社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2005－SLDM－122 (30)
2005／12／2

限量子付ブール式の充足可能性判定を用いた論理式の最小因数分解手法

吉田 浩章† 池田 誠†† 浅田 邦博††

† 東京大学 大学院 工学系研究科 電子工学専攻
†† 東京大学 大規模集積システム設計教育研究センター (VDEC)
〒 113–8656 東京都文京区本郷 7–3–1
E-mail: †{hiroaki,ikeda,asada}@silicon.u-tokyo.ac.jp

あらまし 本稿では不完全定義論理関数の最小因数分解形表現を発見する厳密手法を提案する．提案手法では因数分解問題を限量子付ブール式として表現し，この解を汎用の充足可能性判定手法によって求める．また我々は X-B 木と呼ばれる新しいグラフ構造を提案する．X-B 木は暗黙的にすべての可能な二分木構造を網羅しており，これを用いることにより因数分解問題を効率的に表現することが可能となっている．最後に例題を用いた計算機実験の結果を示し，本手法の妥当性を示す．

キーワード 限量子付ブール式，充足可能性判定，因数分解，二分木

Exact Minimum Logic Factoring via Quantified Boolean Satisfiability

Hiroaki YOSHIDA†, Makoto IKEDA††, and Kunihiro ASADA††

† Department of Electronic Engineering, University of Tokyo
†† VLSI Design and Education Center(VDEC), University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–8656, Japan
E-mail: †{hiroaki,ikeda,asada}@silicon.u-tokyo.ac.jp

Abstract This paper presents an exact algorithm which finds the minimum factored form of an incompletely specified Boolean function. The problem is formulated as a Quantified Boolean Formula (QBF) and is solved by general-purpose QBF solver. We also propose a novel graph structure, called an X-B tree, which implicitly enumerates binary trees. Using this graph structure, the factoring problem is compactly transformed into a QBF and hence the size of solvable problems is extended. Experimental results show that the proposed algorithm successfully finds the exact minimum solutions to the problems with up to 12 literals.

Keywords Quantified Boolean formula, Boolean satisfiability, logic factoring, binary tree

1. はじめに

論理式の因数分解 (logic factoring) は与えられた論理関数を表現する因数分解形の論理式を求める操作である．因数分解形は論理関数の効率的な表現方法の一つであることが知られており，多段論理合成技術の基礎となっている．また因数分解形はスタティック CMOS ゲートに対応するため，ブーリアンネットワーク [1] の回路面積の見積もりにも利用される．例えば，図 1 に示すスタティック CMOS ゲートは $y = \overline{((ab + c) * d) + ef}$ という因数分解形論理式を実現している．

論理式の因数分解は多段論理合成における最も基本的な問題の一つであるものの，厳密解を求める効率の良い解法はいまだ知られていない．初期の研究 [2] [3] では厳密な多段論理最小化アルゴリズムを提案しているものの，効率が悪く非常に小さな関数にのみ適用可能であった．最近になり，[3] の効率を改善したアルゴリズムが提案されたが，それでも 12 個の 2 入力 NAND ゲートからなる回路の合成に数時間を要する [4]．またその計算複雑性に加えて，このアルゴリズムは 2 入力 NAND ゲートの数を最小化することをその目的としており，最小因数分解形を求めるものではない．

近年，命題論理の充足可能性判定手法は劇的な進歩を遂げており [5] [6]，EDA の分野においても自動テストパターン生成 [7] や記号モデル検査 [8] などで現実的な規模の問題に適用されるまでになっている．一方で，命題論理の一般化の一つである限量子付ブール式 (QBF; Quantified

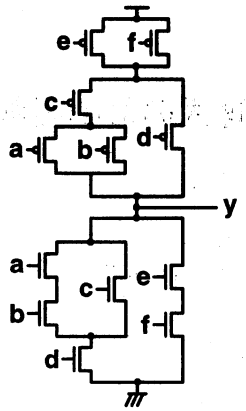

図1　スタティック CMOS ゲート

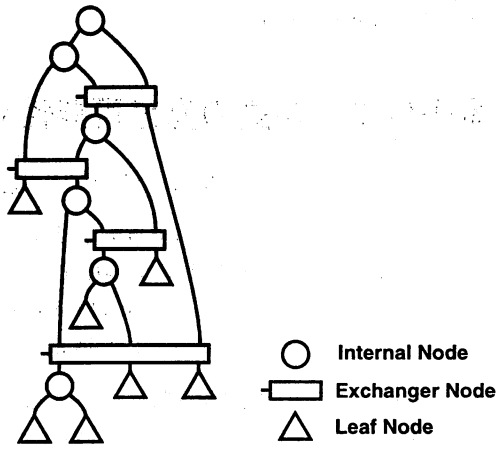

図2　6個の内部ノードを持つ X-B 木

Boolean formula) はこの種の問題をより自然にモデルすることが可能であることが知られている．このような背景の下で，QBF の充足可能性判定においてもまた活発な研究が行われており，効率の良い判定手法が提案されている [9]-[12].

