設計自動化 63－3
(1992. 7.17)

# 論理検証システム CONDOR

向山 輝 *, 若林 一敏 *, 藤田 友之 *, 田中 英俊 *, 菅波 和幸 **
* 日本電気 (株), ** 北陸日本電気ソフトウエア (株)

大規模な回路を高速かつ完全に検証することを目的として開発した論理検証システム CONDOR について報告する。本システムは、仕様と回路を BDD によって表現し、両者の論理照合によって検証を行うシステムである。BDD による論理照合を大規模な回路に適用するためには、記憶領域と処理時間の両面で BDD のノード数をなるべく少なくする事が特に重要となる。本システムでは、BDD における入力変数順序の考慮、および中間信号による回路の分割照合によってノード数の削減を図った。実回路によってシステムの評価を行った結果、本システムの手法の有効性が実証された。

# Logic Verification System: CONDOR

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

This paper describes a formal logic verification system called Condor, which aims perfect and rapid verification of large scale circuits. Condor verifies circuits by using boolean comparison between gate level circuit and its specification. The boolean comparison is performed based on BDDs. In order to represent large scale circuits with BDDs, it is important to reduce the number of nodes in the BDDs. We propose a new input variable ordering technique and a new circuit partitioning technique for reducing the number of BDD nodes. Experimental results showed that the proposed techniques are quite efficient for verification of large scale circuits, and that Condor runs quite faster than conventional verification system based on logic simulation.

## 1 はじめに

近年の集積回路の大規模化に伴い、回路設計に必要とされる工数は膨大なものとなり、その効率化が急務となっている。論理回路の設計工程は、与えられた機能仕様の詳細化と検証という2つの工程に大きく分けることができる。機能仕様の詳細化は、自動合成技術の発達により効率化が進んできたため、論理検証の効率化が非常に重要となってきている。

従来、論理検証の方法として、仕様と設計された回路との比較を論理シミュレーションによって行う手法が用いられてきた。論理シミュレーションによる方法は、順序回路動作をも含めて、検証の対象となっている回路の等価性を判定することができる。しかし、すべての回路動作の振る舞いを求めるためには、膨大なシミュレーション用パターンが必要となるため、現実には非常に限られた部分の検証しかできない。したがって、回路設計の効率化を図るために、検証の高速化、完全化を実現することが課題となっていた。

組合わせ回路を対象とした論理検証は、与えられた機能仕様と、詳細化されたゲート回路の両者の論理表現を照合することによって行うことができる。論理表現の完全な比較を行うために、Boolean Comparison[1]の研究が行われてきたが、膨大なメモリ量と時間を必要とし、現実の大規模回路を検証することは不可能であった。

そこで、我々は大規模な論理回路を高速かつ完全に検証することを目的として、論理検証システム「CONDOR」を開発した。CONDORは、RTレベルの機能仕様とゲートレベルの回路を二分決定グラフ(BDD)[2][3]によって表現し、両者の論理的等価性を証明することによって論理検証を行うシステムである。

BDDは、グラフによる関数の論理表現法であり、

- 多くの論理関数を比較的コンパクトに表現することができる。
- グラフのパス上に現れる入力変数の順序を定めることにより、関数の論理表現が一意に決まる。

という特長を持つ。これらの特長のために、論理関数をBDDで表すことにより、論理的等価性の判定を高速かつ完全に行うことが可能となる。

しかし、大規模回路をBDDで表現した場合、記憶領域と処理時間の両面で限界に達するため、BDDのノード数をなるべく少なくすることが重要である。

本システムは、入力変数の順序最適化や、回路の分割照合によってBDDのノード数の削減を図り、現実の大規模論理回路の検証を可能にするものである。

CONDORはEWS4800上に実装され、現在、実設計回路によって評価中であるが、シミュレーションによる既存の検証システムと比較して、検証時間を大幅に短縮できることが実証された。

本稿では、まずBDDについて簡単に説明し、次にCONDORの検証方式の概要とBDDノード削減のために提案した変数順序決定法と回路の分割照合法について述べる。最後に、実設計回路によるシステムの評価結果を示す。

## 2 BDD(Binary Decision Diagram)

BDDは、論理関数の変数を表す変数ノードと、関数値を表す定数ノードから構成される根付き有向グラフである。変数ノードは、0エッジ、1エッジと呼ばれる2つの枝を持ち、それぞれ変数の値が0の場合と1の場合に対応する。BDDは、根ノードから定数ノードまでの各パスによって変数値の組み合わせとそれに対応する関数値を表し、関数の論理を表現する。

