# プレスブルガー文真偽判定アルゴリズムのためのBDDの応用とそれを用いた回路検証

景山洋行† 北道淳司† 船曳信生†

我々は，回路設計における高位の形式的設計検証法を提案し，提案する検証法のための検証支援システムを作成し，例題回路の設計検証による評価を行っている．検証支援システム内部では，検証問題をいくつかの部分問題に分割し，各証明問題を Presburger 文と呼ばれる整数上の加減算，大小比較などを含む制約論理式の真偽判定問題に帰着させ，それらを Presburger 文の真偽判定ルーチンで判定するという手順で検証を行う．本論文では，Presburger 文の真偽判定のための新しいデータ構造とその処理系を提案し，それを回路検証に適用した結果について述べる．

# Operations of Presburger Arithmetic Using BDD-like Data Structures and Circuits Verification with them

HIROYUKI KAGEYAMA,† JUNJI KITAMICHI† and NOBUO FUNABIKI†

We have proposed a formal design verification method for logic circuits and have implemented a verification support system for our method. Our verification support system divides the decision problem for design correctness of the implementation circuit into some decision problems written in Presburger arithmetic, and decides each problem with a decision procedure for Presburger arithmetic. In this paper, we propose new data structures for Presburger arithmetic, and implement the algorithms for the data operations over the proposed data structures. We describe the results of some experiments of high level design verification using our methods.

## 1. はじめに

設計する回路が大規模化するにつれ，設計は論理設計レベルからより高位である方式・機能設計レベルから行われるようになりつつある．設計に対する検証も，方式・機能設計レベルにおける手法が望まれている．我々は，設計の高位における形式的設計検証手法として，代数的手法を用いた方法を提案してきた[1),2)]．

本論文では，我々の回路検証支援系[1)]において用いられるプレスブルガー文（Presburger sentence）[3)]と呼ばれる整数上の加減算，大小比較などを含む論理式の真偽判定系の省メモリ化および高速化を目指した新しいデータ構造の提案を行い，それを用いて作成した判定系について述べ，いくつかの回路検証のためのベンチマークによる評価を行う．今回作成した判定系は，まだ改良の余地が残されているが，大規模な回路検証において従来の判定系[4)]よりも高速に判定できる場合がある．

提案するデータ構造は，論理関数を簡潔に表現することが可能であり関数への処理を高速に行える BDDs（Binary Decision Diagrams，二分決定グラフ）[5),6)]を応用したものである．提案するデータ構造では，プレスブルガー文の真偽判定アルゴリズムの 1 つである Cooper のアルゴリズム[3),7)]の実行において大量に生成される部分式を共有し，消費するメモリを削減できるように工夫している．

以下，2 章では，プレスブルガー文の定義と従来行われてきたプレスブルガー文に対する処理アルゴリズムの概要について，3 章では，今回提案するデータ構造とそれに対する処理について，4 章では，例題回路の検証実験によって提案するデータ構造を用いて実現されたプレスブルガー文の真偽判定系の評価を行う．

## 2. プレスブルガー文の処理に関する関連研究

近年，ディジタル回路の論理設計レベルでの形式的検証では，論理関数を BDD を用いて表現し論理関数

† 大阪大学大学院基礎工学研究科情報数理系専攻
Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University

に対する処理を BDD に対する高速な処理アルゴリズムによって実現するという方法が用いられる場合が多い．また，算術演算を含む回路など記述対象によっては，BMDs (Binary Moment Diagrams)[8),9)] などが用いられることもある．

順序回路の設計に対する回路検証においては，データパスと制御部からなる回路モデルを採用することが多く，より抽象度の高いレベルでは，たとえば，データパスにおけるデータレジスタの型として整数型のものを用いるほうが自然に記述できるし，データ転送や実行条件判定における演算として整数に対する加減算や大小比較などを用いるほうが自然に記述できると考えられる．また，パラメータ付きの回路記述に対する回路検証においても，整数型の資源を用いた記述および検証が行われる．これらの検証においては，整数型のデータタイプを持つ変数，定数および演算等からなる論理関数の処理が必要である．

本章では，整数型の変数，加減算および大小比較などの演算を含む論理式の 1 つであるプレスブルガー文の定義およびプレスブルガー文に対する処理アルゴリズムとして知られている Cooper のアルゴリズムについて述べる．

まず，本論文で取り扱うデータタイプおよび演算のクラスであるプレスブルガー文の定義を行う．

**定義 1** プレスブルガー文☆

プレスブルガー文を構成する終端記号として， true, false, $\wedge$, $\vee$, $\neg$(not), (, ), +, −, $\forall$, $\exists$, =, <, 0, 1, 2, ..., $x_1, \ldots, y_1, \ldots$ が許される．ここで $x_1, \ldots$ は整数変数であり，$y_1, \ldots$ は論理変数とする．限定子 $\forall$ および $\exists$ によって束縛された変数を束縛変数，束縛されていない変数を自由変数と呼ぶ．自由変数の存在しない論理式 $\varepsilon$ のことをプレスブルガー文と呼ぶ．

