計算機アーキテクチャ　103－7
設　計　自　動　化　69－7
(1993. 12. 16)

# BDDサイズを考慮した回路分割に基づく形式的論理検証手法

中岡 敏博　若林 真一　小出 哲士　吉田 典可

広島大学 工学部
〒724 広島県東広島市鏡山一丁目4番1号

あらまし　VLSIの設計規模の増大に伴い，論理検証の分野では，既存の設計資産の利用容易化のため形式的検証が必要となりつつあり，2分決定グラフ（BDD）の導入により，形式的検証は実用化されつつある．BDDは，論理関数を表すデータ構造であり，データ量の削減，等価判定の容易性等の特長をもつ．そのため，BDDを用いた形式的検証法が数多く報告されている．しかし，これらの多くは，論理関数を表すBDDのサイズの削減のみを考慮しており，論理回路の構造を直接考慮しているものは少ない．従って，本稿では，BDDのサイズを削減するために，回路分割により複数のBDDを作成することにより，与えられた2つの組合せ論理回路の等価判定を行う手法を提案する．

和文キーワード　形式的検証，組合せ回路，BDD，回路分割，generalized cofactor

# A Formal Logic Verification Method Based on Circuit Partitioning Considering BDD Size

Toshihiro NAKAOKA, Shin'ichi WAKABAYASHI,
Tetsushi KOIDE and Noriyoshi YOSHIDA

Faculty of Engineering, Hiroshima University,
4-1, Kagamiyama 1 chome, Higashi-Hiroshima, 724 Japan

Abstract　Due to advances in VLSI technology, the circuit design of VLSI chip becomes complex. For the area of logic verification, formal verification has been neccesary because of flexibility of the design resources. Recently, with Binary Decision Diagram (BDD), formal verification has become practical. BDD is a representation of a boolean function, and it has good characteristics such as small amount of memory size, easiness in equivalence check, etc. Several reseachers have presented formal verification methods based on BDDs. However, many of them don't explicitly take account of the relation between the size of BDD and circuit structure. In this paper, we present a formal verification method based on circuit partitioning which make a number of BDDs so that it reduces the total BDD size.

英文 key words　formal verification, combinational circuit, BDD, generalized cofactor

# 1 はじめに

半導体技術の進歩により，VLSIチップの回路設計規模が増大してきている．このため設計資産の一括管理，再利用の容易化が急務となってきている．論理検証の分野でも，設計した回路に対して開発者が規定したシミュレーションを行うだけでは不十分であり，仕様と実現した回路の形式的検証が必要となりつつある [20]．近年，検証技術は，2分決定グラフ (Binary Decision Diagram)[1, 2] の利用により急速に実用化されつつある．組合せ論理回路の形式的検証は，これまでに，BDDの変数の順序づけの最適化を行う手法 [5, 6, 9, 12] や，拡張したBDDに基づく手法 [13, 16, 17]，回路構造を考慮する手法 [4, 10, 18] 等，様々な提案がなされてきた．しかし，最初の2つのカテゴリに分類される多くの手法はBDDのサイズ（節点数）が大きくなる回路に対する直接的な手法とは成り得ない．また，3番目のカテゴリに属する手法は，一部の回路には有効であると理論的に証明されているが，一般的な回路に有効である保証はないものが多い．

本稿では，回路を分割して複数のBDDを生成し，2つの組合せ回路の等価判定を行う手法を提案する．これにより，BDDのサイズを削減し，実用的な計算時間とメモリ量で実行できる検証法の確立を目指す．

# 2 BDDを用いた検証法

現在のデジタルシステムの設計では，検証法としてシミュレーションと形式的検証とが知られている．シミュレーションでは，与えられたテストパターンについての妥当性の保証を行うが，形式的検証では，与えられた設計が与えられた仕様を満たすか否かを形式的に（数学的に）証明する．設計対象が複雑になるに従い，シミュレーションによる誤りの完全除去は難しくなり，形式的検証が重要となる．デジタルシステムの設計では，2つの回路の動作の等価性を判定する場合と，与えられた仕様を実現された順序回路が満たすかどうか検証する場合とが存在する [8]．本稿では，前者における2つの組合せ論理回路の形式的検証に絞り考察する．

論理回路における多くの問題は，ブール式や論理関数の効率的な処理法に依存しており，論理検証もこれに含まれる．論理関数の表現法としては，これまで，真理値表表現，解析木表現，積和形表現，等が用いられていた．これらの表現法はいずれも，必要記憶量と等価判定の容易性のいずれかに問題点をもつ [11]．

