設 計 自 動 化 60－18
（1991．12．13）

enumerationによる二分決定グラフ構成法

伊藤雅樹† 西田隆夫†† 清水嗣雄†
†(株)日立製作所 ††(株)日立製作所
中央研究所 神奈川工場

組合せ回路の論理検証を二分決定グラフ(BDD)を用いて行うためには、設計された回路の結線記述から、回路の実現する論理関数のBDD表現を構成する必要がある。従来、記号シミュレーションが用いられているが、記号シミュレーションは内部信号線の論理関数をも必要とするため、多大な記憶量を要するという問題点がある。本報告では、記号シミュレーションに代わり、enumerationを用いたBDDの構成法を提案する。enumerationは、外部出力値を0/1に決定する外部入力値の組合せを二分探索によって全て並べ上げる手法である。机上評価では、必要記憶量は記号シミュレーションに比べ、約1/8に削減される。

Binary Decision Diagram Construction Using Enumeration

Masaki Ito† Takao Nishida†† Tsuguo Shimizu†
†Central Research Laboratory ††Kanagawa Works
Hitachi, Ltd.

It is an inevitable task to construct a BDD(Binary Decision Diagram) from a designed combinational circuit, in order to verify the design using BDD's. Although symbolic simulation is often used, it requires high memory overhead because the functions of internal lines are required. In this paper, we propose a method for BDD construction using enumeration. Enumeration utilizes binary search technique to find all of the input patterns that yield fixed values (0/1) at primary outputs. Desk-top performance evaluation demonstrates that our method requires only 1/8 of the storage needed by symbolic simulation.

1. はじめに

形式的論理検証手法は、仕様と設計された論理回路をそれぞれ形式的に記述し、両者の等価性を数学的に証明するものである。これまでに様々な手法が提案されており[1]、対象を組合せ回路に限定すれば、実用化も不可能ではなくなってきた。特に論理関数の表現手法として二分決定グラフ(BDD: Binary Decision Diagram)を用いると、効率よく論理照合が実施できることが知られており、BDDによる論理照合手法の研究やBDD自身の性質に関する研究も盛んである[2][3][4]。

BDDを用いて論理照合を行うためには、設計された論理回路の結線情報から回路が実現する論理関数を抽出し、BDDで表現することが必要となる。従来、記号シミュレーションが用いられているが[2][5][6]、記号シミュレーションは、回路の外部出力信号線の論理関数を求めるために、内部信号線の論理関数まで必要とするため、膨大な記憶量を要するという問題点があった。

そこで、我々は少ない記憶量でBDDを構成する手法BOXER (BDD represented lOgic function eXtraction based on EnumeRation)を考案した。BOXERは、外部出力値を決定するような外部入力値の組合せを二分探索する手続きであるenumerationを基本とする。本稿ではBOXERのアルゴリズムを述べ、必要記憶量に関する机上評価結果を示す。

以下、第2章でBDDを紹介し、従来の問題点を明らかにする。第3章ではBOXERのアルゴリズムを詳説し、第4章でBOXERの性能向上手法を説明する。最後に第5章でBOXERの性能の机上評価結果を示す。

2. BDDと記号シミュレーション

2.1 二分決定グラフ(BDD)

BDDは次の特徴を持つ論理関数のグラフ表現の一種である。

(1) 多くの実用的論理関数を比較的コンパクトに表現できる。

(2) 論理関数の入力変数の順序を定め、グラフを既約形にすることにより、関数の表現が一意に定まる。

以下、BDDについて簡単に説明する。BDDはループを持たない有向グラフであり、終端ノードと非終端ノードの二種類のノードを持つ。非終端ノードは論理関数の入力変数に対応付けられる。全ての非終端ノードは二本のエッジを持ち、それぞれ0エッジ、1エッジと呼ばれる。エッジの値がエッジの始点のノードに対応する入力変数の値を表す。終端ノードは論理関数の値に対応する。値に応じて0ノード、1ノードという。