グラフの各パス上に現れる変数の順番が固定されている場合に、このBDDをOBDD(Ordered BDD)と呼ぶ。さらに、冗長なノードの削除と同形なサブグラフの共有を行ったものをROBDD(Reduced Ordered BDD)と呼ぶ。

例として、$A\cdot(B+C\cdot(D+E))$を$A,B,C,D,E$の変数順序で表すROBDDを図1に示す。

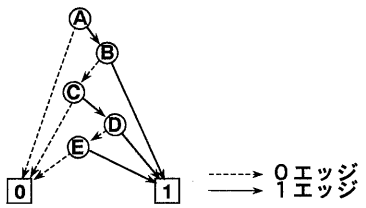



図1 $A\cdot(B+C\cdot(D+E))$を表すROBDD

論理関数をROBDDによって表現すると、関数の論理に対してグラフの形が一意に定まる。したがって、複数の関数の論理的等価性の判定を、グラフの形の比較によって行うことができるため、ROBDDは論理照合の手段として非常に有効である。

なお、本稿では、ROBDDを単にBDDと記述する。

## 3 CONDORの論理検証方式

CONDORのシステム概要を図1に示す。

本システムは、RTレベルの機能記述言語FDL(Function Description Language)[4]によって記述された仕様と、

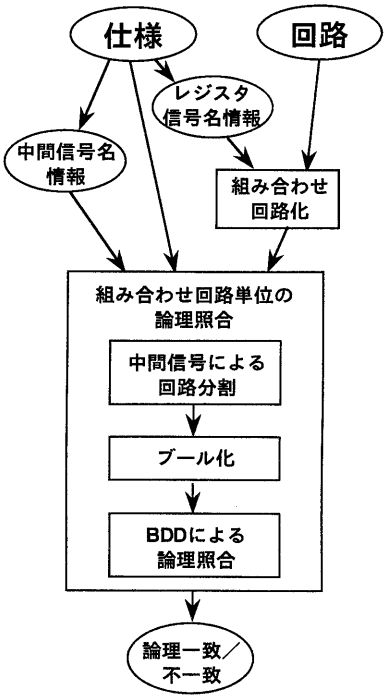



図2 システムの概要

ゲートレベルの回路記述を入力とし、両者の論理的一致判定の結果を出力するものである。

まず、仕様の記述からレジスタ信号名の情報を取りだし、その情報を利用して回路をレジスタからレジスタ間の組合せ回路に分割する。そして組合せ回路単位で、仕様と回路の論理照合をBDDを利用して行なう。

しかし、組合せ回路の規模が大きい場合は、BDDのノード数が記憶容量の限界に達し、BDDによる表現が不可能な場合がある。そこで、本システムでは、組合せ回路を一括してBDDに変換せず、部分回路に分割し、部分回路毎にBDDに変換することによって論理照合を可能としている。部分回路への分割は、仕様記述から抽出された中間信号名を利用して行なう。

部分回路に分割することによって、大規模な論理回路の検証が可能となるだけでなく、回路が論理的誤りを含む場合は、その誤りの存在する部分を中間信号単位に絞り込んで設計者に知らせることが可能となる。

次節以降では、BDDのノード数を少なくするための変数順序決定法と回路の分割照合法について詳しく述べる。

### 3.1 BDDの変数順序決定法

BDDは、入力変数の順序によってノード数が大きく変化することが知られている[3]。BDDを記憶するための領域は限られており、また、BDDによる論理照合は、ノード数にほぼ比例する時間がかかるため、ノード数をなるべく少なくするような入力変数の順序づけを行うことが重要である。最適な入力変数の順序を決定する手法が[5]に提案されているが、この手法は指数オーダの計算時間を必要とするため、入力数の大きい回路には適用できない。そこで、簡単なヒューリスティックによってなるべく良い変数順序を決定することを考える。

BDDのノード数を少なくするための入力変数の順序として、これまでに次のような性質が知られている[6]。

**(1) 制御性の高い変数を上位に順序づける。**

例えば、$[\log n]$本の制御入力と$n$本のデータ入力を持つデータセレクタの出力関数に対しては、データ入力を上位に順序づけるとノード数は$O(2^n)$になるのに対して、制御入力を上位にすると$O(n)$で済む。

**(2) 局所計算性のある入力同士は順序を近づける。**

AND-OR 2段回路では、ANDゲートを共有する入力を隣接するように順序づけると、多くのサブグラフを共有することができ、ノード数を少なくすることができる。

本システムでは、制御性の最も高い変数を最上位に順序付け、その変数とANDゲートを共有する変数を隣接させるという方針によってBDDの変数順序を決定している。以下に、変数順序の決定方法について説明する。