整数を値域とする項 $\tau$ は以下のように定義される．

(1–1) $\tau ::= x_1, \ldots, 0, 1, 2, \ldots$

(1–2) $\tau ::= (\tau + \tau), (-\tau)$

論理式 $\varepsilon$ は以下のように定義される．ただし $z$ は整数変数または論理変数である．

(2–1) $\varepsilon ::= y_1, \ldots, true, false$

(2–2) $\varepsilon ::= (\tau < \tau), (\tau = \tau)$

(2–3) $\varepsilon ::= (\varepsilon \wedge \varepsilon), (\varepsilon \vee \varepsilon), (\neg\varepsilon)$

(2–4) $\varepsilon ::= (\forall z\varepsilon), (\exists z\varepsilon)$ □

以下では簡単のため，自明な括弧 "(" および ")" を省略し，整数変数 $x$ を用いた式 $x + x$ を $2x$ というように整数定数の係数を用いて表現する．また，表記上の便利のため，$\alpha > \beta$ および $\alpha \leq \beta$ を，それぞれ $\beta < \alpha$ および $\alpha - 1 < \beta$ の代わりに用いるなど，定義 1 での複数の構文を組み合わせた記号も用いる．

定義 1 における式の構成法により構成され，自由変数を含むものをプレスブルガー関数と呼ぶ．

プレスブルガー文は，その真偽を決定することが可能である．プレスブルガー文の真偽判定のためのアルゴリズムとして，Cooper のアルゴリズム[3),7)]，Fourier-Motkin の変数消去に基づくアルゴリズム[10),11)]，式を表す構造としてのオートマトンを用いた手法[12),13)] 等が知られている．プレスブルガー文の真偽判定は，式長を $n$ とするとその計算量の下限として非決定的アルゴリズムを用いたとしても $O(2^{2^{cn}})$ ($c$ は定数) を必要とすることが知られている[14)]．

プレスブルガー文の真偽判定には膨大な計算時間を必要とするので，そのサブクラスを定め，それに対する効率的なアルゴリズムの提案が行われている．たとえば，文献 4), 15) および 16), 17) では，すべての変数が冠頭で全称限定子 $\forall$ で束縛されている文を対象としており，それぞれ，Cooper のアルゴリズム[4)]，Fourier-Motkin の変数消去に基づくアルゴリズム[15)]，Shostak の SUP-INF 法[16),17)] を採用している．また，文献 18) では，文中に 2 つの存在限定子 $\exists$ が存在する文に対する高速な判定アルゴリズムを提案している．

これらの処理アルゴリズムの形式的回路検証への応用は，たとえば，回路の設計検証のための定理証明システムでの一証明機能としての使用[1),17)]，時刻を表す変数として整数変数を用いたタイミング検証への応用[15)]，記号モデル検査法[19)] のレジスタとして整数を保持するものを許すというような拡張[20)] など，さまざまな検証対象に対して行われている．

本論文では，プレスブルガー文の真偽判定を行う Cooper のアルゴリズムの効率的な実行を目標としたデータタイプを提案するが，それに先立ち，Cooper のアルゴリズムについて述べる．

Cooper のアルゴリズムが対象とするプレスブルガー文では，定義 1 に加えて以下の構成の論理式も含む．

(2–5) $\varepsilon ::= (n|\tau)$

ただし，$n$ は整数定数である．$n|\tau$ は項 $\tau$ が整数 $n$ で割り切れることを意味する．

ここでは，整数変数 $x$ を引数に含む任意のプレスブルガー関数 $F(x)$ に対して，$\exists xF(x)$ から変数 $x$ を含まずかつ論理的に等価な論理式への変換アルゴリズムを述べる．以下では，$x$ 以外の自由変数については省略する．式 $\forall xF(x)$ に関しても，$F(x)$ に否定を施し

☆ もともとのプレスブルガー文には論理変数は含まれていないが，以下での議論では，このように拡張された定義のもとで議論をすすめる．

た関数に対して，この変換アルゴリズムを適用し，得られた式に，さらに否定を施すことによって，$\forall x F(x)$ から変数 $x$ を含まずかつ論理的に等価な論理式へ変換することができる．論理変数 $y$ の場合は，$\exists y F(y)$ および $\forall y F(y)$ を，それぞれ $F(\mathrm{true}) \vee F(\mathrm{false})$ および $F(\mathrm{true}) \wedge F(\mathrm{false})$ と変換する．プレスブルガー文は，すべての変数に対し限定子が施されるので，限定子が冠頭となるような部分式に対し，この変換アルゴリズムを繰り返し適用することにより，論理的に元の文と等価でありかつ限定子および限定子の施された変数の存在しない文に変換することができる．得られた式の各部分項あるいは部分論理式には，変数が存在しないので具体的な整数値あるいは論理値を求めることができ，文全体の真偽を判定することができる．

Cooper のアルゴリズムは，直感的には，$\exists x F(x)$ に対して，$F(x)$ が真となりうる可能性のある整数関数を $F(x)$ から抽出し，それぞれを $x$ に代入し，代入されたものすべての論理和を求めるというアルゴリズムである．