全ての入力変数に全順序を与え、BDDのルートノードから終端ノードへの全経路上に現われる入力変数の順序が与えられた順序に従うとき、このBDDを順序付き二分決定グラフ(OBDD: Ordered Binary Decision Diagram)と呼ぶ。本稿では、BDDというときには全てOBDDを指すこととする。

BDDは、次の二条件を満たすとき既約形であるという。

(条件1) 0エッジと1エッジが同じノードを指すような非終端ノードが存在しない。

(条件2) 同じ形の部分グラフは高々一つしか存在しない。

BDDを既約形に変換する操作を既約化と呼ぶ。特に、(条件1)を満たすようにBDDを変換することを「冗長ノードの削除」、(条件2)を満たすように変換することを「部分グラフの共有」と呼ぶ。既約化のアルゴリズムは[2]で示されている。

2.2 論理照合方法

BDDを既約化すると、論理関数の表現が一意に定まる。すなわち、同じ関数を表す二つのBDDが、同じ入力変数の順序を用いており、かつ既約形であれば、そのグラフは常に同じ形になる[2]。従って、二つの論理関数の照合は二つのグラフの構造一致判定に帰着され、高速に実施できる。

2.3 BDDの大きさ

BDDは変数の順序によって、そのノード数が大きく変化する。最小のBDDを与える変数の順序づけアルゴリズムも提案されているが[4][7]、実用的な計算量とはいえない。そこで、多くの研究は良い順序づけを得る発見的手法の開発に対して行われている。これまでに、次の性質が知られている[8]。

(性質1) 制御系の入力変数はBDDの上位に置き、データ系の入力変数はBDDの下位に置くとBDDが小さくなる。

(性質2) 局所計算性のある入力変数の組は極力近い順序にすることによりBDDが小さくなる。

2.4 記号シミュレーション

論理照合をBDDを用いて行うためには、比較する論理関数をそれぞれ同じ変数の順序を用いてBDDで表現する必要がある。従来、回路の結線情報からBDDを構成する方法としては記号シミュレーションが用いられている[2][5]。これは、入力値として0/1の信号値以外に記号をも許す論理シミュレーションである。記号シミュレーションでは各外部入力に異なる記号を割り当て、内部信号線の信号値を割り当てた記号の論理関数として求めて行く。外部出力における信号値が求まったなら、それが回路の実現する論理関数に他ならない。

記号シミュレーションでは目的とする外部出力端子の論理関数を求めるために、回路内部の各信号線の論理関数をも計算しなければならない。従って、一般に多大な記憶量を必要とするため、大規模な回路に適用できないという問題点があった。そこで、少ない記憶量でBDDを構成する手法であるBOXERを考案した。次章ではBOXERのアルゴリズムを解説する。

3. BDD構成方法BOXER

3.1 BOXERの概要

BOXERはenumeration(並べ上げ)という操作に基づいてBDDを構成する。enumerationとは外部出力の値を0/1に決定するような外部入力値の組合せを全て探索する手続きである。一般的なenumerationの手続きの概略は次のようになる。ただし、次の手続きは、簡単のため単一出力回路を仮定している。

Step 1. 全外部入力に不定値Xを割り当てる。

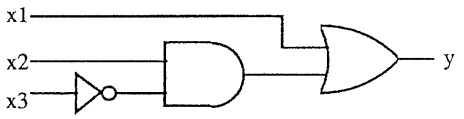
Step 2. 不定値Xの割り当てられている外部入力を一つ選び、0または1の論理値を割り当てる。