BDDは，非巡回有向グラフによる論理関数の一表現である．BDDは，多くの実用的な回路を実用範囲の記憶量で表現でき，変数順序が固定されれば，論理関数に対する表現が一意に求まるという特長をもつ．このため，論理検証を始め，様々なコンピュータサイエンス分野の応用が考えられている [3, 8]．本稿では回路の論理関数の表現としてBDDを使用する．

## 2.1 従来手法

2つの組合せ回路の等価性を判定するには，2つの回路の出力関数に対応するBDDを作成しそれらが一致するか調べればよい．この場合，重要なことは，どれだけ大きな回路を扱えるかである．これまでに発表されたBDDを用いた検証法としては以下のものが挙げられる．

- 変数順序の最適化手法

  BDDは変数順序に非常にセンシティブであり，変数順序によって扱える回路規模が異なる．しかし，最適な変数順序を決定する問題は計算困難であろうと予測されており [2, 11]，現在のところ変数の数 $n$ に対して $O(n^2 3^n)$ の最適変数順序決定アルゴリズム [7] しか知られていない．このため，様々なヒューリスティク手法が提案されている．

- BDDの機能拡張

  大規模な回路や，これまでのBDDの扱えなかった回路を表現するために，BDD自体の機能拡張を行った手法が幾つか提案されている．例えば，湊等 [17] は，2つ以上のBDDの等価な部分グラフを共有化することにより，複数のBDDのサイズを減少させた共有2分決定グラフ (SBDD) を提案した．このSBDDのもう1つの特長は等価判定が $O(1)$ で行えることである．また，Jeong 等 [14] は，節点に演算機能を持たせたextended binary decison diagram(XBDD) を提案した．

- 回路の構造を考慮した手法

  これらの手法は，理論的に考察されているものが多い．例えば，濱口等 [10] は回路を一次元配置に分割した場合について考察している．この中で，乗算器や隠れ重み付きビット関数については従来よりサイズの小さいBDDで等価判定できることを証明している．また，木村等 [15, 18] は，内部変数を利用した回路の等価判定法を提案している．さらに，Burch[4] は対象の回路を乗算器に絞った手法を提案している．

- その他

  正確には，形式的検証とはいえないが，確率的手法がJain 等 [13] によって提案されている．この手法は決定的手法に比べてBDDサイズを小さくでき，エラー確率もかなり低く押えられる．

## 2.2 提案手法

提案手法は，回路の構造を考慮した手法として考えることができる．本稿の手法の概要は，まず，等価判定用の

回路を構成する．次に，回路を入力変数から探索していきながら，いくつかのブロックにクラスタリングすることで，回路を分割し，同時に，生成される複数のBDDのサイズが小さくなるようにする．そして，これらのBDDから2つの組合せ論理回路の等価判定を行う．回路の出力のBDDを作成しないで，複数のBDDから等価判定を行うことで，使用するBDDのサイズの減少を計っている．なお，本稿ではBDDとしてSBDD[17]を使用している．

## 3 準備

本稿では，2つの1出力組合せ論理回路の形式的検証（等価判定）を議論する．そこで，問題を数式で表現できるように，用語の定義，問題の定式化を行う．さらに，先に述べた回路分割を行うため，回路出力の等価判定問題を別の問題に変換する必要がある．また，問題を変換する過程で，導入する概念についても述べる．

### 3.1 定義

問題の入力となる回路からの情報を以下に示す．

- 入力変数：$x_1, x_2, \ldots, x_n$
- 入力回路：$C_1, C_2$（検証される2つの1出力組合せ論理回路）
- 出力関数：$F_1, F_2$（入力変数$x_1, \ldots, x_n$で表現される$C_1, C_2$の出力関数）

提案手法では入力された回路から等価判定用の回路を構成し（図1），回路を複数のブロック$(B_1, B_2, \cdots, B_k)$に分割し（図2），BDDの作成を行う．これらの操作に用いる変数を以下に示す．

- 合成回路：$C$（図1参照）
- 出力関数：$F$（$F_1, F_2$の排他的論理和，入力変数は$x_1, \ldots, x_n$のみ）
- ブロック:$B_1, \ldots, B_k$（回路分割時における部分回路）
- ブロック中間入力変数：$y_1, \ldots, y_m$
- ブロック中間出力関数：$Z_1, \ldots, Z_m$
- 出力関数：$F_y$（$F_1, F_2$の排他的論理和，入力変数に$y_i (i = 1, 2, \ldots, m)$が含まれる）
- ブロック中間関数：$Y_j = y_j \equiv Z_j \ \ (j = 1, 2, \ldots, m)$
- 分割最小BDDサイズ：$S_p$