**Cooper のアルゴリズム**

整数変数 $x$ を含む関数 $F(x)$ に対して，

（**手順 1**） $\neg$ および $=$ を含まない形に変形する

否定は，$\neg(p < q)$ を $q < p+1$ に，$\neg(p \wedge q)$ を $(\neg p) \vee (\neg q)$ に，$\neg\neg p$ を $p$ に，等式 $p = q$ は，$(p-1 < q \wedge q < p+1)$ というように，各変換ルールを繰り返し適用し，これらを消去する．ただし，$n|p$ および $\neg(n|p)$ はそのままの形とする．$x$ を含む論理式は，$p < q$，$n|p$ および $\neg(n|p)$ の形をしたもののみである．

（**手順 2**） 変数 $x$ の係数を最小公倍数でそろえる

$F(x)$ において変数 $x$ のすべての係数を求め，それらの最小公倍数を求め，$H$ とする．$F(x)$ における $x$ のすべての係数が $H$ となるように，$p < q$ および $n|p$ の形の論理式の両辺に適当な数をかける．得られた式の $Hx$ を $X$ と置き換え，それと $H|X$ との論理積を求め，得られた式を $F(X)$ とする．$F(X)$ を真とするような $X$ が存在したとしても，その $X$ が $H$ で割り切れない場合，$Hx = X$ を満たすような整数 $x$ は存在しない．よって，$\exists x F(x)$ と $\exists X F(X)$ は，それぞれの自由変数に同じ値を与えたときにそれぞれ同じ値をとる論理的に等価な関数である．

（**手順 3**） 変数 $X$ に代入する候補を求める

手順 2 で得られた $F(X)$ において，$X$ を含む $(\alpha_i < X), (X < \beta_j), (p_k|\delta_k), \neg(q_l|\theta_l)$ の形の論理式を列挙する．$\alpha_i$，$\beta_j$ は，変数 $X$ を含まない整数変数，整数定数からなる式である．$p_k$，$q_l$ は整数定数である．

（**手順 4**） 代入されたすべての式の論理和を求める

すべての $p_k$，$q_l$ の最小公倍数を求め，$N$ とする．以下の (1) および (2) において求められる各式の論理和を求める．

(1) $F(X)$ の $(\alpha_i < X)$ を真に，$(X < \beta_j)$ を偽に置き換えた式 $F'(X)$ の $X$（式 $\delta_k$ と式 $\theta_l$ 中に現れる）にそれぞれ 0 から $N-1$ の値を代入した式 $D_1, \ldots, D_N$

(2) 各 $\beta_j$ に対して，$F(X)$ の $X$ に，$\beta_j - 1$ から $\beta_j - N$ の値を代入した式 $D_{\beta_{j_1}}, \ldots, D_{\beta_{j_N}}$

以上の手順により求められた関数が，$\exists x F(x)$ と論理的に等価でかつ変数 $x$（および $X$）を含まない関数である． □

手順 4 では，代入の候補として，$X$ に適当な大きな値を考え（その場合，$(\alpha_i < X)$ が真に，$(X < \beta_j)$ が偽となる），そのときに真となるかどうか（(1) の候補），さらにそこから $X$ の値を小さくしていった場合，$F(X)$ が真となりうるのは，$X$ が $\beta_j - 1$ のいずれかの値となった場合（(2) の候補）である．ただし，$F(X)$ には，部分論理式として $(p_k|\delta_k)$ と $\neg(q_l|\theta_l)$ を含んでおり，これらの論理式の真偽のとりうる組合せすべてを調べるために，上述の候補からさらに 0 から $N-1$ まで連続した数を減算した値も候補として加える必要がある．

上述の手順では，$X$ の候補として適当な大きな数から値を小さくすることを考えたが，逆に小さな数から値を大きくすることを考えるということもできる．限定子ヨの消去アルゴリズムとしてはいずれか一方を実現すればよい．

## 3. 提案するデータ構造とその処理系

本章では，定義 1 のプレスブルガー文の真偽判定系のためのデータ構造を提案する．提案するデータ構造は，プレスブルガー文の任意の部分式（すなわち，プレスブルガー関数）を表現することができる．処理系は，定義 1 での各構成法それぞれに対して，データ構造に対する処理アルゴリズムを実現している．

### 3.1 プレスブルガー関数を表現するデータ構造

提案するプレスブルガー関数を表現するデータ構造は，定義 1 の構成法 (1–1)，(1–2)，(2–2) および (2–5) によって構成される式を表現するための整数部分と，それ以外の部分を表現するための論理部分からなる．限定子を含むプレスブルガー関数は，2 章で述べた Cooper のアルゴリズムにより論理的に等価でかつ，

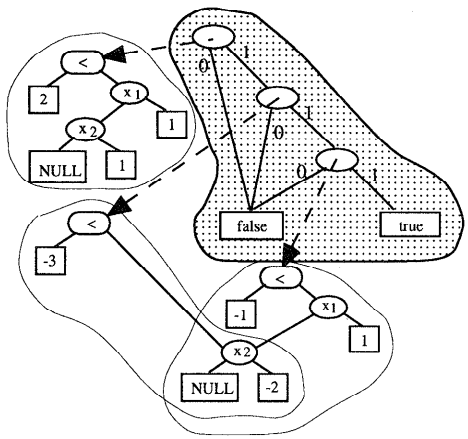



図 1 提案するデータ構造の一例
Fig. 1 An example of our proposed data structure.

