判定された故障数	テストが生成された故障数	打ち切り故障数	使用ノード数	正常時のみのノード数	CPU時間(秒)
c432	4	4	0	0	141,026	137,403	57.1
c499	8	8	0	0	27,954	79,914	83.9
c1355	8	8	0	0	19,883	174,481	68.0
c1908	9	9	0	0	55,013	50,852	57.2
c5315	59	58	0	1	200,000	——	75.1
c880	942	0	942	0	36,394	25,217	625.9

表1　実行結果(SUN3/60, 主記憶 8MB)