設 計 自 動 化 75－1
（1995．5．26）

論理回路フォーマル検証システム Condor の多値論理対応について

向山 輝 *, 若林 一敏 *, 藤田 友之 *, 山際 肇 **, 菅波 和幸 **, 田中 英俊 **
* 日本電気 (株), ** 日本電気ソフトウェア北陸 (株)

2 値論理回路の形式的論理検証を行なうシステム Condor の多値論理回路検証への拡張手法を提案する。多値論理の表現には複数の定数ノードを持つマルチターミナル BDD を用いる。マルチターミナル BDD は、2 値の BDD 演算アルゴリズムで処理することができず、否定枝を使用した時に一意性の保証ができないといった問題がある。我々は、BDD 上で到達可能な定数ノードを示すフラグの設定と、否定枝使用ルールの設定によってこの問題を解決し、多値論理回路検証のためのマルチターミナル BDD および多値論理演算を提案する。これらを用いて多値論理回路の形式的検証システムを実現した。

Multi-Value Logic Verification System

Akira MUKAIYAMA* Kazutoshi WAKABAYASHI* Tomoyuki FUJITA*
Hajime YAMAGIWA** Kazuyuki SUGANAMI** Hidetoshi TANAKA**
*NEC Corp., **NEC Software Hokuriku Ltd.

This paper describes the extension of our formal verification system called Condor to the multi-value logic verification system. The Multi-Terminal BDD is introduced for representing the multi-value logic. The Multi-Terminal BDDs can not be dealt with existing algorithm for logic operations of ordinary BDDs. We develop new algorithm for dealing with the Multi-Terminal BDDs by using flags to indicate reachability to the terminal nodes and defining a rule for the usage of complement edges.

1 はじめに

コンピュータ等の装置設計において、設計対象の論理回路の規模は年々急激に増大している。これにともなう設計期間の増大に対処するために、われわれは形式的論理検証システム Condor[1] を開発し、当社で開発する装置の設計検証に運用してきた。設計工程における Condor の位置付けは以下の通りである。

コンピュータ設計においては FDL(Function Description Language)[2] と呼ぶハードウエア記述言語を用いたレジスタトランスファレベル (RT レベル) で設計を行なう。われわれは、FDL の論理シミュレーションを行なうハードウエアアクセラレータ HALIII [3] を既に開発しており、論理の正当性を確認した FDL を基にしてゲートレベル回路の設計を行なう。

ゲートレベルの設計は多人数並行開発で行なうために、装置はトップダウンに分割階層化される。装置レベルに統合した論理シミュレーションは FDL を対象に行なう。このため、各設計単位でゲートレベルの回路と FDL が論理的に一致していることを確認することが必要となる。従来はこれを多数のパタン入力によるシミュレーションによって行なっていたため、膨大な計算機資源と工数を必要とし、設計工程における大きなボトルネックとなっていた。そこで、形式的論理検証システム Condor を開発し、設計単位で FDL とゲートレベル回路との論理的な等価性を高速に検証し、設計工数を大幅に削減することに成功した [1]。

Condor は、図 1 に示すように、RT レベルの仕様と、設計したゲートレベルの回路とをそれぞれレジスタからレジスタまでの組合せ回路に分割し、対応する 2 つの組合せ回路の論理を二分決定グラフ (BDD)[4][5] で表し比較することによって論理の等価性を検証するものである。論理が一致しない場合は論理の差異を検出するための入力パタンを生成する。

当初検証対象とした回路では、バスは回路の外部にしか存在しなかったため、ハイインピーダンスを検証で考慮する必要はなかった。ところが LSI の集積度の向上と共にバスが LSI の内部に入るようになり、バスを含んだ回路の論理検証を行なうことが必要となってきた。

そこで、0、1 の他に ハイインピーダンス、不定値、ショートの論理値を持つ多値論理回路の形式的検証が可能となるように Condor を拡張する。多値論理関数を表す方法としては、論理値を複数ビットで符合化する方法 [7] が提案されている。これは 2 値の処理プログラムが使える利点があるが、ハイインピーダンスの不正伝搬や回路がショートするパタンの検出といった我々の目的に即した多値論理演算を定義することが困難であるため、5 つの定数ノードを持つマルチターミナル BDD を用いる。

マルチターミナル BDD は、2 値の BDD 演算アルゴリズムで処理することができず、否定枝を使用した場合に一意性の保証ができないといった問題がある。本稿では、マルチターミナル BDD および多値論理演算の定義と、マルチターミナル BDD を実現する上での問題を解決する方法について報告する。