これらの記号の関係を示す．回路$C_1, C_2$の出力関数$F_1, F_2$は入力変数$x_1, x_2, \ldots, x_n$を入力とする論理関数であり，$F_1(x_1, x_2, \ldots, x_n), F_2(x_1, x_2, \ldots, x_n)$とも記される．等価判定のために，入力を接続し，$C_1, C_2$の出力の排他的論理和を計算する回路$C$を構成するが，回路$C$の出力関数は$F$または$F(x_1, x_2, \ldots, x_n)$と表現される．以上のように回路を構成すると，$C_1, C_2$の等価判定を行うには，論理関数$F(x_1, x_2, \ldots, x_n)$が恒偽（どんな入力が与えられても論理関数の出力が0）であるか調べるだけでよい．本稿では，BDDのサイズの減少を計るために，回路を図2のようにブロック$B_1, B_2, \ldots, B_k$に分割する．ブロック間を横切るネットは，出力側には論理関数$Z_j (j = 1, 2, \ldots, m)$が割り付けられ，入力側には中間変数$y_j (j = 1, 2, \ldots, m)$が割り付けられる．また，分割後の出力関数が$F$から$F_y$へ変更されているのは，入力変数が$x_1, x_2, \ldots, x_n$だけでなく，$y_1, y_2, \ldots, y_n$も含まれているためである．このため，$F_y(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$とも記述される．$F = F_y$が成立するためには，同一のネットの論理関数$Z_j$と中間変数$y_j$には，$Z_j \equiv y_j$という関係が成立することが必要十分条件となる．即ち，$F(x_1, x_2, \ldots, x_n) = F_y(x_1, x_2, \ldots x_n, y_1 = Z'_1, y_2 = Z'_2, \ldots, y_m = Z'_m)$と記述できる．ここで，$Z'_j = Z_j(x_1, x_2, \ldots, x_n, y_1 = Z'_1, y_2 = Z'_2, \ldots, y_{j-1} = Z'_{j-1}, y_{j+1} = Z'_{j+1}, \ldots, y_m = Z'_m)$は入力を$x_1, x_2, \ldots, x_n$のみとする論理関数である．最後に$S_p$は，分割アルゴリズムを適用するときに，BDDサイズを小さくするために用いられるパラメータである．

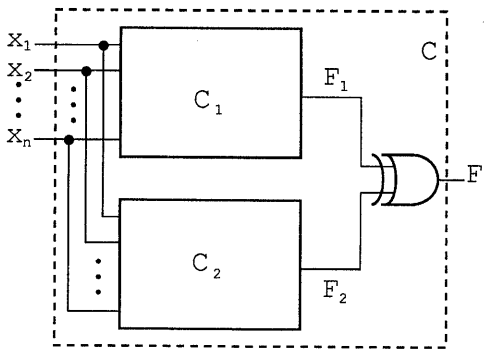


図1 検証回路の構成

論理関数の演算に対する定義を行う．

**定義1** *generalized cofactor[15, 22]*：任意の$n$入力変数論理関数$A, C$が与えられるとする．また，入力変数$(x_1, x_2, \ldots, x_n)$から$(y_1, y_2, \ldots, y_n)$への写像関数$\pi$を導入する．もし，$C$に関して，$\pi_C$が，次の条件

もし，$C(x_1, x_2, \ldots, x_n) = 1$ ならば，

$$\pi_C(x_1, x_2, \ldots, x_n) = (x_1, x_2, \ldots, x_n)$$

もし，$C(x_1, x_2, \ldots, x_n) = 0$ ならば，

$$\pi_C(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$$

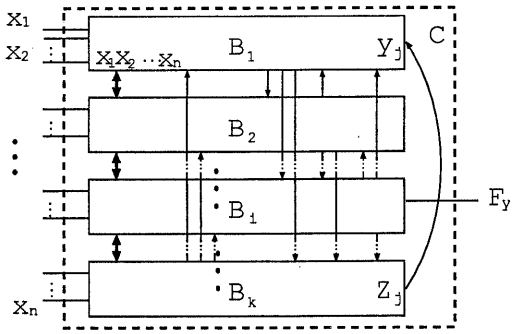


図 2 回路分割とブロック入出力

ただし，$C(y_1, y_2, \ldots, y_n) = 1$.
を満たすならば，$C$に関する $A$ の *generalized cofactor* は，

$$cofact(A, C) = A(\pi_C(x_1, x_2, \ldots, x_n))$$

と定義される． □

generalized cofactor は，1 変数の restrict 演算の拡張とも考えられる．また，不完全に定義された関数 $A \cdot C + \overline{C} \cdot d$（$d$ は任意の関数で $\overline{C} \cdot d$ はドントケアを表す）を一意に求めるヒューリスティク手法としてもみることができる [22]．この演算子は次に示す有用な性質をもつ．