Step 3. Xを含む3値の論理シミュレーションを行い、外部出力値を求める。

Step 4. 出力が0または1であれば、Step 5へ。出力がXのときは、Step 2へ。

Step 5. 全ての入力の組合せを尽くせば終了。尽くしていなければ、論理値の割り当てをまだ割り当てていない組み合わせに変更し、Step 3へ戻る。

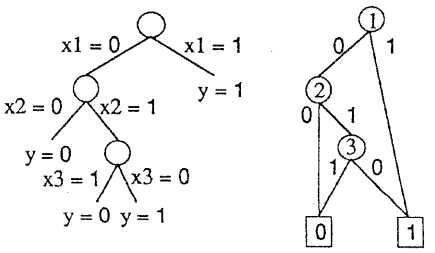
Step 5の割り当て変更は、通常、最も後に割り当てた変数の値から変えて行く。enumerationの計算例を図3.1に示す。図3.1の回路のように3入力であれば、$2^3=8$通りの入力値の組合せがあるが、不定値Xが論理値0も1も同時に表していると考えることにより、図に示すように、6通りの組合せで全ての組合せを表現できる。



x1	1→0	0	0	0	0	
x2	X	X→1	1	1	1→0	
x3	X	X	X→1	→0	→X	
y	1	X	X	0	1	0

図3.1 enumerationの計算過程

この計算の過程をグラフで表現すると、図3.2(a)に示す二分探索木を深さ優先でたどる過程として表現できる。一方、図3.1の回路が実現する論理関数y=x1∨x2∧¬x3をBDDで表現すると図3.2(b)となる。これらはそっくりであることがわかる。すなわち、上述のenumerationの計算過程を二分探索木として保存することが、そのままBDDを構成することになるのである。以上の処理がBOXERの骨子をなす。一般には本処理によって



(a) 二分探索木　(b) BDD

図3.2 二分探索木とBDD

生成されるBDDは既約形ではないため、生成されたBDDに対して既約化を行う必要がある。

3.2 BOXERの詳細アルゴリズム

単一出力の組合せ回路を対象とするときは、生成されるBDDは一つであるから、enumerationの計算過程、すなわち二分探索木の探索過程をそのまま保存(し、既約化)すれば求めるBDDとなる。多出力回路の場合は、一般に外部出力端子ごとに実現される論理関数が異なるので、生成されるBDDも異なる。従って、これらを一元的に管理することができない。そこで、enumerationの計算過程を二分探索木としてBDDとは別の管理を行う。二分探索木は外部出力の値を決定する入力値の組合せの探索状態を保持する。基本的には、全ての外部出力に対して、二分探索木と同じ形のBDDを生成する。ただし、外部入力値を割り当てていく過程で、出力値が0/1に定まった外部出力については、その時点で0ノード、1ノードへのエッジを生成し、それ以下のノード生成は行わない。これらのノードは生成したとしても、既約化において抹消されるものである。これにより、無駄なノード生成を削減することができる。

BOXERのアルゴリズムを図3.3に示す。二分探索木はBDTと表記している。本アルゴリズムで用いられる変数は次の5種類である。

- cur_node…現在処理中の二分探索木上のノードを指す。
- next_var…今回のループで、新たに値を割り当てる入力変数を表す。next_varがNULLであるときは、今回のループで、論理値を新たに割り当てるのではなく、現在の割当てを変更することを示す。
- index…各ノードが一つずつ持つ変数であり、ノードが対応付けられている入力変数を保持する。
- complete…BDDが完成したときに1となる変数。
- assign…外部入力値の割当てが決定したときに1になる変数。

二分探索木をそのまま生成すると、外部入力数に対して指数オーダのノード数が必要となるが、実際には、外部入力数を最大長とするLIFOスタックで木探索過程を管理すればよい。従って、二分探索木が記憶量を圧迫することはない。

4. BOXERの処理量削減

4.1 BOXERの処理効率とBDDの大きさ

BOXERの中で最も処理時間を要するのは論理シミュレーションである。BOXERにおいては論理シミュレーションは二分探索木の枝の数だけ実施される。従って、二分探索の空間を削減する、即ち、枝刈りを効率良く行うことができれば、論理シミュレーションの回数を削減することができる。枝刈りを行うということは二分探索木が小さくなるということであり、すなわち、BDDを小さくすることにつながる。逆に言えば、小さなBDDを生成することを考えれば、必然的にBOXERの処理効率も向上すると考えられる。従って、以下では小さなBDDを生成するための手法について説明する。