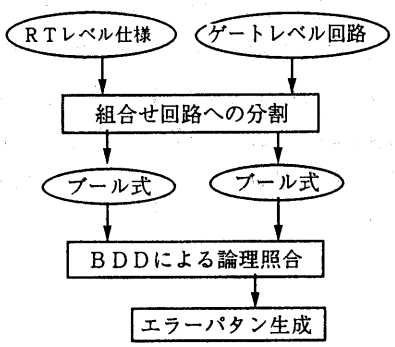


図 1 Condor の概要

2 BDD の多値化

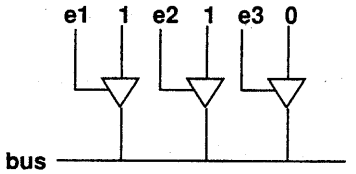
2.1 定数ノードの追加

BDD の定数ノードとして、従来の 0、1 以外の定数を表すノードを新しく定義する。ここではその種類と必要となった背景を説明する。

2.1.1 ハイインピーダンス、ショートの表現

バスを含む回路が検証対象となったことによって、例えば図 2(a) のような回路の論理を ROBDD で表すことが必要となった。図 2(b) は、この回路の仕様を表す FDL である。FDL では、“=” の左辺の信号の機能を右辺に記述する。機能の分岐は、「IF ～

THEN ～ ELSE ～」で構成される文で記述することができる。ハイインピーダンスは "Z"、ワイアード結合は、".WIRD." で表す。



(a) ゲート回路

w1 = IF *e1* THEN 1 ELSE Z;
w2 = IF *e2* THEN 1 ELSE Z;
w3 = IF *e3* THEN 0 ELSE Z;
bus = *w1* .WIRD. *w2* .WIRD. *w3*;

(b) FDL

図 2 バスを含む回路の例

バスに複数の確定値が伝搬するとバスはショートする。図 2 の *bus* は 図 3 に示すように *e1*, *e2*, *e3* の値によって、0、1、ハイインピーダンス、ショートのいずれかの状態値を取る。*bus* の論理を BDD で表す時にこれらの値を表現することが必要である。

e1	e2	e3	bus
0	0	0	ハイインピーダンス
0	0	1	0
0	1	0	1
0	1	1	ショート
1	0	0	1
1	0	1	ショート
1	1	0	ショート
1	1	1	ショート

図 3 *bus* の状態値

2.1.2 不定値の表現

図 4 の FDL のように ELSE 節がない IF 文で記述される信号は、ELSE 条件の時は未定義 (禁止入力) であり、このときの状態値は不定値となる。

一方、図 5 のような FDL で記述される回路では、*e1* の値によってハイインピーダンスがゲートに伝搬される場合がある。ゲートの入力がハイインピーダンスであると、その出力は 0、1 のどちらになるか不定である。

このような信号の論理を表すために、不定値を表す BDD が必要である。

sig1 = IF *cond1* THEN *sig2* ;

図 4 ELSE 節の無い FDL の例

w1 = IF *e1* THEN *a* ELSE Z;
sig2 = *w1* .AND. *b*;

図 5 ハイインピーダンスが伝搬される FDL の例

2.1.3 BDD のマルチターミナル化

以上のような背景から信号の状態としてハイインピーダンス、不定、ショートを表現するために、BDD の定数ノード (ターミナルノード) として、0、1 の他に、Z(ハイインピーダンス)、X(不定)、S(ショート) を定義する。

多値論理を正確に表現するためには、多分岐の決定グラフを用いることが必要であるように思われるが、検証対象とする回路では 0、1 以外の値は回路内部でのみ発生し、回路のバウンダリで入力されることはあり得ないため、0、1 の二分岐の決定グラフで十分である。

n 値の論理関数を表す方法としては、論理値を log n ビットで符合化しlog n個の 2 値 BDD で関数を表す方法がある [7]。この方法は 2 値 BDD の処理プログラムをそのまま使える利点があるが、次節で述べるような演算を定義することが困難であるため、5 つの定数ノードを持つマルチターミナルの BDD を使用する。

図 2 の *bus* を表すマルチターミナル BDD は図 6 のようになる。

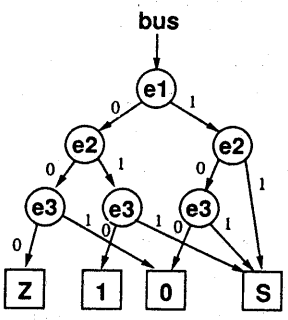


図 6 マルチターミナル BDD

2.2 演算子の追加・再定義

2.2.1 ワイアード結合

バスを含む回路を扱うために、2本の信号のワイアード結合を表す演算子が必要となった。そこで、新しい二項演算子 WIRD を図7 に示すように定義する。

WIRD	0	1	X	Z	S
0	S	S	S	0	S
1	S	S	S	1	S
X	S	S	S	X	S
Z	0	1	X	Z	S
S	S	S	S	S	S

図7 WIRD 演算の定義

2.2.2 既存の論理演算の再定義

AND、OR、NOT について、0、1、X、Z、S の5値に対する演算の再定義を行なう。2値論理の場合と比べて新しく定義する部分の特徴は、

- ショートを出力に伝搬させること。
- ハイインピーダンスがこれらの演算の入力に来た時は、出力を不定値 (X) にすること。
- 不定値、ショートの否定をそれぞれ不定値、ショートにすること。