**性質 1** 任意の論理関数 $A, B, C$について以下の性質が成立する *[15, 22]*.

*P1)* $\overline{cofact(A, C)} = cofact(\overline{A}, C)$
*P2)* $cofact(A \cdot B, C) = cofact(A, C) \cdot cofact(B, C)$
*P3)* $cofact(A + B, C) = cofact(A, C) + cofact(B, C)$
*P4)* $cofact(cofact(A, C), C) = cofact(A, C)$
*P5)* $cofact(C, C) = 1$
*P6)* $cofact(C, \overline{C}) = 0$ □

### 3.2 問題の定式化

本研究では 2 つの 1 出力組合せ回路の等価性の判定問題を扱う．

【入力】検証される 2 つの 1 出力組合せ論理回路 $C_1, C_2$，最大使用可能記憶容量（BDD の節点数）

【出力】$F_1, F_2$の等価性の真偽の判定．

【制約】使用記憶容量が与えられた上限以下．

### 3.3 問題の変換

回路を分割しているため，直接 2 つの回路の出力の論理関数だけでは検証はできない．このため，$F \equiv 0$ の問題を回路分割に適用できるように変換しなくてはならない．変換は以下に示す 2 つの式 (1)，(2) に基づいて行われる．各補題，定理の証明は紙面の都合上省略する．詳細は文献 [19] を参照されたい．

$F \equiv 0 \Leftrightarrow F' \equiv 0$ となる $F'$を求める．先の定義から，文献 [10] のように，

$$F' = \left( \prod_{i=1}^{m} y_i \equiv Z_i \right) \cdot F_y \qquad (1)$$

が得られる．

**補題 1** 入力変数 $x_1, x_2, \ldots, x_n$について，式 *1*は成立する． □

さらに，$F' \equiv 0 \Leftrightarrow F'_{(j)} \equiv 0$ となる $F'_j$を求める．これは，generalized cofactor を用いて，次のように表される．

$$F'_{(j)} = cofact(F', \overline{Z_j}) \cdot \overline{y_j} + cofact(F', Z_j) \cdot y_j \qquad (2)$$

**補題 2** 入力変数 $x_1, x_2, \ldots, x_n$，中間出力関数 $Z_j$，中間入力変数 $y_j$について，もし，$y_j = cofact(y_j, Z_j)$ ならば，式 *(2)* は成立する．ここで，$j = 1, 2, \ldots, m$ である． □

**性質 2** 入力変数 $x_1, x_2, \ldots, x_n$，中間出力関数 $Z_j$，中間入力変数 $y_j$について，もし，$y_j = cofact(y_j, Z_j)$ ならば，$cofact(Y_j, \overline{Z_j}) \cdot \overline{y_j} + cofact(Y_j, Z_j) \cdot y_j = 1$ である．ここで，$j = 1, 2, \ldots, m$ である． □

**定理 1** 任意の関数 $F$と回路分割において，$F \equiv 0 \Leftrightarrow F'_{(j)} \equiv 0$ は成立する．

略証： 補題 1,2 から導ける． □

**性質 3** 式 *(2)* の計算を定理の条件を満たすように実行すると，性質 *(2)* から中間関数 $Y_j$の値は *1* となり，$F'_{(j)}$は以下のように展開される．

$$\begin{aligned} F'_{(j)} &= \prod_{i \neq j}^{m} \left\{ cofact(Y_i, \overline{Z_j}) \cdot \overline{y_j} + cofact(Y_i, Z_j) \cdot y_j \right\} \\ &\quad \left\{ cofact(F_y, \overline{Z_j}) \cdot \overline{y_j} + cofact(F_y, Z_j) \cdot y_j \right\} \end{aligned} \qquad (3)$$