第2章で述べたように、BDDの大きさは変数の順序の影響を大きく受ける。BOXERにおいては、3.1節で述べたenumerationのStep 2で選ばれる変数の順序がこれに対応する。BOXERでは

- 可観測性尺度の利用
- ゲート入力探索

の二つによりBDDのノード数を削減する。以下の節でそれぞれを説明する。

4.2 BDDノード数削減手法

4.2.1 可観測性尺度

第2章で述べた(性質1)は、出力へ与える影響の大きい入力変数を上位に置くとBDDが小さくなることを示唆している。そこで、BOXERでは可観測性尺度(Observability measure)を用いることにより、出力へ与える影響の大きさを判断する。可観測性尺度とは論理回路の故障検査/診断に用いられる概念であり、回路内の全ての信号線に対して与えられる尺度である。直感的には、その信号線の信号値が変化することにより外部出力の値まで変化するかを表す程度をいう。通常、その信号線の値の制御のしやすさを表す可制御性尺度(Controllability measure)とペアで計算される。

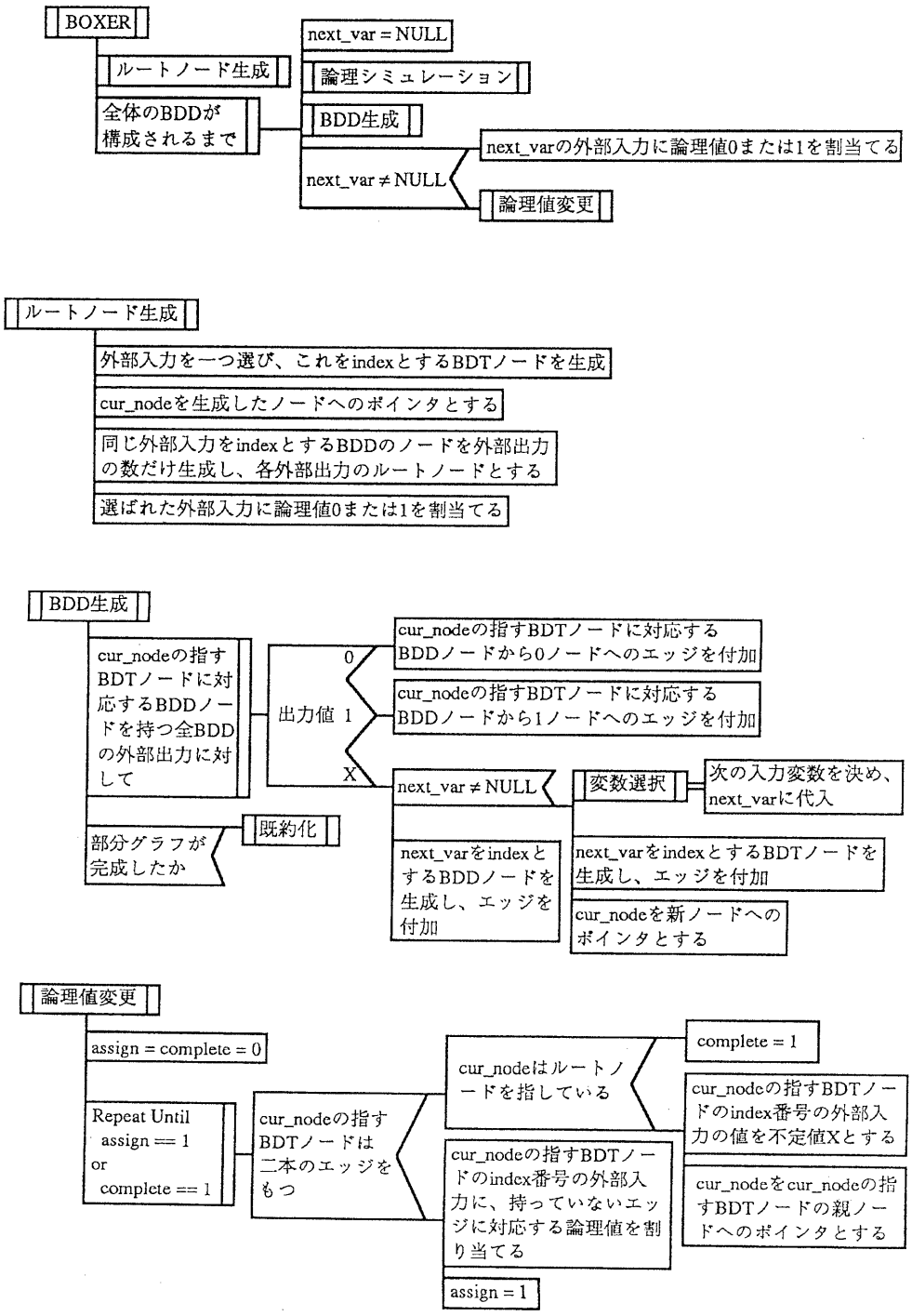


図3.3　BOXERのアルゴリズム

可制御性尺度/可観測性尺度はその目的に応じて様々な計算法が提案されている。BOXERでは、[9]の計算法を採用した。[9]は、再収れんを考慮に入れており、より正確な可制御性尺度/可観測性尺度が計算される。例えば、1可制御性が0であるような信号線は、その信号線の値を1に制御したいとしたとき、矛盾なく外部入力に値を割り当てることができることが証明されている[9]。

4.2.2 ゲート入力探索

本節では局所計算性をもつ入力変数を探索する手法について考える。BOXERでは、局所計算性をもつ入力変数の探索を、enumerationの過程で実行される論理シミュレーションで生成されるデータを流用して行う。

論理シミュレーションはレベルソート・イベント駆動方式を採用する。これは、enumerationでは、二分探索木を終端ノード方向へ降りる際、1入力しか変化せず、イベント数が極めて少ないことを理由とする。