である。

回路がショートする可能性がある場合は、そのパタンを設計者に知らせるために、ショートを出力まで伝搬させることが必要である。また、ハイインピーダンスがゲートの入力に伝搬する場合は明らかに設計エラーであり、出力は不定となる。このような意味を持つ不定値やショートが、自身の否定との AND や OR で消えることが無いように、不定値、ショートは補数を持たないようにする。

5値に対するこれらの演算の定義を 図8 に示す。

2.2.3 条件分岐記述用の論理演算子

従来2値の回路を扱っていた場合は、「IF (条件句) THEN (返却値1) ELSE (返却値2)」という記述を

(条件句) AND (返却値1) OR $\overline{\text{(条件句)}}$ AND (返却値2)

というブール式に変換していた。しかしハイインピーダンスを考慮する場合、AND や OR を使ったブール式では正しい動作を表現することは出来ない。返却値がハイインピーダンスを含む場合に、前述の AND の定義によって、出力が不定値となってしまうからである。

ハイインピーダンスについては、記述の場所 (条件句あるいは返却値) によって扱いを変える必要もある。返却値として記述される場合は、この文で記述する信号の状態を表すものであるが、条件句にハイインピーダンスが記述される場合は、ゲートの入力にハイインピーダンスが不正に伝搬されたものと見るべきだからである。

そこで、新しい演算子、IF_AND、IF_OR を定義し、IF 文の論理を

(条件句) *IF_AND* (返却値 1)
IF_OR $\overline{\text{(条件句)}}$ *IF_AND* (返却値 2)

というブール式で表現する。

IF_AND は、必ず左項を条件句、右項を返却値とし、ハイインピーダンスが左項にある場合と右項にある場合とで出力を変えるように定義する。

IF_AND と IF_OR の演算定義をそれぞれ図9(a),(b) に示す。

AND	0	1	X	Z	S
0	0	0	0	0	S
1	0	1	X	X	S
X	0	X	X	X	S
Z	0	X	X	X	S
S	S	S	S	S	S

(a) AND 演算の定義

NOT	0
0	1
1	0
X	X
Z	X
S	S

(b) NOT 演算の定義

OR	0	1	X	Z	S
0	0	1	X	X	S
1	1	1	1	1	S
X	X	1	X	X	S
Z	X	1	X	X	S
S	S	S	S	S	S

(c) OR 演算の定義

図8 AND,OR,NOT の多値論理演算の定義

IF-AND		右項 0	1	X	Z	S
	0	0	0	0	0	0
左	1	0	1	X	Z	S
	X	X	X	X	X	S
項	Z	X	X	X	X	S
	S	S	S	S	S	S

(a) IF-AND 演算の定義

IF-OR		右項 0	1	X	Z	S
	0	0	1	X	Z	S
左	1	1	1	1	1	S
	X	X	1	X	X	S
項	Z	Z	1	X	Z	S
	S	S	S	S	S	S

(b) IF-OR 演算の定義

図 9

3 多値 BDD のインプリメント

以上のように多値に拡張した BDD をインプリメントする上で、次のような問題があった。

- 否定枝を使用した場合に BDD の論理に対する一意性が保てない。
- 2 値論理用の演算アルゴリズムがそのまま使えない。

3.1 否定枝を使用した場合の一意性の保証

Condor では、否定演算を表す属性を持つ枝 (否定枝)[4][6] を使用する BDD によって論理を表現する。否定枝の使用によって否定演算を大幅に高速化でき、BDD のノード数を削減することができる。ところが、S の否定を S、X の否定を X と定義したために、同じ論理を表す複数の BDD が存在する場合が生じた。

例えば、図 10 の (a)、(b) は形状は異なるがどちらも

$$\overline{(IF\ a\ THEN\ f\ ELSE\ X)}$$

の論理を表す BDD である。(図 10 で、破線で示す枝は、否定枝を表す。)

このような場合に BDD の論理に対する一意性を保つために、「0 枝が S あるいは X を指す場合は 1 枝は否定属性を持てない」というルールを設定する。これと、[7] で提案されている「0 枝には否定枝を用いない」というルールと合わせて、全ての論理に対して BDD の一意性が保証できる。

