剰余 BDD を用いた算術演算回路の検証 *

6B－7

木村 晋二 †
奈良先端科学技術大学院大学 情報科学研究科 ‡

1 はじめに

論理回路の大規模化とともに、その形式的な設計検証への期待が高まっている。現在 BDD (Binary Decision Diagram, 二分決定グラフ) と呼ばれるデータ構造を用いた論理関数の処理手法が広く用いられている ([1], [2])。しかし、乗算回路など、BDD で表したときに入力数の指数の記憶容量が必要であるような回路が知られており、そのような回路に対する検証手法の確立が重要な問題となっている ([3])。

本稿では、算術演算回路で用いられている剰余数表現を応用して BDD の幅を剰余をとる数 p の多項式で押える手法とそれを用いた検証について述べる。

2 剰余を用いた算術回路の検証

算術演算回路の分野では、ある数 n を互いに素な k 個の数 $p_1, p_2, ..., p_k$ の各々に関する剰余 (modulo) の組で表すという剰余数表現 ([4]) が用いられ、回路規模の抑制に効果を上げている。そこでこれを検証へ応用すると、以下のようになる。

まず回路の出力ビット数 n に対し、互いに素な数 $p_1, p_2, ..., p_k$ を、$p_1 \times p_2 \times ... \times p_k \geq 2^n$ となるように選ぶ。

つぎに各 p_i について、p_i で剰余をとる操作を回路および仕様に付加し、剰余をとった結果の関数を比較する。すべての p_i で出力関数が等価であれば剰余数表現の理論 ([4]) から仕様と回路の等価性が保証される。出力関数の等価性は、BDD を用いることで判定できる。

算術演算の出力について p で剰余をとった結果を表す BDD では、変数順序によらず同じラベルを持つ節点数 (BDD の幅) が p の多項式以下になる。ただし、演算の途中結果の BDD の幅については、この性質は成り立たない。また、このような直接的な応用では、途中で通常の回路の出力を計算するので、BDD が節点爆発する場合には使用できない。そこで演算の途中結果の BDD の幅を剰余をとる数 p の多項式で限定する手法を提案する。

*Verification of Arithmetic Circuits using Residue BDD's
†Shinji KIMURA
‡Nara Institute of Science and Technology

3 剰余 BDD

3.1 剰余 BDD

もっとも単純には、まず入力の剰余を取った後に演算を行い、さらに出力についてもう一度剰余をとるという手法が考えられる。$(a \circ b)_{\mathrm{mod}p} = ((a)_{\mathrm{mod}p} \circ (b)_{\mathrm{mod}p})_{\mathrm{mod}p}$ の右辺を行うわけである。これは非常に効率が良いが、左辺と右辺の算術演算は同じ演算ではないので、このままでは回路の検証へ適用できない。

そこで論理演算毎に幅を限定する以下のような演算を考え、論理回路の結線情報から BDD を構成する時に、この幅限定操作を行う。限定操作では、BDD の各節点への経路に対応する数を用いる。

ここでは、n 変数論理関数 $f(x_{n-1}, x_{n-2}, ..., x_0)$ に対応する BDD を考える。入力に関しては、x_i を 2^i に対応づける。

f を表す BDD において、ラベル x_i を持つ節点 v へ至る経路を考えると、経路上の各節点でどちらの枝を選ぶかに応じて、経路に対応して二進数を対応づけることができる。この二進数の剰余をとって、経路に対する剰余数を定義する。

上記の剰余数の定義では、経路に現れない入力変数について 0 を選んでいることに注意する必要がある。0 を選ぶ以外に、経路に現れない入力変数について 1 であるとして剰余数を定義することや、処理する BDD をレベライズするように変形してから剰余数を定義する方法等が考えられる。

上記のように剰余数を定義すると、同じ変数ラベルを持つ BDD の節点について、同じ剰余数を持つものが生じる。それらを一つで代表させたものを剰余 BDD と呼ぶと、この BDD の幅は p 以下になる。限定操作を $[]_{\mathrm{mod}p}$ と表すと、論理演算 ($\diamond$) に関しては以下の性質が成立する ([5])。

- 論理演算 $\diamond$ について、$f = g \diamond h$ とすると、$[f]_{\mathrm{mod}p} = [[g]_{\mathrm{mod}p} \diamond [h]_{\mathrm{mod}p}]_{\mathrm{mod}p}$ である。
- 入力の剰余のみに依存する論理関数 f については、$[f]_{\mathrm{mod}p} = f$ である。