論理シミュレーション中で、次に示す手法により局所計算性をもつ入力変数の探索を効率的に行うことができる。すなわち、イベント・キューから取り出した論理素子から回路内を外部入力方向へバックトレースするのである。バックトレースはテスト生成アルゴリズムFAN[10]において用いられている多重後方追跡と同様に行う。ただし、後方追跡に際して、可観測性尺度のよいものから優先して追跡を行う。

複数のランクにイベントが生じている場合には小さなランク(外部入力に近い方)から行う。バックトレースすることによって外部入力端子へ到達したとき、この外部入力を次の入力変数として採用する。バックトレースの過程で、すでにバックトレースした論理素子に至ったときにはバックトレースを終了することにより、同じ外部入力へ二度以上到達するような無駄な処理を排除することができる。

4.2.3 例

図4.1の回路を例に用いて変数の順序づけの方法を具体的に示す。回路内の信号線の可制御性尺度(C0、C1)、可観測性尺度(Ob)は[9]の計算法を用いると表4.1に示す値となる。表から最も可観測性尺度の良い外部入力端子はb、即ちx2であることがわかる。そこで、まずx2に0を割り当てる。するとゲートB及びCがイベントを受け取る。Cは他の入力を持たないのでバックトレースの対象から外す。Bからb1以外の入力信号線をたどるとa即ちx1へ到達する。従ってx2の次はx1とする。

BやCが受け取ったイベントを評価するとgは0に、fは1に変化し、イベントを生じる。従ってD及びEがイベントキューに登録される。バックトレースはレベルの小さいほうから行うので、Dからバックトレースをするが、Cはトレース済みであるからへ到達する。Aの二つの入力は可観測性尺度が等しいのでいずれを上位に用いても良く、例えばx3を上位におくと、x2, x1, x3, x4の順序が得られる。Eがイベントキューに残っているが、全ての変数の順序が定まったので、順序づけの手続きは終了する。