#### 3.1.1 重み付けによる変数の制御性評価法

制御性の高い変数を見つけるために、論理関数を2段で表現した時に現れる頻度（その変数を含む積項の数）を求める。そして現れる頻度の高い変数ほど、制御性が高いものとみなす。例えば、

$$\begin{aligned} F &= (a+b+c)\cdot d+\overline{c}\cdot e\cdot f \\ &= a\cdot d+b\cdot d+c\cdot d+\overline{c}\cdot e\cdot f \end{aligned}$$

に対して、$a$、$b$、$c$、$d$、$e$、$f$の制御性の高さはそれぞれ1、1、2、3、1、1と評価される。

しかし、実際には大規模な論理関数を2段に展開することは困難なため、変数の現れる頻度は、次に示すように論理関数を回路で表し、その回路上での重み付け法によって求める。

(1) [論理関数の回路表現]
論理関数を、AND、OR、INVERTERゲートからなる回路で表す。この時、INVERTERは、ファンイン側へ伝搬させ、すべてのINVERTERが、プライマリ入力のみに存在する様にする。

(2) [総積項数の算出]
各入力変数に重み1を与え、ANDゲートでは入力線の重みの算術積、ORゲートでは入力線の重みの算術和を取るようにして、入力から出力に向かって重みを伝達させる。すると、出力に現れる重みが、この関数を2段に展開したときの積項の総数となる。

(3) [特定変数を含む積項数の算出]
1つの入力変数だけ重みを0として(2)と同じように重みを伝達させる。出力に現れる重みは、重み0とした入力変数を含まない積項の数となるから、(2)で得られた重みからこの重みを差し引いた値が、この変数を含む積項数となる。

(4) 全ての入力変数について、(3)を繰り返す。

(1)の処理によって、回路には、$\overline{a+b}$のようなOR演算の否定をとるようなゲートがないため、各ORゲートにおける積項の数は、入力される積項数の和となり、各ANDゲートにおける積項の数は、入力される積項数の積となる。したがって、ここに示した手続きによって、各変数を含む積項数が算出できることは明らかである。

関数$F=(a+b+c)\cdot d+\bar{c}\cdot e$に対してこの方法を適用した場合の例を図3に示す。(a)は、入力変数にそれぞれ重み1を与え、各ゲートに重み付けを行った結果である。出力に現れる重み4が、$F$を和積表現したときの積項の総数を表す。(b)は入力変数$a$の重みだけ0にして重み付けを行った場合を示している。このとき出力に現れる重み3は$a$を含まない積項の数を表す。したがって、4から3を差し引いたものが$a$を含む積項の数となる。

### 3.1.2 制御性による変数順序決定法

前節に示した手続きによって評価された各変数の制御性に基づいて、入力変数の順序づけを行う。まず、制御性の最も高い変数（$a$とする）を最上位に順序づける。$a$を表すノードの0エッジは$F_{a=0}$のBDDを指し、1エッジは$F_{a=1}$のBDDを指す。$a$は制御性の高い変数であるから、$F_{a=0}$は大幅に簡約化されているはずである。そこで、$F_{a=1}$をなるべく簡約化するように次の変数を選ぶ。このとき、局所計算性のある入力同士を近付ける、という方針にしたがって、$a$とANDゲートを共有する変数を列挙し、これらの変数に対して制御性の高い順に上位に順序付ける。以上の2つの方針に従った変数の順序づけは、次のような手続きで行う。

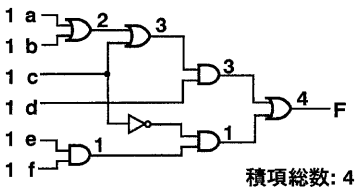
[変数順序決定アルゴリズム]

(1) まだ順序づけされていない変数のうち、制御性の最も高い変数を上位に順序づける。

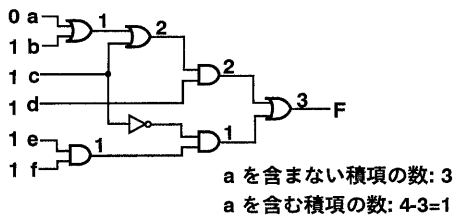
(2) その変数から出力までのパス上に存在する全てのANDゲートの入力につながる変数を列挙する。