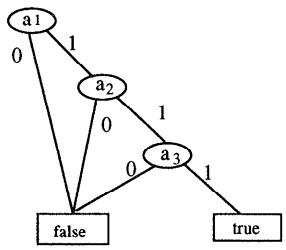



図 2 BDD の例
Fig. 2 An example of BDD data structure.

限定子および束縛変数を消去した後のデータ構造として表現される．

例を用いて提案するデータ構造を説明する．図 1 は，関数 $F(x_1, x_2) = (2 < x_1 + x_2) \wedge (2x_2 < 3) \wedge \neg(x_1 - 2x_2 < 0)$ を，提案するデータ構造で表現したものである．

論理部分は，一般に用いられている BDD と同様の構造である．図 1 においては，論理部分を表現する部分は灰色で囲まれている部分であり，$a_1 \wedge a_2 \wedge a_3$ を表現した BDD と同様の構造となっている（図 2）．ここで $a_1$，$a_2$ および $a_3$ はそれぞれ部分論理式 $2 < x_1 + x_2$，$-3 < -2x_2$ および $-1 < x_1 - 2x_2$ に対応する．また，否定枝などを用いて頂点の数を減らすことが可能である．

整数部分を表現するデータ構造の根は $=$，$<$ および $|$ の演算子のいずれかを示しており，根が $=$ および $<$ の場合，根の右側の頂点以降が表すデータ構造が整数変数の多項式，左側の頂点が定数項を表す．整数部分は，論理部分と同様に頂点の共有を行っており，すべての共通グラフは共有される．図 1 では，式 $-2x_2$ が共有されている．整数定数の共有も行うが図 1 では省略されている．整数部分は，整数変数への代入により整数変数がすべて消去された場合かつそのときのみ，それぞれの根が持つ演算子の意味定義に従って，整数部分を表すデータ構造全体を true または false と変換する．

2 章で述べた手順と比較すると，提案するデータ構造およびその処理アルゴリズムでは以下の特徴がある．

(1) データ構造において $\neg$ は消去されない
(2) データ構造において $=$ は消去されない
(3) 共通のデータ構造は共有する

特徴 (1) については，BDD が否定を自然に表現でき，表現するデータ構造を簡潔に表現できる可能性により，本データ構造では否定記号を消去しない．一方，Cooper のアルゴリズムでの手順 3 における代入の候補をデータ構造から選び出す際には，探索経路上の否定枝の数などから，$>$ および $=$ を表すノードが，それぞれ $\alpha > \beta$ か $\neg(\alpha > \beta)$ および $\alpha = \beta$ か $\neg(\alpha = \beta)$ のいずれかを表しているかを決定している．

特徴 (2) は，Cooper のアルゴリズムの手順 4 における代入の候補を削減するためのものである．$X = \alpha$ の形の部分論理式は，代入する候補として $\alpha$ のみを考えればよい．また，入力によっては，処理中に必要となるノード数を削減できる．

特徴 (3) は，BDD の特徴である．特に，整数定数部分と整数変数のリスト構造に分けている理由は，Cooper のアルゴリズムの手順 4 において，変数 $X$ に対して連続した値の整数関数が代入され定数部分のみ異なり整数変数のリスト部分が同じである構造が大量に生成されるからであり，本データ構造ではそれらの中で共通の整数部分のノードはすべて共有され，使用メモリの削減が行える．

提案するデータ構造では，表現すべきプレスブルガー関数に対して複数のデータ表現が存在する．たとえば，$(x > 10) \wedge (x > 15)$ と $(x > 15)$ は等価なプレスブルガー関数であるが，本データ構造では異なる表現を持つ．したがって，本データ構造は BDD の有する性質の 1 つである，表現すべき論理関数の表現の一意性は有していない．

### 3.2 提案するデータ構造に対する処理アルゴリズムとその処理系

本節では，提案するデータ構造に対して，プレスブルガー文で許される各演算に対する処理とその実現について述べる．

論理部分に対する処理（$\wedge$，$\vee$ および $\neg$ に対する処理）は，BDD と同じ ITE[21] と呼ばれる 3 引数の if-then-else 関数に対するデータ構造への処理アルゴ

リズムを用いて実現される．特に整数部分へのポインタを保持している内部ノードに対しても，論理変数を表すノードと同じ処理が実行される．

論理変数$y$に対する限定子$\exists yF(y)$および$\forall yF(y)$に対する処理は，それぞれ$F(\text{true})\vee F(\text{false})$，$F(\text{true})\wedge F(\text{false})$を求めるのに必要なコファクタ演算，論理和，論理積などの処理アルゴリズムが実行され，対応するデータ構造が生成される．

算術演算（+，−，>，=など）に対する処理は，拡張されたBMD上において，各演算に対する整数リストに対するノードの挿入，削除，リストの探索など基本処理アルゴリズムの組合せによって実現されている．$\leq$，$\geq$などもすべて$\alpha<\beta$（ただし$\alpha$は整数定数）の形に変形される．