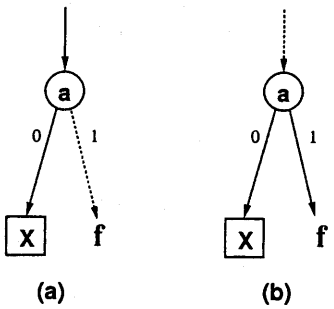


図 10 同じ論理を表す異なる BDD

3.2 演算の効率化

BDD 上の二項演算は、[8] で提案されたアルゴリズムによって行なう。すなわち、2 つの BDD を同時に深さ優先でたどりながら各 BDD ノードのペアに対する演算を再帰的に行なう。従来の 2 値論理の BDD では、この演算の再帰が深くならないようにするために、例えば $F*1$ のように演算の答がすぐ分かる場合に答 F をすぐ返すことによって効率化を図っていた。

ところが、ハイインピーダンス、不定値などの状態値を考慮すると、この効率化がそのまま行なえない。例えば、関数 F の BDD にハイインピーダンスが含まれている場合、$F*1 = F$ とはならない。$Z*1 = X$ であるからである。

そこで、各 BDD に、その BDD がハイインピーダンス、ショート、不定値のどの定数ノードに到達可能かを表すフラグを設定し、このフラグによって可能な効率化演算を判断するようにする。例えば、AND 演算については図 11 に示すような、含まれる定数ノードと可能な演算の関係がある。

	X	Z	S
F*0=0	○	○	○
F*1=F	○	×	○
F*F=F	○	×	○
F*F'=0	×	×	×

図11 定数値と可能な演算の対応

また否定演算は、2値BDDの場合は枝の否定属性の操作で行なっていたが、これもフラグを参照してX、Z、Sを含まないBDDに対してのみ適用する。

4 エラー検出パタンの生成

Condorは、仕様と回路に論理的な差異がある場合(すなわち両者のBDDが不一致である場合)に、それを検出するための入力パタンを自動的に生成する。エラー検出パタンを求めるための二項演算子TPGを新しく次のように定義する。

TPG	0	1	X	Z	S
0	0	1	X	1	S
1	1	0	X	1	S
X	X	X	X	X	X
Z	1	1	X	0	S
S	S	S	X	S	S

図12 TPG 演算の定義

TPGの演算結果は、0,1,S,Xのいずれかである。0は論理一致となる、1は論理エラーとなる入力パタンを意味する。一方がショートとなるパタンはSとし、論理エラーとなるパタンと区別する。また、一方がXの時はTPGの演算結果をXとしている。これは禁止入力パタンでBDDが不一致となる場合を、論理エラーとなるパタンと区別するためである。

TPGのBDDを作った後、定数ノードへのパスを求め、エラー検出パタンとする。定数1、S、Xへの全てのパタンを出力すると後のエラー解析作業の負担が大きくなるので、もっとも本質的なエラーを表すと考えられる、定数1へのパスのみを優先して求める。前述したように、BDDの各ノードは、到達可能な定数ノードの種類を表すフラグを持っており、これを利用してBDDの段数に比例する手間で定数ノードへのパス探索が可能である。

5 まとめ

BDDを用いた形式的論理検証システムを多値論理に拡張する上での課題とそれを解決する方法について提案した。本稿で述べた方法により多値論理回路の取り扱いが可能となり、当社で開発する装置の設計検証へ適用することが可能となった。新しい機能として、回路をショートさせる入力パタンの検出を実現した。複数の論理値の表現を2値の符合化によらず独立したBDDで表し、論理演算の定義が自由にできるため、今後の設計環境に対応した機能拡張も容易に行なえると考えられる。

参考文献

[1] 向山, 他: 論理検証システムCONDOR, 情処研報 Vol.92, No.56, pp17-22, 1992.

[2] Kato S. and Sasaki T.: FDL: A Structual Behavior Description Language, CHDL83(1983), pp.137-152.

[3] 高崎 茂 他: HALIII:機能レベル・ハードウェア・シミュレータ・システム, 情報処理学会論文誌, Jan. 1991, Vol.32 No.1

[4] Akers, S.B.: Binary Decision Diagrams, *IEEE Trans. Comput.*, Vol. C-27, No.6, pp.509-516, 1978.

[5] Bryant, R.E.: Graph-Based Algorithms for Boolean Functional Manipulation, *IEEE Trans. Comput.*, Vol.C-35, No.8, pp.677-691, Aug. 1986.

[6] Madre, J.C. and Billon, J.P.: Proving Circuit Correctness Using Formal Comparison Between Expected and Extracted Behavior, *ACM/IEEE Proc. 25th DAC*, pp.205-210, 1988.

[7] 湊, 石浦, 矢島: 論理関数の共有二分決定グラフによる表現とその効率的処理手法, 情報処理学会論文誌, Vol. 32, No.1, pp. 77-85, 1991.

[8] Brace, K.S., Rudell, R.L. and Bryant, R.E.: Efficient Implementation of a BDD Package, *ACM/IEEE Proc. 27th DAC*, pp.40-45, 1990.