(3) 列挙された変数でまだ順序が決定されていない変数を、制御性の高い順に上位に順序付ける。

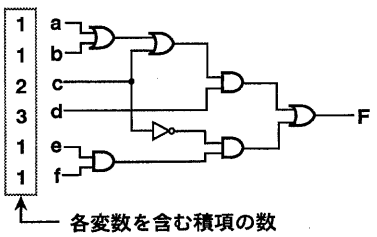
(4) すべての変数が順序づけされるまで、(1)～(3)を繰り返す。




(a) 最初の重み付け




(b) aの重みを0とした重み付け




(c) 各変数の制御性の算出結果

図3 重みづけの例

図3(c)に示す例では、制御性の最も高い変数である$d$が最上位に順序付けられる。次に、$d$とANDゲートによってつながる変数、$a$、$b$、$c$を制御性の高い順に$c$、$a$、$b$と順序づける。最後に残った$e$、$f$が順序づけられ、結果的に、$d$、$c$、$a$、$b$、$e$、$f$という順序付けが決定される。

これまでに、ゲート回路記述から良い変数順序を決定する手法がいくつか提案されている[6][7]。しかし、これらの手法によって決定される変数順序は回路の構成に依存し、回路が最適化されていない場合には、必ずしも良い変数順序が得られるとは限らない。

これに対して、本稿で提案する手法は回路の構成に依存しないため、回路が最適化されていない場合や、回路記述のない場合にも良い変数順序を与えることができる。

### 3.2 組み合わせ回路の分割照合

CONDORでは、10万ノード程度のBDDを扱うことができるが、大規模な組み合わせ回路に対してはBDDのノード数が10万を超え、論理照合できない場合がある。そこで、組み合わせ回路を一括してBDDに変換せず、部分回路に分割した上で部分回路毎にBDDに変換して論理照合を行う。

部分回路に分割するために、仕様に記述される中間信号名を、互いの参照関係を表すようなツリー形式で取り出す。例えば、図4(a)に示す仕様からは、図4(b)に示すような中間信号名情報が抽出される。

実際に設計される回路は、これらの中間信号をすべて含んでいるとは限らない。また、同じ名前の中間信号が回路中に存在しても、現実には仕様の記述と一致しない場合もある。そこで、ツリー形式で表された中間信号のうち、入力に近いものから回路中の中間信号との論理照合を行い、論理が一致した場合に限ってその中間信号で回路を分割するようにする。

例えば、図4(b)に示す中間信号によって図4(c)の回路を分割照合する場合を考える。まず、中間信号$T_1$、$T_2$、$T_4$について回路との論理照合を行う。$T_1$は回路と一致しないが、$T_2$、$T_4$は一致するので、$T_2$、$T_4$によって図4(d)のように回路を分割する。次に、$T_3 = a \cdot b + T_2 + \overline{T_2} \cdot d$について、分割された回路との論理照合を行う。そして$T_3$によって、図4(e)のように回路をさらに分割する。最後に$X = T_3 + T_4$について図4(e)の回路と論理照合を行う。

また、このように回路を分割することによって、回路に論理的誤りがある場合にその誤りを含む範囲を絞り込むことができる。例えば図4の例で、$T_3$と$T_4$についての照合結果は正しく、$X$の照合結果が誤りだった場合は、$X$から$T_3$、$T_4$までの部分に誤りが含まれることになる。