□

定理 1では，条件 $y_j = cofact(y_j, Z_j)$ が必要だが，これを除くには次の定理 2のようにすれば良い．

**定理 2** $F'$に関して，任意の $Y_i, Y_j$または $F', Y_j$について，次式が成立する．

$$\begin{aligned} F' &= \left\{ cofact(Y_i, \overline{Z_j}) \cdot \overline{y_j} + cofact(Y_i, Z_j) \cdot y_j \right\} \cdot Y_j \\ &\quad \cdot \prod_{l \neq i}^{m} (Y_l) \cdot F_y \\ &= \left\{ cofact(F_y, \overline{Z_j}) \cdot \overline{y_j} + cofact(F_y, Z_j) \cdot y_j \right\} \cdot Y_j \\ &\quad \cdot \prod_{l=1}^{m} (Y_l) \end{aligned} \qquad (4)$$

□

ここで示した変換式は提案アルゴリズムの等価判定に用いられる．

# 4 提案アルゴリズム

## 4.1 アルゴリズムの概要

アルゴリズムは，前処理，回路分割，等価判定の3つのフェーズから構成されている．まず，回路分割とBDD処理のために，与えられた2つの論理回路から回路$C$を構成し，PIの変数順序付けを行う．次に回路分割では，初期分割としてPI(*Primary Input*)のみ，ブロックに属するようにした上で，PO(*Primary Output*)へ向かって分割手続きを進めることで，各ブロックのBDDを作成しながら，各ブロックにゲートを取り込む．本研究では新たに追加されるゲートに直接接続のあるブロックに含まれるゲートの集合を前線という．回路分割の後，作成されたBDDを合成して2つの組合せ回路の等価判定を行う．

- フェーズ1
  1) 図1のように，回路$C_1, C_2$から排他的論理和を出力とした回路$C$を構成する．
  2) PIの順序付け [9] を行う．
- フェーズ2
  1) PIを分割し変数分割集合（初期ブロック）を作成し，これらを分割における初期の前線とする．
  2) ブロックに追加される次候補のゲートを選出する．(候補の全入力はブロック（前線）に属している)
  3) 選出した候補がどのブロックに追加するか決定し，BDDを作成する．
  4) もし，候補がPOならば終了．そうでなければ2）へ．
- フェーズ3
  1) フェーズ2で得られたBDDを基に，$Y_j(j = 1, 2, \ldots, m)$を表すBDDを作成する．
  2) 式(1),(3),(4)を計算し，$F \equiv 0$であるか判定する．

## 4.2 回路分割（フェーズ2）

フェーズ2では回路$C$をブロックに分割し，各ブロックの入出力を表すBDDを作成する．ここで作成されたBDDの使用サイズは，$F$を表すBDDサイズより小さい．ステップ毎にフェーズ2の内容を述べる．

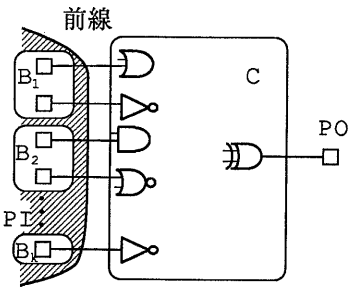


図3 初期分割と前線

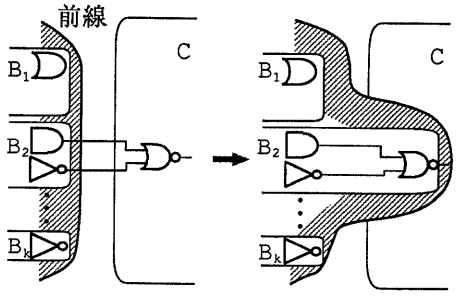


図4 入力が同一ブロックの場合

1) 図3のように，PIのみをブロックに分割し，PIの集合を前線とする．現在，本アルゴリズムでは，分割は入力の順序 [9] によって，等分に$k$分割を行っている．
2) 候補ゲートの選定は幅優先探索で行われる．
3) 候補が属するブロックを決定し，候補のBDDを作成する．このステップは場合1～3の条件に従って処理が実行される．

場合1 全ての候補の入力が，同一のブロックに含まれる場合（図4）

候補は，入力が接続されているブロックに加えらる．また，前線から候補の入力となるゲートを取り除き，新たに候補を加える．候補のBDDは入力のゲートのBDDから演算される．

場合2 候補の入力が，異なるブロックに含まれる場合（図5）

候補は，入力のゲートの中で，一番小さいBDDをもつブロックに加えられる．前線の操作は場合1と同様である．候補のBDDは，入力のゲートのBDDのサイズが$S_p$以上の場合については，中間変数が導入され，そうでない場合は場合1と同様に入力のゲートのBDDか