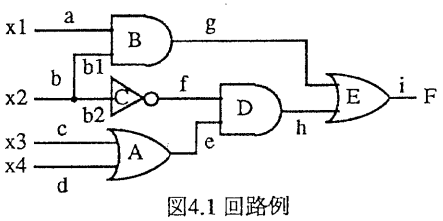


図4.1 回路例

表4.1 可制御性尺度、可観測性尺度

信号線	C0	C1	Ob
a	0	0	1
b	0	0	0
b1	1	1	0
b2	1	1	0
c	0	0	1
d	0	0	1
e	0	0	1
f	1	1	0
g	0	1	0
h	0	1	0
i	0	1	0

4.3 既約化のタイミング

BDDが比較的コンパクトに論理関数を表現できるのは、「冗長ノードの削除」、「部分グラフの共有」を行うからである。enumerationとは、終端ノードへのエッジを持つ「冗長ノードの削除」をあらかじめ行っておくことであると見ることができる。ところが、「部分グラフの共有」はenumeration中に行うことができない。これは、enumerationによってノードを生成した時点では、そのノードは枝を持たず、部分グラフの形状が判明していないことに起因する。

従って、enumerationによって生成されるBDDは、完全な二分木の構造をなす。この事実は、既約化すると入力数に比例するノード数となるBDDでも、enumeration終了時点には指数に比例するノード数を持つ危険性があることを意味する。

そこで、BOXERでは、部分グラフが完成次第、部分グラフ単位に、逐次既約化を実施する。部分グラフの完成とは、あるノードから終端ノードへ至る全てのノードが2本のエッジを持つことをいう。これにより、最終的に入力数に比例するノード数となるBDDであれば、BOXERの処理中にノード数が指数爆発することは回避できる。

5. 必要記憶量机上評価

BOXERに必要となるメモリは二つに大別される。

・BDDを生成する過程で必要となるメモリ

・生成されるBDDを保持するためのメモリ

の二つである。記号シミュレーションと比較すると、同じ変数の順序を用いたなら、同じBDDを生成するため、後者はいずれも同じメモリ量を要する。ところが、前者については大きな違いが生じる。上述のように、記号シミュレーションでは回路内部の信号線の論理関数を記憶する必要があるが、BOXERでは論理シミュレーションに必要なメモリ量、すなわち各信号線に対して2ビット(0, 1, Xの3値を表現)でよい。

以上の考察に基づき、4ビットの全加算器を対象に必要メモリ量の机上評価を行った。ゲート数は151ゲートである。結果を図5.1に図示する。各項目の意味を以下に示す。

計算過程…内部信号値のためのメモリ量。

結果のBDD…結果として出力されたBDDのメモリ量。

記号シミュレーション…信号値の内部表現にもBDDを用いた記号シミュレーション。

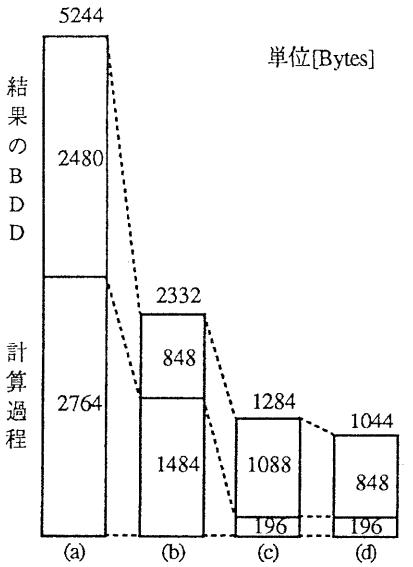
順序考慮せず…入力変数の順序には特別の注意を払っていないものの一例。

BOXERの順序…可観測性尺度、ゲート入力探索の両者を行ったときのBOXERで得られる順序を用いた場合。

BOXER…本稿で述べたenumerationに基づくBDD構成手法。

可観測性尺度のみ…可観測性尺度の良いものから順に変数の順序をつけたもの。

ゲート入力探索も…ゲート入力探索も行って順序づけを行ったもの。



(a) 記号シミュレーション(順序考慮せず)
(b) 記号シミュレーション(BOXERの順序)
(c) BOXER(可観測性尺度のみ)
(d) BOXER(ゲート入力探索も)

