

二次記憶に格納された共有二分決定グラフを効率良く処理するための 4 L - 4 幅優先アルゴリズム

越智 裕之[†] 安岡 孝一[‡] 矢島 脩三[†]

[†] 京都大学 工学部 情報工学教室

[‡] 京都大学 大型計算機センター

1 はじめに

共有二分決定グラフによる論理関数の表現 [1][2] は比較的コンパクトであり、しかも表現が一意であるという優れた性質がある。この共有二分決定グラフに基づく論理関数の処理手続きは、サブルーチンパッケージとしてワークステーション上に実現され [3][4]、論理照合、タイミング検証、テスト生成、論理合成、順序機械の設計検証などに広く応用されている。

しかし、より大規模・複雑な集積回路を設計するため、より大きな共有二分決定グラフを効率良く操作するための手法の開発が切望されている。高速化のため、グラフの操作を並列化する試みはこれまでにもみられ、効果が確認されている [5][6]。しかし、実用的には計算時間よりはむしろ計算機の主記憶容量の制約が問題となっている。

本報告では、全ての情報を主記憶上に格納できないような大きなグラフを取り扱うため、大容量の二次記憶に格納された共有二分決定グラフを効率良く操作する手法を提案する。これは、ランダムアクセスを行なうとアクセス速度が著しく低下する二次記憶を効率良く利用するため、従来深さ優先で行なわれていた共有二分決定グラフの操作を幅優先の操作に置き換え、グラフの各レベルごとに、必要な情報を二次記憶から取り出し、操作を行ない、結果を格納しようというものである。

2 準備

2.1 共有二分決定グラフ

共有二分決定グラフ (SBDD : Shared Binary Decision Diagram、または ROBDD : Reduced Ordered Binary Decision Diagram) は、非巡回有向グラフによる論理関数の表現法であり、(1) 入力変数の順序を固定すれば、論理関数に対してグラフの根が一意に定まる [1][2]、(2) 実用的な多くの論理関数を比較的少ないノード数で表現できる、(3) 論理関数同士の論理演算がグラフのノード数にほぼ比例する時間で効率よくおこなえる、などの優れた特長がある。

本報告では、図1のような否定エッジ付き準既約共有二分決定グラフを扱う。これは論理関数を表現する二分決定木の同形なサブグラフを可能な限り共有し、定数関数を表すサブグラフは全て定数を表す葉で置き換えて得られるグラフである。論理変数を表す各ノードから出る

2本のエッジはそれぞれ、一つ下のレベル(添字)の変数を表すノード、または定数を表す葉のいずれかを指す。否定エッジは、二分決定グラフのエッジに否定演算を表す属性を与えたもので、グラフをたどる際に、否定エッジを通った回数だけ論理値を反転させる。これにより、互いに否定の関係にあるサブグラフをも共有することができ記憶量が最大 50% 削減される。また、否定演算でグラフをたどる必要がなくなる、 $f = \bar{g}$ の判定が容易になる、などのことから処理速度も向上する。なお、否定エッジを無制限に使用するとグラフの一意性が失われてしまうため、(1) 各ノードの "0" エッジには否定エッジを使わない、(2) グラフの葉の値には 0 だけをを用いる、という制限を課す。図1で小さな丸印の付いたエッジが否定エッジである。

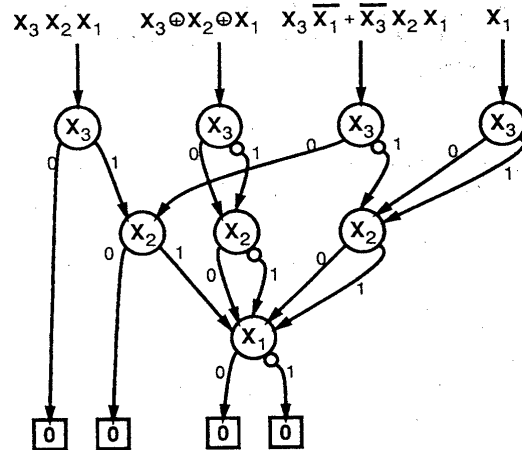


図1: 否定エッジ付き準既約共有二分決定グラフ

2.2 従来の論理演算アルゴリズム

例として、共有二分決定グラフで表された2つの論理関数 f 、 g の論理積 $h = fg$ を表すグラフを生成することを考える。 f または g が 0 の場合は、 $h = 0$ であるから、葉 (0) が解である。また、 $f = 1$ の場合、 $h = g$ であるから、 g とグラフを共有する ($g = 1$ の場合も同様)。 f を表すエッジと g を表すエッジが完全に一致している場合 (また、そのときに限り)、 $h = f = g$ なので、この場合も f や g とグラフを共有する。 f を表すエッジと g を表すエッジが否定属性のみ異なる場合 (また、そのときに限り) $f = \bar{g}$ であるから、 $h = 0$ なので、葉 (0) が解である。このように自明ではない場合、 $h|_{x_m=0} = f|_{x_m=0}g|_{x_m=0}$ および $h|_{x_m=1} = f|_{x_m=1}g|_{x_m=1}$ を表すグラフを生成し、 $h|_{x_m=0}$ を "0" エッジが指し、 $h|_{x_m=1}$ を "1" エッジが指す

変数 x_m のノードを生成して解とする。但し、 $h|_{x_m=0} = h|_{x_m=1}$ ならば新しいノードは生成しない。ここで変数 x_m は、論理関数 f または g が依存する変数の中でレベルが最大のものとし、上の手続きは再帰的に行う。