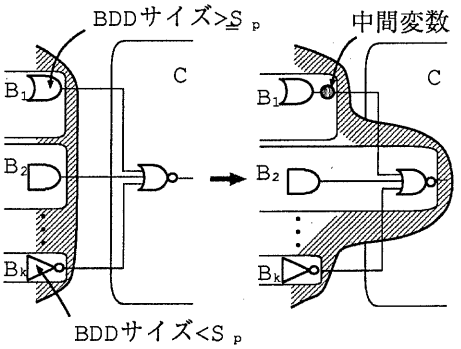


図 5 入力が異なるブロックの場合

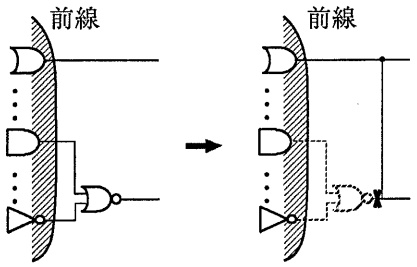


図 6 等価ゲートの削除

ら直接演算される．ここで，全ての入力に対して中間変数を作成しない理由は，中間変数の極端な増加を抑制し，全体の BDD のサイズの増加を防ぐためである．

場合 3 候補の BDD が，前線に含まれるゲートの BDD と同一な場合（図 6）

候補の出力を等価な BDD をもつゲートの出力に接続し，候補を回路 $C$ から切り離す．

### 4.3 等価判定（フェーズ 3）

まず，フェーズ 2 で得られた各ブロックの BDD と中間変数から，$Y_j$ を表す BDD を作成する．次に，$F \equiv 0$ となるとき，またその時に限り恒偽となる関数を計算する．式 (1),(3),(4) に基づいた恒偽性の判定のアルゴリズムを以下に示す．

- 式 (1) に基づいた等価判定アルゴリズム（$ECA1$）

  $F'$ は $Y_j$ と $F_y$ の論理積で表現されるため，これらから 2 つの式を選び，2 項論理積の実行を繰り返して計算する．疑似コードによる記述を図 7 に示す．

- 式 (3) に基づいた等価判定アルゴリズム（$ECA2$）

  補題 2，定理 1 の条件を満たすためには，中間変数を含む全てのゲートにおいて，入力側の優先順位が

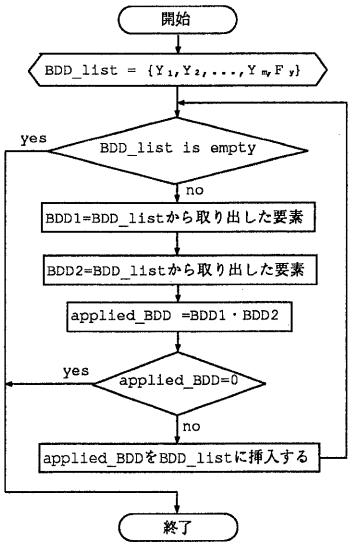


図 7 式 (1) に基づいた等価判定アルゴリズム ($ECA1$)

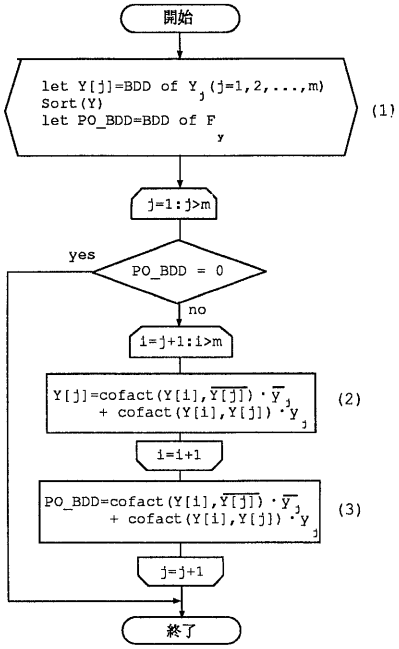


図 8 式 (3) に基づいた等価判定アルゴリズム ($ECA2$)