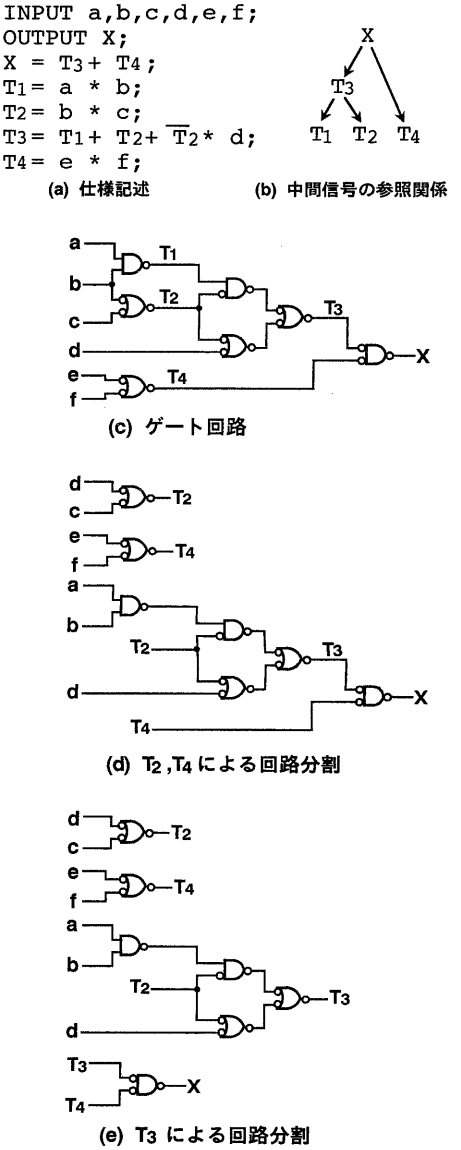



図4 組み合わせ回路の分割照合

## 4 実験と評価

CONDOR を EWS4800 ワークステーション (30Mips) 上に実現し、社内で設計された回路を対象に評価を行なった。その結果を表 1 に示す。

組合せ回路の分割照合の効果を確かめるために、組合せ回路を一括して照合する方式との比較を行なった。回路 C、F は大規模な組合せ回路を含むため一括照合方式では検証不可能であるのに対して、分割照合をすることによって 100% の検証が可能となり、検証時間も大幅に短縮することができた。しかし一方で、回路 D や E のような小規模な回路に対しては、分割照合をするよりも組合せ回路単位で一括して照合した方が検証時間が短い。これは、分割照合を行なうと論理照合の回数が増え、BDD 生成に必要なオーバーヘッド時間の影響が大きくなるためと考えられる。

また、シミュレーションを用いた既存の検証システムによる実験も行なった。その結果、シミュレーションによる検証と比べて CONDOR は検証時間を 30 分の 1 に短縮できることが実証された。

また、本稿で提案した BDD の変数順序づけ法の有効性を実証するために、BDD のノード数について、論理式に現れる順番で単純に順序づけを行なった場合との比較を行なった (表 2)。

## 5 むすび

現実の論理回路の設計検証に適用することを目的として開発した論理検証システム CONDOR について述べた。CONDOR は、与えられた機能仕様と回路の論理照合を BDD を利用して行い、検証するものである。現実の大規模な論理回路を扱うために、BDD のノード数の増加を抑制する方法として、入力変数の順序決定法と照合向きの回路分割法を提案した。評価実験の結果、提案した入力変数順序決定法により BDD サイズが削減できること、また、全体を BDD で一括して表現できない大規模な回路を、分割照合する事によって、検証できることを実証した。

表 1: 評価結果

| 回路 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 回路規模 (FB 数) | 79 | 130 | 335 | 611 | 1028 | 2002 |
| 組合せ回路総数 | 6 | 20 | 1 | 191 | 219 | 13 |
| CONDOR(分割照合) による検証時間 (秒) | 4 | 6 | 69 | 199 | 1512 | 161 |
| 検証不能な組合せ回路数 | 0 | 0 | 0 | 0 | 0 | 0 |
| CONDOR(一括照合) による検証時間 (秒) | 7 | 6 | 352 | 67 | 846 | 1767 |
| 検証不能な組合せ回路数 | 0 | 0 | 1 | 0 | 0 | 5 |
| シミュレーションによる検証時間 (秒) | 251 | 191 | — | 2520 | 19600 | 28800 |

## 参考文献


[1] Smith, G.L., et al., "Boolean Comparison of Hardware and Flowcharts", IBM J. of Res. and Develop. (Jan 1982), 106-116.

[2] S. B. Akers. "Binary Decision Diagrams", IEEE Transaction on Computers, Vol.C-27, No.6, pp.509-516, Jun. 1978.

[3] R. E. Bryant. "Graph-Based Algorithms for Boolean Functional Manipulation", IEEE Transactions on Computers, Vol.C-35, No.8, pp.677-691, Aug. 1986.

[4] Kato, S. and Sasaki, T., "FDL: A Structural Behavior Description Language", CHDL83 (1983), 137-152.

[5] S. J. Friedman, K. J. Supowit, "Finding the Optimal Variable Ordering for Binary Decision Diagrams," IEEE Transduction on Computers, Vol.39, No.5, pp.710-713, May. 1990.

[6] S. Minato, N. Ishiura, S. Yajima. "Shared Binary Dicision Diagram with Attributed Edges for Efficient Boolian Function Manipulatin", Proc. 27th ACM/IEEE DAC, pp.52-57, Jun. 1990.

[7] 藤田, 藤沢, 松永, 角田, "2 分決定グラフのための変数順序決定アルゴリズムとその評価 ", 情処論, Vol. 31, No.4, 532-541.


表 2: 変数順序づけによる BDD ノード数の違い

| 論理式 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 変数 | 57 | 53 | 38 | 17 | 15 | 11 |
| 変数順序 (1) | 324 | 113 | 92 | 59 | 33 | 15 |
| 変数順序 (2) | 758 | 307 | 160 | 88 | 49 | 31 |

変数順序 (1) : 本手法の順序づけによるノード数

変数順序 (2) : 論理式に現れる通りの順序によるノード数