新しいノードを生成する際、生成しようとしたノードと等価なものがすでに存在していないかどうかを確認し、すでにあれば、それを共有しなければならない(さもなくば、グラフの一貫性が失われる)。このため、全てのノードをノードのレベル(変数の添字)、および"0" エッジ、"1" エッジをキーとしてハッシュテーブルに登録して管理する方法がとられる。このハッシュテーブルは節テーブルと呼ばれる。

また、同じ演算を何度も繰り返すことがないように、演算の結果が得られたらその結果を演算子および被演算数をキーとするハッシュテーブルに登録し、演算に先立って、すでに同じ演算が行われていないかをこのハッシュテーブルに照会する方法がとられる。このハッシュテーブルは演算結果テーブルと呼ばれる。

2.3 二次記憶

現在広く使われている計算機には、一般に主記憶よりも大きな記憶領域をもつ記憶媒体として二次記憶が備えられている。本報告で想定している二次記憶は次のようなものである。

1. ワークステーションのハードディスク

磁気記憶装置であり、連続する領域に格納された大量のデータは比較的高速にアクセスできる。

2. ベクトル計算機の拡張記憶

ダイナミック RAM などによる半導体補助記憶であり、主記憶とのデータ転送は数キロバイトのブロックを単位として高速に行なうことができる。

これらに共通の欠点として、記憶領域上に散在する小さなデータをランダムアクセスする場合には著しく速度が低下することがあげられる。

3 提案するアルゴリズム

従来の共有二分決定グラフの処理手法は深さ優先処理(再帰的手続き)に基づくものであるが、これでは共有二分決定グラフが格納されている記憶領域をランダムアクセスすることになり、グラフが二次記憶に格納されている場合は効率の良い処理を行なうことができない。そこで、準規約な共有二分決定グラフを幅優先処理に基づいて操作することを考える。

提案するアルゴリズムは、新しいノードを根から葉に向かって幅優先で生成していく展開フェーズと、展開フェーズで生成されたノードをグラフの葉から根に向かって順に幅優先でチェックして不要なものを取り除いていく正規化フェーズよりなる。

〔展開フェーズ〕与えられた論理関数同士の論理演算の結果が自明な場合や演算結果テーブルに結果がすでに登

録されている場合を除き、その結果を表す仮のノードを生成する。仮ノードを生成したら演算結果テーブルに登録し、さらにそのノードの"0"エッジと"1"エッジが指すべきノードを求める要求をキューにたくわえる。このキューが空になるまで仮ノードの生成を続ける。

〔正規化フェーズ〕展開フェーズで生成された仮ノードをレベルの低いものから順に、これと等価なノードがすでに節テーブルに登録されていないかどうかチェックする。等価なノードが節テーブルに登録されていない仮ノードは節テーブルに登録する。この操作をレベルの低い仮ノードから順に全ての仮ノードについておこなう。ここで、チェックしようとした仮ノードの"0"エッジや"1"エッジが節テーブルに登録されていないノードを指していた場合は、これと等価なノードが節テーブルに登録されているので、チェックに先立ちエッジをつなぎかえる。

準規約な共有二分決定グラフのノードの"0"エッジと"1"エッジが指すノードは葉(定数ノード)又は一つ下のレベルのノードであるから、展開フェーズ、正規化フェーズは共に、操作中のレベルのノード及びそれに隣接するレベルのノードに関する情報だけ主記憶上にあれば実行できる。即ち、あるレベルのノードの操作が終了したら結果を二次記憶に書き出し、次に操作するレベルのノードの情報を取り出せばよい。2つのハッシュテーブル(演算結果テーブル及び節テーブル)もレベル別に作り、ノードの情報と共に出し入れする。

なお、各レベルのノードなどの情報を保持する配列領域の大きさが必要以上に大きいと、2次記憶の読み書きに必要な以上の時間がかかってしまう。各レベルの配列領域の大きさがそのレベルのノード数に比べて大きくなりすぎないように、適宜ガベージコレクションを行なって配列の大きさを適正に保つ必要がある。

4 おわりに

本報告では大規模な共有二分決定グラフを扱うため、計算機の二次記憶上に格納されたグラフを効率良く操作するための手法を提案した。現在、提案したアルゴリズムに基づく論理関数処理システムの開発とその評価を進めている。なお、本研究の一部は文部省科学研究費補助金(特別研究員奨励費)による。

参考文献

1. S. B. Akers : Binary Decision Diagrams, IEEE Trans. Comput., vol. C-27, no. 6, pp. 509-516, (June 1978).
2. R. E. Bryant : Graph-Based Algorithms for Boolean Function Manipulation, IEEE Trans. Comput., vol. C-35, no. 8, pp. 677-691, (Aug. 1985).
3. S. Minato, N. Ishiura and S. Yajima : Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation, Proc. 27th ACM/IEEE DAC, pp. 52-57, (June 1990).
4. K. S. Brace, R. L. Rudell and R. E. Bryant : Efficient Implementation of a BDD Package, Proc. 27th ACM/IEEE DAC, pp. 40-45, (June 1990).
5. H. Ochi, N. Ishiura and S. Yajima : Breadth-First Manipulation of SBDD of Boolean Functions for Vector Processing, Proc. 28th ACM/IEEE DAC, pp. 413-416, (June 1991).
6. S. Kimura and E. M. Clarke : A Parallel Algorithm for Constructing Binary Decision Diagrams, Proc. 1990 IEEE ICCD, pp. 220-223, (Sep. 1990).