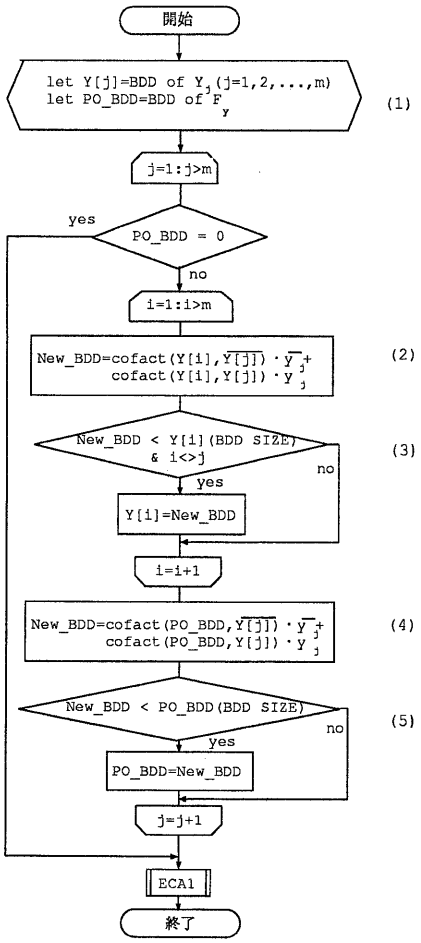


図 9 式 (4) に基づいた等価判定アルゴリズム ($ECA3$)

高くなるようにトポロジカルソートを行い，この順序（図 8 (1)）に従って $F'_{(j)}$の計算（図 8 (2),(3)）を進めればよい．$Y_j$がソート済みとすると，$i,j = 1,2,\ldots,m$ に関して，

$$Y_i \leftarrow \begin{cases} cofact(Y_i, \overline{Z_j}) \cdot \overline{y_j} \\ \quad +cofact(Y_i, Z_j) \cdot y_j & (i > j) \\ 1 & (i = j) \end{cases}$$

$$F_y \leftarrow cofact(F_y, \overline{Z_j}) \cdot \overline{y_j} + cofact(F_y, Z_j) \cdot y_j$$

を繰り返して計算すると最終的に $\forall_j[Y_j = 1]$ と変換でき，出力関数 $F_y$ の恒偽性さえ調べればよい．

- 式 (4) に基づいた等価判定アルゴリズム（$ECA3$）

  式 (1) と式 (3) の手法を組み合わせた方法を考える．フェーズ 3 では中間関数と PO の論理積の等価判定を行うが，論理積を実行する前に，generalized cofactor を用いて，全体の BDD のサイズを小さくすることを試みる．即ち，各中間関数において，generalized cofactor を用いた演算結果の BDD（図 9 (2),(4)）サイズが演算前よりも小さければ（図 9 (3),(5)），中間関数を表す BDD は演算結果の BDD に変換される．もし，そうでなければ変更しない．この操作を全ての中間関数$Y_j$について繰り返す．そして最後に，変換された $F'$を $ECA1$ で等価判定を行う（図 9）．

## 5 計算機実験

前節で提案したアルゴリズムについて計算機実験を行った結果について述べる．アルゴリズムは，サンマイクロシステムズ社の SPARC station2（28．5MIPS）上で，C 言語を用いて実現し，BDD 用パッケージとしては，文献 [17] の SBDD パッケージ ver 6.0 を使用した．データは，ISCAS'85 のベンチマークデータを使用した．本手法では 2 つの組合せ回路の比較を行うため，SIS[21] を使用してベンチマークの回路を簡単化し，元のベンチマークの回路と簡単化された回路を入力回路として与えた．

実験では，各データに対して，フェーズ 3 での等価判定手法として $ECA1$, $ECA2$, $ECA3$ を適用した．また，比較データとして，回路の入力変数に対する出力論理関数 (PO) の BDD のサイズを示す．実験結果を表 1 に示す．表 1 のように，約半分のデータで $ECA3$ の BDD サイズが 1 番少ないことがわかる．しかし，比較的大きなデータに対しては結果がよいとはいえない．これから，さらに BDD サイズを減らすための等価判定手法の改良が必要だと思われる．

## 6 あとがき

本稿では，BDD サイズを考慮しながら回路分割を行った後に，generalized cofactor と論理積を用いた等価判定を行う手法について考察した．また，計算機により本手法について実験的評価を行った．今後の課題としては，提案した等価判定に対してより有効な回路分割手法を検討すること等が挙げられる．

## 文献

[1] S. B. Akers: "Binary decision diagrams", IEEE Trans. on Computers, **Vol. C-27**, No. 6, pp. 509 – 516 (1978).

[2] R. E. Bryant: "Graph-based algorithms for boolean function manipulation", IEEE Trans. on Computers, **Vol. C-35**, No. 8, pp. 677 – 691 (1986).

[3] R. E. Bryant: "Symbolic boolean manipulation with orderd binary-decision diagrams", ACM Computing Surveys, Vol. 24, pp. 293 – 318 (1992).