存在限定子∃に対する処理では，2章で述べたCooperのアルゴリズムが適用される．全称限定子∀に対する処理では，¬，∃，¬の順にそれぞれに対する処理が行われる．

ChenのBDDとBMDの統合パッケージであるBXDライブラリ[22]を元に，提案するデータ構造を処理するライブラリを作成した．

BXDライブラリにおけるBMDの部分がデータ構造における整数部分の保持を行うが，終端ノードは整数定数を保持できるように拡張している．BMDでは変数どうしの乗算が表現できるが，本データ構造では線形リストの形のものしか扱わない．

BXDライブラリがC言語で約14000行であるのに対し，本ライブラリ全体では約16000行となった．ライブラリで扱える変数の数は約30000である．

BDDは変数順序によりその表現の大きさが著しく変わることが知られている[5]が，本ライブラリでは，構文解析において出現する順に変数順序を与えており，変数順序に関する効率化手法はまだ取り入れていない．

次に，本ライブラリを用いてプレスブルガー文の真偽判定を行うプログラム（以下判定系と呼ぶ）を作成した．プレスブルガー文の真偽判定を行うプログラムは，入力に対する構文解析を行い，ライブラリを用いて入力に対するデータ構造を構築する．構文解析終了時に，入力のプレスブルガー文に対するデータ構造はtrueノードあるいはfalseノードのみとなっており，それぞれ入力が真あるいは偽と判定する．構文解析およびそれからライブラリを呼び出すプログラムはyaccを用いて約1000行程度で実現した．

**図3**に，図1の例の構成の手順を示す．入力を文頭から順にたどりながら時点(a)，(b)および(c)におけるデータ構造をそれぞれ，図3(a)，(b)および(c)に示す．時点(a)では，整数部分の定数部と読み込まれた変数$x_1$とその係数1までのグラフがそれぞれ生成される．時点(b)では，最初の整数部分が読み込まれ，その部分を指す論理変数ノードが生成されている．時点(c)では，次の整数部分の解析が終わっている時点で，2個目の整数部分のデータ構造が生成されている．さらに最後まで処理が進むと，図1が得られる．

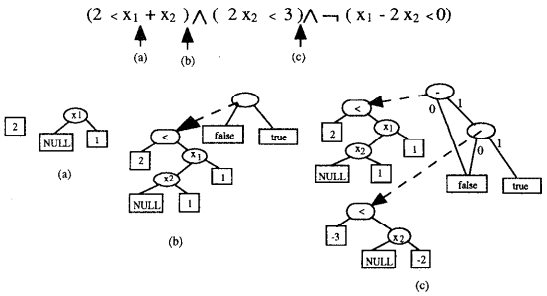



図3 データ構造の構成手順の例
Fig. 3 An example of process of building data structure.

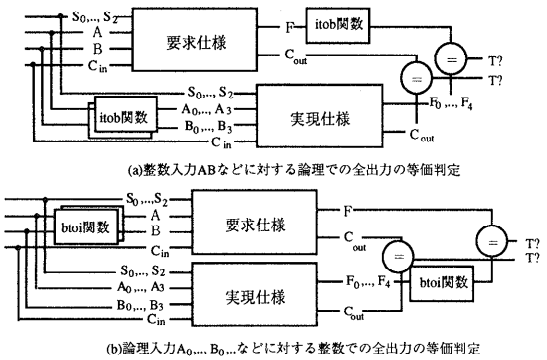
また，本データ構造では，以下のように式を簡単化し変数を消去する場合がある．たとえば，$(2<x_1+x_2)\wedge\neg(2<x_1+x_2)$の場合，$(2<x_1+x_2)$の部分のデータ構造は共有化され，整数部分を指すノードはそれぞれ同じ変数番号を持つ．したがって，引き続いて実行される論理積の処理を実行した後，式全体のデータ構造が得られるが，それはfalseノードのみとなる．この例では，もし，2章で述べた手順に従った場合，最初に否定記号が消去され，式$(2<x_1+x_2)\wedge(x_1+x_2<3)$が得られ，変数$x_1$，$x_2$に対する限定子の処理が行われるまでこの式をfalseと判定できない．

## 4. 提案手法の評価実験

提案手法の評価のための検証実験として，以下で述べる2つの組合せ回路の設計検証を行った．ここでは，4 bit ALU（TTL74382）とHLSynth95ベンチマークセットのfp-addの一部の検証を取り上げる．

4 bit ALUの検証について述べる．入出力が整数変数で記述された要求仕様と，それをCADで論理合成した結果（imp1）や規格表を参考に人手で書き下した実現（imp2）の入出力がすべて論理変数で記述されるレベルである実現仕様との出力等価の検証を行う．

要求仕様は，データ出力や各種出力フラグそれぞれに対し，論理変数の機能選択入力や整数のデータ入力からなるプレスブルガー関数との等式として表される．実現仕様は，74382を実現する具体的ゲートなどからなるネットリストを，内部頂点も含めて論理式に変換したものを用いる．したがって，入出力および内部端