以上の性質から入力の剰余のみに依存する論理関数に対応する BDD の計算では、計算途中で生成された

表 1: 16 ビット乗算器の比較

法	時間 (秒)		総節点数		出力
	AA	C6288	AA	C6288	節点数
29	5517	4266	86545665	69889435	25713
23	2198	1700	37149663	29473623	15931
19	1197	887	21004722	15596155	12323
17	849	636	15008271	11716224	5278
13	351	216	5838646	3601627	4479
11	225	137	3822546	2337087	3840
7	83	37	1226558	542226	800
5	48	22	618016	237615	596
3	30	14	254838	148073	207
2	12	6	9083	8274	2

AA : 配列型　　C6288 : ISCAS ベンチマーク

BDD を剰余 BDD に変換しても良い。

二引数の算術演算に対応する論理関数では、引数毎で独立に剰余を計算すれば、幅は p^2 以下で限定できる。

3.2 分割検証

二引数の演算に対しては剰余 BDD の幅が最大で p^2 となるが、これは分割して扱うことにより、p 以下にできる。具体的には、二引数の一方の引数の経路数の剰余が現在選択している剰余 i 以外の場合に論理値 0 にすることにより、法 p に対して剰余 0, 1, ..., $p-1$ の各場合を分割して扱う。

分割により、各剰余で必要な節点数は法 p のオーダー ($O(p)$) となり、記憶容量の点から有用である他に、各場合を並列に実行することができるので、並列計算環境を用いることで高速に処理ができる。さらに、検証すべき場合の数もそれほど大きくはならない。

4 算術演算回路の検証例

以下、乗算器を例にして、等価性判定を行なったときの計算手法および計算時間等を述べる。比較対象の乗算器としては、配列型の乗算器と、ISCAS ベンチマークの C6288 の乗算器を用いた。プログラムの実現には、SUN SS10/51 (50 MHz SuperSPARC, 1MB SuperCache, 64 MB Main Memory) を用いた。

実験では、16 ビットの配列型の乗算器と ISCAS ベンチマークの C6288 に対し、これらの出力に剰余を計算する回路を付け、最終的な出力の BDD を計算してファイルに出力し、それらのファイルの比較を行なった。また、入力変数順は、$a_{n-1}, a_{n-2}, ..., a_0, b_{n-1}, b_{n-2}, ..., b_0$ とした。

表 1 に比較結果を示す。互いに素な数としては、2, 3, 5, 7, 11, 13, 17, 19, 23, 29 の 10 個の素数を用いることにした。これらを掛け合わせると 6469693230 となり、2^{32}(= 4294967296) よりも大きな数となる。AA が配列型を、C6288 が ISCAS ベンチマークを表す。表には、剰余をとった数、構成に必要な CPU 時間 (秒)、構成された総節点数、出力の節点数を示す。出力ファイルの比較時間は示していないが、出力節点数に比例したサイズのファイルとなり、すべての場合で数秒以内であった。なお、BDD の最大の節点数を 150 万節点として構成を行った。150 万節点を越える場合は、ガーベージコレクションを行ないながら作成した節点の数の総数である。

法が大きくなるに従い、時間および節点数が大きくなっているが、これは本来必要とする節点数が法 p の二乗のオーダー ($O(p^2)$) であるためである。実行時間については分割手法で対処できると考えている。

5 おわりに

本稿では、剰余 BDD を提案し、その性質および、算術演算回路の検証への適用について示した。剰余数表現では、数を互いに素な k 個の数字の剰余の組で表し、加算、減算、乗算などを剰余毎で独立に行なう。ここではそれを用い、節点爆発しない BDD を示した。

剰余 BDD では、変形により回路の情報をある程度落しているので、剰余 BDD を用いた場合の検証能力を明らかにする必要がある。また、今後順序回路への本手法の適用、プロセサレベルでの検証、多項式に比例した節点数で表せる BDD のクラスに関する理論的な考察などを行ないたいと考えている。

謝辞 剰余数表現について御教示いただいた名古屋大学高木直史先生に感謝します。また、日頃から御討論いただく本学渡邊勝正教授はじめ渡邊研究室の皆様に感謝します。とくに、永見康一君と中江達哉君には本稿に関するコメントをいただきました。ここに記して感謝します。

References

[1] R. E. Bryant. Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. *ACM Computing Surveys*, Vol. 24, No. 3, pp. 293–318, Sep. 1992.

[2] S. Minato, N. Ishiura, and S. Yajima. Fast Tautology Checking Using Shared Binary Decision Diagram –Benchmark Results–. In *Proc. IFIP International Workshop on Applied Formal Methods for Correct VLSI Design*, pp. 580–584, Nov. 1989.

[3] Jerry R. Burch. Using BDD to Verify Multipliers. In *Proc. of 28th DA Conference*, pp. 408–412, 1991.

[4] Norman R. Scott. *Computer Number System & Arithmetic, Chapter 7.6.* Prentice Hall, 1985.

[5] 木村晋二. 剰余 BDD およびそれを用いた算術演算回路の検証. 信学技報 COMP94–73, pp. 59–68, Dec. 1994.