図5.1 BOXERの必要メモリ量

まず、BOXERと記号シミュレーションのメモリ効率の比較を行う。同じ変数の順序を用いたときの両者のメモリ量は(b)と(d)である。この評価

に用いた順序はBOXERで得られたもので、全加算器を表わすBDDのノード数を最小にする順序であることが知られている。同じ変数の順序であるから、結果として生成されるBDDの大きさは同じである。上述の様に、差は計算過程に現われる。計算過程で必要とした記憶量はそれぞれ1484バイト、196バイトであった。全加算器は最適な入力変数の順序を用いると、外部入力数にノード数が比例するという意味で、BDDを用いた記号シミュレーションに有利な回路である。にも係わらず、計算過程のメモリ量に7.6 (1484/196)倍の差が生じたという事実から、より複雑な回路に対する必要メモリ量の差はさらに拡大されると予想される。

次に変数の順序づけの影響を考える。まず、(a)と(b)による比較から入力変数の順序がBDDのノード数に与える影響の大きさが読み取れる。上述のように、全加算器はBDD表現に適しているにもかかわらず、結果として生成されるBDDの大きさが約3倍(2480/848)も異なってくるのである。この結果は、BOXERで得られる順序の有効性を示すものでもある。最後に変数の順序づけの手法について考察する。(a)と(c)の結果のBDDの比較から、可観測性尺度だけでも大きな効果があることが読み取れる。しかし、(c)と(d)から、さらにゲート入力探索を行うことがより大きな効果をもたらすことがわかる。

6. おわりに

本稿では、組合せ回路の結線情報からその実現する論理関数のBDD表現を構成する手法BOXERのアルゴリズムを述べた。BOXERは、記号シミュレーションと異なり、enumerationに基づいてBDDを構成する。机上評価の結果、記憶量は記号シミュレーションの約1/8しか必要としないことがわかった。

計算時間に関しては、BOXERは記号シミュレーションに対して、優位とは言えないと考えられる。ただし、enumerationは並列処理による高速化の効果が大きいことが示されており[11]、BOXERの並列化が今後の課題となる。

7. 参考文献

[1] L. J. M. Claesen: Formal VLSI Correctness Verification - VLSI Design Methods-II; North-Holland, (1990)

[2] R. E. Bryant: Graph-Based Algorithms for Boolean Function Manipulation; IEEE Trans. Comp., Vol. C-35, No. 8, pp.677-691 (Aug. 1986)

[3] N. Ishiura, et al.: A Class of Logic Functions Expressible by Polynomial-Size Binary Decision Diagrams; SASIMI-90, pp.48-54 (Nov. 1990)

[4] S. J. Friedman, et al.: Finding the Optimal Variable Ordering for Binary Decision Diagrams; IEEE Trans. Comp., Vol.39, No. 5, pp.710-713 (May 1990)

[5] S. Minato, et al.: Fast Tautology Checking Using Shared Binary Decision Diagrams - Benchmark Results -; in [1], pp.107-111(1990)

[6] A. L. Fisher, et al.: Performance of COSMOS on The IFIP Workshop Benchmarks; in [1], pp.101-105(1990)

[7] 澤田、他: 論理関数を表現する二分決定グラフの最小化; 電子情報通信学会コンピュテーション研究会, COMP91-15, pp.27-36(May 1991)

[8] 湊: 共有二分決定グラフの「幅」に着目した変数の順序づけ手法; 情報処理学会第42回全国大会予稿集, pp. (6-)158-159 (Mar. 1991)

[9] M. Abramovici, et al.: SMART and FAST: Test Generation for VLSI Scan-Design Circuits; IEEE Design & Test of Computers, pp.43-54 (Aug. 1986)

[10] Hideo Fujiwara, et al.: On the Acceleration of Test Generation Algorithms; IEEE Trans. Comp., Vol.C-32, No.12, pp.1137-1144 (Dec. 1983)

[11] H-K. T. Ma, et al.: Logic Verification Algorithms and Their Parallel Implementation; IEEE Trans. CAD, Vol. 8, No. 2, pp.181-189(Feb. 1989)