(a)整数入力ABなどに対する論理での全出力の等価判定




(b)論理入力A0,... B0...などに対する整数での全出力の等価判定

**図 4** 74382 の検証の概要
Fig. 4 Outline of verification for 74382.

点を表す論理変数とそれらの間の論理演算からなる．これらの各記述の詳細は，文献 23) で述べられている．

これらの要求仕様と実現仕様 impl1 が，各変数の組合せに対して，全出力が同じ値を持つことを検証する．

**図 4** (a) および (b) は，要求仕様および実現仕様 impl1 に対して，それぞれ同じ入力を与えたときに出力が同じであるかという出力等価の検証を表している．itob および btoi 関数は，それぞれ，整数入力を論理出力に，論理入力を整数出力に変換する関数である．(b) に対応する証明すべき式は以下のような形をしている．要求仕様および実現仕様 impl1 は，出力と入力との関係を等式で表しており，それらの関係を前提として実現仕様と要求仕様の出力が等しいということを表している．ただし，$a$ imply $b$ は，$(\neg a) \vee b$ を表す．

∀ 全変数
((要求仕様
　∧
　実現仕様 impl1
　∧
　要求仕様の入力=btoi 関数（実現仕様の入力）
　∧
　btoi 関数の出力=btoi 関数)（実現仕様 impl1 の出力）
　imply
　　btoi 関数の出力=要求仕様の出力)　　(1)

ただし，(a) では，整数変数の定義域（4 bit データであるので 0〜15）を前提として証明を行う．たとえば，以下のような式を判定する．

∀ 全変数
(((0 ≤ データ入力 A ≤ 15) ∧ (0 ≤ データ入力 B ≤ 15))
　imply 証明すべき式)　　(2)

入力および出力として論理型，整数型のいずれを用いるかによって 4 通りの式が得られる．これらの式の真偽はいずれも同じであり，実際の検証では，判定系はこれらの式のいずれか 1 つの真偽を判定すればよい．これら 4 つの結果について述べる．

比較として用いた判定系は，文献 4) で提案されている方法である（従来手法と呼ぶ）．この判定系は，本手法と同じ Cooper のアルゴリズムを採用している．また，すべての変数が冠頭の全称限定子で束縛されているプレスブルガー文のみを入力とすることができる．そのことを利用して，全称限定子を適用する順番（変数を消去する順番）をヒューリスティックに決定するなどの方法により高速な式判定を実現している．しかしデータの内部表現は，本手法のように内部頂点の共有は行っていない．

判定結果を**表 1** に示す．従来手法，提案手法および提案手法より整数部分の共有機能を取り除いた判定系の計算時間，および使用したノード数について使用されたノード数を示す．

およそ，従来手法に比較して提案手法では，5 倍程度の時間を要している．この検証では，入力を整数型とする式の真偽判定が計算時間が短く使用するノード数も少ない．また，判定すべき式中に整数変数が少ないので，整数部分の共有機能を取り除いた判定系は，ノード数は若干多い程度の消費であり，計算時間は提案手法とほぼ同等である．

参考として，impl1 と imp2 どうしの等価性を判定した結果も示す☆．

次に，さらに大規模な組合せ回路 MCNC ベンチマークセットのなかの HLSynth95 の浮動小数点加減算器 fp-add の検証実験について述べる．浮動小数点は，1 ビットの符号，23 ビットの仮数および 16 ビットの指数の組（計 40 ビット）で表現される．要求仕様および実現仕様では，仮数および指数を整数変数で表現する．

要求仕様は，各出力ごとに，変数として入力変数のみからなるプレスブルガー関数を用いて記述される．実現仕様は，fp-add の VHDL 記述を元に，中間ネットも整数変数あるいは論理変数として扱い，各出力，各入力および各中間ネットの間で成り立つべき関係をプレスブルガー関数を用いて記述される．したがって，この例題では，要求仕様と実現仕様とでは，入出

☆ 文 ∀ 全変数 $(F(\ldots) = G(\ldots))$ の真偽判定問題を考える．$F$ および $G$ を表現するデータ構造に一意性がある場合は，全称限定子に対する処理を行う必要はなく，$F$ および $G$ のデータ構造を構築し，同じかどうかで真偽を判定できる．提案するデータ構造は表現の一意性を有しないので，さらに，全変数に対する全称限定子に対する処理を行って得られた結果（真か偽のみが得られる）により真偽を判定している．

表 1 例題に対する実行結果
Table 1 Results of verifications.

| | 論理変数 | 整数変数 | 従来時間 | 提案判定系 | | 提案–整数共有 | |
|---|---|---|---|---|---|---|---|
| | | | (秒) | 時間 (秒) | ノード | 時間 (秒) | ノード |
| 74382 入力 出力 | | | | | | | |
| 論理 論理 | 56 | 4 | 0.3 | 6.8 | 338581 | 6.8 | 338809 |
| 整数 論理 | 56 | 4 | 0.3 | 1.6 | 13395 | 1.6 | 13734 |
| 論理 整数 | 56 | 4 | 0.4 | 1.6 | 16295 | 1.7 | 24801 |
| 整数 整数 | 56 | 4 | 0.3 | 1.6 | 12897 | 1.6 | 13186 |
| impl と imp2 | 81 | 0 | 0.3 | 1.8 | 14682 | 1.8 | 14682 |
| HLSynth95 fp-add の出力 nan_res | 45 | 39 | 256.1 | 211.6 | 98958 | — | 524359 |