[4] J. Burch: "Using BDDs to verify multipliers", Proc. Design Automation Conference, pp. 408 –

表 1 実験結果

| | BDD サイズ（節点数） | | | |
|---|---|---|---|---|
| データ | PO | ECA1 | ECA2 | ECA3 |
| c1355 | 2286 | 3466 | 6737 | 1274 |
| c1908 | 1357 | 1364 | 1518 | 620 |
| c2670 | 1501 | 370 | 370 | 225 |
| c3540 | 73252 | 580408 | 67184 | N.F. |
| c432 | 39325 | 84554 | 84554 | 62115 |
| c5315 | 53833 | 143163 | 169632 | 47751 |
| c6288 | 106937 | N.F. | N.F. | N.F. |
| c7552 | 831 | 3597 | 2911 | 3028 |

N.F.: 実行不可（BDD サイズ＞1000000）


412 (1991).

[5] K. Butler, D. E. Ross, R. Kapur and M. R. Mercer: "Heuristics to compute variable orderings for efficient manipulation of ordered binary decision diagrams", Proc. Design Automation Conference, pp. 417 – 420 (1991).

[6] P.-Y. Chung, I. N. Haji and J. H. Patel: "Efficient variable ordering heuristics for shared ROBDD", Proc. Int'l Symp. on Circuits and Systems, pp. 1690–1693 (1993).

[7] S. J. Friedman and K. J. Supowit: "Finding the optimal variable ordering for binary decision diagrams", IEEE Trans. on Computers, **Vol. C-39**, No. 5, pp. 710 – 713 (1990).

[8] 藤田 昌宏, E. M. Clarke："BDD の CAD への応用", 情報処理, **第 34 巻**, 5 号, pp. 609 – 616 (1993).

[9] M. Fujita: "Variable ordering algorithms for ordered binary decision diagrams and their evaluation", IEEE Trans. on Computers, **Vol. 12**, No. 1, pp. 6 – 12 (1993).

[10] 濱口, 矢島："組合せ回路の等価判定問題の計算量と回路構造の関係について", 電子情報通信学会春季大会講演論文集, No. SA-2-3, pp. 1–370 – 1–371 (1993).

[11] 石浦 菜岐佐："BDD とは", 情報処理, **第 34 巻**, 5 号, pp. 585 – 592 (1993).

[12] N. Ishiura, H. Sawada and S. Yajima: "Minimization of binary decision diagrams based on exchanges of variables", Proc. Int'l Conference on Comput.-Aided Des., pp. 472 – 475 (1991).

[13] J. Jain, M. Abadir, J. Bitner, D. S. Fussell and J. A. Abraham: "IBDDs: An efficient functional representation for digital circuits", Proc. European Design Automation Conference, pp. 440 – 446 (1992).

[14] S.-W. Jeong, B. Plessier, G. Hachtel and F. Somenzi: "Extended BDD's: trading off canonicity for stucture in verification algorithms", Proc. Int'l Conference on Comput.-Aided Des., pp. 464 – 467 (1991).

[15] S. Kimura: "Equality checking of two logic circuits with different internal variables using BDD's", Proc. Int'l Workshop on Logic synthesis (1993).

[16] Y.-T. Lai and S. Sastry: "Edge-valued binary decision diagrams for multi-level hierarchical verification", Proc. Design Automation Conference, pp. 608 – 613 (1992).

[17] S. Minato, N. Ishiura and S. Yajima: "Shared binary decision diagram with attributed edges for efficient boolean function manipulation", Proc. Design Automation Conference, pp. 52 – 57 (1990).

[18] 中江, 木村："BDD を用いた異なる内部変数を持つ論理回路の等価判定", 第 47 回（平成 5 年後期）情報処理学会全国大会講演論文集, pp. 6–137 – 6–138 (1993).

[19] 中岡, 若林, 小出, 吉田："BDD を用いた形式的論理検証の一手法", Technical Report No. 93-26, Hiroshima Univ. ECS (1993).

[20] 中山 俊一："BDD 技術が実用に，完全な論理検証で ASIC 設計資産の再利用促進", 日経エレクトロニクス, No. 571, pp. 107 – 118 (1993).

[21] E. M. Sentovich, K. J. singh, L. Lavagno, et al.: "SIS: A system for sequential circuit synthesis", Electronics Research Laboratory (1992).

[22] H. J. Touati, H. Savoj, B. Lin, R. K. Brayton and A. Sangiovanni-Vincentelli: "Implicit state enumurations of finite state machines using BDD's", Proc. Int'l Conference on Comput.-Aided Des., pp. 130 – 133 (1990).