力のデータタイプが同じである．ここでは，実現仕様によって表される整数データの処理構造が，要求仕様の指定する入出力関係を満たすかどうかという機能検証を行っている．

本例題では，出力のうち，異常入力が与えられたときに立つ NaN フラグについてのみ，要求仕様における出力と実現仕様における出力が等価になるかを調べた．この結果を表 1 の最下段に示す．

要求仕様では，出力 nan_res は，入力の仮数および指数 op1_exp，op1_mant などのありえない値の組合せが入力されたときに真となるように，以下のプレスブルガー関数で指定されている．

```
nan_res =
(((($op1_exp = 255) and
  not ($op1_mant = 0))
 or (($op2_exp = 255) and
  not ($op2_mant = 0)))
```

実現仕様では，主プロセスは 8 段のステップで記述されており，その第 2 ステップで nan_res の値が指定されている部分の記述は以下のようになる．

```
  :
and (if ($op1_exp = 255) then
      (($exp_result = $exp)
       and (special_flag2_1 = true)
       and (if ($op1_mant = 0) then
              :
            else (
              (nan_res = true)
            )
      )
    else (
      :
```

この例では，提案手法から整数部分の共有の機能を削除したものでは，メモリが不足し（途中終了時のノード数は，提案手法で消費したノード数に比べ 5 倍以上のノードを消費している），判定することができなかった．この例では，入力出力ともに同じデータタイプであるので，各変数のデータ変換の必要はない．したがって，限定子の処理においては，消去される整数変数には，要求仕様や実現仕様等の記述に現れる等式および不等式で与えられる整数型の式が代入される．この例では，整数変数が多く，整数部分の共有化の効果が現れていると考えられる．

NaN フラグ出力は，依存する入力変数が 4 個と少ない例である．けれども，従来手法が出力変数自身あるいは，それらにデータ依存関係のある端点に対応する変数から消去するヒューリスティックを採用しており，この検証入力に対しても効率的な変数消去順を決定して判定を行っているのに対して，本手法では証明すべき式において，構文解析において出現する順に変数を消去するという単純な方法を採用しているにもかかわらず，20%程度短時間で判定を行える．

本例題は，1 つの整数変数が最大 23 ビットに対応する場合もあり，検証例題としてはかなり大きな例にあたると考えられるが，本実験により，実際に検証可能であることが示された．その他の出力においては，従来手法でも 10000 秒以上要したので実行を打ち切り，本手法でも全称限定子を施す前までのデータ構造の作成は行えたが，全称限定子に対する処理を実行する途中においてメモリ不足による実行中止となりデータを収集できなかった．このような浮動小数点演算の設計検証など大規模な回路の検証については，今後の課題である．

## 5. あとがき

本論文では，整数上の制約論理の 1 つであるプレスブルガー文の真偽判定のためのデータ構造の提案を行い，そのライブラリを作成した．そして本ライブラリを用いてプレスブルガー文の真偽判定手続きを実装し，回路検証に適用して本手法の評価を行い．そこで従来の判定系との比較を行った．

従来の判定系ではいくつかの高速化のための手法を組み合わせて適用しているのに対して，本プログラムでは，現段階では Cooper のアルゴリズムを実現しただけであるにもかかわらず，大規模な検証例題においては，本手法が短い実行時間で検証を行うことができる場合もあった．

本論文では，提案する方法の評価として組合せ回路の例を用い，全称限定子が冠頭でのみ出現するプレスブルガー文に対する従来の判定系との比較を行った．提案する手法は，この限定されたクラスより広いプレスブルガー文の判定を実行することができる．今後，回路検証においてこのクラスに帰着できる興味ある回路検証問題や，我々が従来提案してきた順序回路の設計検証[1),2)] に取り組みたいと考えている．

今後さらなる高速化および省メモリ化のために，従来手法あるいは BDD において提案されている改善手法のうち提案するデータ構造においても有効であると考えられるものを適用したり，また，提案するデータ構造特有の改善手法を考案し，本ライブラリの改良を行う予定である．

## 参 考 文 献

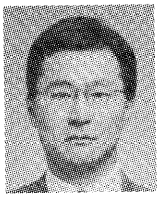

1) Kitamichi, J., Morioka, S., Higasino, T. and Taniguchi, K.: Automatic Correctness Proof of Implementation of Synchronous Sequential Circuits Using Algebraic Approach, *Proc. 1994 Conference on Theorem Provers in Circuit Design* (*TPCD94*), LNCS, Vol.901, pp.165–184 (1995).
2) 竹中 崇，北道淳司，西川清史：複数の制御部を持つ同期式順序回路の一設計検証法，情報処理学会論文誌，Vol.39, No.7, pp.2308–2322 (1998).
3) 東野輝夫，北道淳司，谷口健一：整数上の線形制約の処理と応用，コンピュータソフトウェア，Vol.9, No.6, pp.31–39 (1992).
4) 森岡澄夫，柴田直樹，東野輝夫，谷口健一：プレスブルガー文真偽判定手続きを用いた算術演 算回路の正しさの証明，信学技報，VLD96-61, pp.49–56 (1996).
5) Minato, S.: *Binary Decision Diagrams and Applications for VLSI CAD*, Kluwer Academic Publishers (1996).
6) 湊 真一：計算機上での BDD の処理技法，情報処理，Vol.34, No.5, pp.593–599 (1993).
7) Cooper, D.: Theorem Proving in Arithmetic without Multiplication, *Machine Intelligence*, No.7 (1972).
8) Bryant, R. and Chen, Y.-A.: Verification of Arithmetic Functions with Binary Moment Diagrams, Technical Report, School of Computer Science, Carnegie Mellon University (1994).
9) Bryant, R. and Chen, Y.-A.: Verification of Arithmetic Circuits with Binary Moment Diagrams, *32nd DAC*, ACM, pp.535–541 (1995).
10) Pugh, W.: A Practical Algorithm for Exact Array Dependence Analysis, *Comm. ACM*, Vol.35, No.8 (1992).
11) Kelly, W., Maslov, V., Pugh, W., Rosser, E., Shpeisman, T. and Wonacott, D.: The Omega Project: Frameworks and Algorithms for the Analysis and Transformation of Scientific Programs, http://www.cs.umd.edu/projects/omega/ (1999 年 3 月 15 日現在).
12) Wolper, P. and Boigelot, B.: An Automata-Theoretic Approach to Presburger Arithmetic Constraints (Extended Abstract), *2nd Int. Symp., SAS'95*, LNCS, Vol.983 (1995).
13) Shiple, T., Kukula, J. and Ranjan, R.: A Comparison of p Presburger Engines for EFSM Reachability, *10th Int. Conf. Computer Aided Verification98*, LNCS, Vol.1427 (1998).
14) Fischer, M. and Rabin, M.: Super-exponential Complexity of Presburger Arithmetic, *Proc. Symp. on the Complexity of Real Computation Processes* (1973).
15) Amon, T., Borriello, G., Hu, T. and Liu, J.: Symbolic Timing Verification of Timing Diagrams using Presburger Formulas, *34th DAC*, ACM (1997).
16) Shostak, R.: On the SUP_INF Method for Proving Presburger Formulas, *J. ACM*, Vol.24, No.4, pp.529–543 (1977).
17) Srivas, M., Rueb, H. and Cyrluk, D.: Hardware Verification Using PVS, *Formal Hardware Verification – Methods and Systems in Comparison*, LNCS, Vol.1287 (1997).
18) Naoi, K. and Takahashi, N.: An $n^3u$ Upper Bound on the complexity for Deciding the Truth of a Presburger Sentence Involving Two Variables Bounded Only by Existential Quantifiers, *IEICE Trans. Inf. Syst.*, Vol.E80-D, No.2 (1997).
19) McMillan, K.: *Symbolic Model Checking*, Kluwer Academic Publishers (1993).
20) Bultan, T., Gerber, R. and Pugh, W.: Symbolic Model Checking of Infinite State Systems Using Presburger Arithmetic, *Int. Conf. on Computer Aided Verification 97* (1997).
21) Hachtel, G. and Somenzi, F.: *Logic Synthesis and Verification Algorithms*, Kluwer Academic Publishers (1996).
22) Chen, Y.-A.: BXD Package Home Page, http://www.cs.cmu.edu/afs/cs.cmu.edu/usr/yachen/www/bxd.html (1999 年 3 月 15 日現在).

23) 景山洋行，北道淳司，船曳信生：整数上のある制約論理式の処理のための BDD の拡張とそれを用いた回路検証，DA シンポジウム'98 論文集，Vol.98, No.9 (1998).
24) 小林英史，竹中　崇，北道淳司，船曳信生，谷口健一：out-of-order 型パイプライン CPU の機能検証，DA シンポジウム'98 論文集，Vol.98, No.9 (1998).
25) Colón, M.A. and Uribe, T.E.: Generating Finite-State Abstractions of Reactive Systems Using Decision Procedures, *International Conference on Computer-Aided Verification, CAV'98*, Vol.1427 (1998).

(平成 10 年 9 月 18 日受付)
(平成 11 年 2 月 8 日採録)

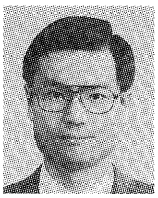
**景山　洋行**（学生会員）

1974 年生．1998 年大阪大学基礎工学部情報工学科卒業．現在同大学大学院博士前期課程在学中．論理回路の形式的検証，BDD 等に興味を持つ．



**北道　淳司**（正会員）

1965 年生．1988 年大阪大学基礎工学部情報工学科卒業．1991 年同大学大学院博士後期課程中退．同年同大学情報工学科助手．1997 年同大学大学院基礎工学研究科助手．工学博士．動的再構成可能 FPGA，ハードウェアの形式的仕様記述，設計，検証等に関する研究に従事．電子情報通信学会会員．



**船曳　信生**（正会員）

1984 年東京大学工学部計数工学科卒業．同年住友金属工業（株）入社．1991 年米ケースウエスタンリザーブ大学大学院修士課程修了．1994 年大阪大学基礎工学部情報工学科講師．現在，同基礎工学研究科助教授．工学博士．ニューラルネットワーク，組合せ最適化アルゴリズム，VLSI 設計，信号処理等に関する研究に従事．IEEE，電子情報通信学会各会員．