本稿では，論理式の因数分解問題を QBF によって表現し，汎用ソルバによって充足可能性を判定することによって最小解を求める厳密手法を提案する．充足可能性判定の計算時間は QBF の大きさによって支配的に決定するため，因数分解問題に対応する QBF は可能な限り小さくする必要がある．この目的で，我々は X-B 木と呼ばれる新しいグラフ構造を提案する．X-B 木は暗黙的にすべての可能な二分木構造を網羅しており，これを用いることにより因数分解問題を効率的に表現することが可能となっている．最後に例題を用いた計算機実験の結果を示し，本手法の妥当性を示す．

2. X-B 木とその生成手法

2.1 X-B 木

二分木は 2 個の子ノードを持つ内部ノードと葉ノードからなるグラフと定義される．ここで子ノードは内部ノードもしくは葉ノードである．X-B(eXchanger Binary) 木は交換ノードと呼ばれる新しい種類のノードを持つ，根付き無順序二分木 (rooted unordered binary tree) である．交換ノードは同数の入力と出力を持ち，ただ一つの入力のみが内部ノードであり，残りの入力は葉ノードもしくは交換ノードである．交換ノードは交換番号 c_x に基づいて入出力の接続関係を決定する．交換番号 c_x は n を交換ノードの入力数として正の整数 $1 \le c_x \le n$ をとり，このとき入出力の関係は以下のようにして与えられる．

$$o_j = i_{((c_x+j-2) \bmod n)+1} \quad (1 \le j \le n) \tag{1}$$

ここで，i_j は j 番目の入力，o_j は j 番目の出力を表す．このように交換ノードはビットシフタのように振る舞う．図2に6個の内部ノードを持つX-B木を，図3に3入力交換ノードの例を示す．

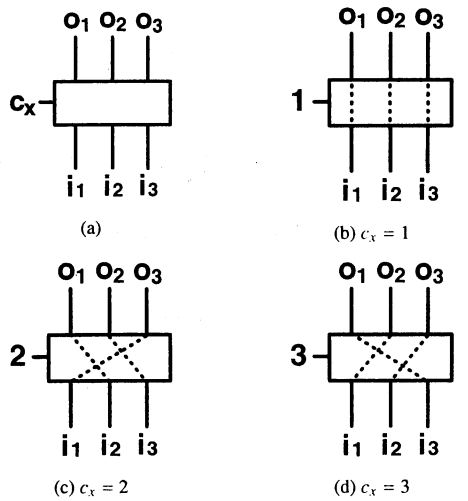

図3　3 入力交換ノードの例

すべての交換ノードの交換番号が決定すると，ある二分木が得られる．この交換番号の割り当てを網羅することで，すべての可能な無順序二分木構造を得ることができる．ここで，交換番号の割り当てと得られる二分木の間には必ずしも一対一対応が成り立つわけではないこと，つまり異なる割り当てから同じ二分木が得られる場合があることに注意したい．

後述するように因数分解問題を QBF として定式化する際に，X-B 木のそれぞれの交換ノードの交換番号を複数の二値変数列によって表現する．ある X-B 木のすべての交換番号を表現するために必要な二値変数の総数を総交換ビット数 n_b と定義し，以下の式で与えられる．

$$n_b = \sum_i \lceil \log_2 n_i \rceil \tag{2}$$

ここで $\lceil x \rceil$ は x 以上の最小の整数，n_i は i 番目の交換ノードの入力数である．

```
Procedure ComputeSignature(Tree T)
  S=ComputeSignatureRecursive(GetRootVertex(T))
  return OmitLastBit(S)
end Procedure

Procedure ComputeSignatureRecursive(Vertex V)
  if V is a leaf node
    return "0"
  end if
  S_L=ComputeSignatureRecursive(GetLeftChild(V))
  S_R=ComputeSignatureRecursive(GetRightChild(V))
  if S_L > S_R
    return "1"+S_L+S_R
  else
    return "1"+S_R+S_L
  end if
end Procedure
```

図4　無順序木のビット文字列表現の計算の基本手続き

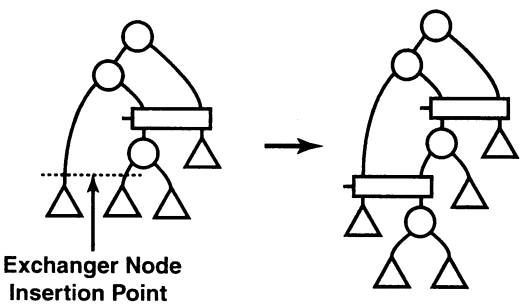

図5　交換ノードの挿入

2.2　無順序二分木の同形判定

X-B 木の生成において，二つの無順序二分木の同形 (isomorphism) を判定する必要がある．本提案手法ではビット文字列表現 (bitstring representation) [13] [14] を利用する．しかしながらビット文字列表現は順序木を対象としているため，直接使用することはできない．そこで我々はこれを拡張し，無順序木のビット文字列表現としている．このビット文字列表現は 2 進数の列 $b_1 b_2 \ldots b_n$ であり，図 4 に示す手続きによって再帰的に求められる．この手続きにおいて最後のビットは常に 0 であるため省略される．

2.3　X-B 木の生成手法

ある数の葉ノードを持つ X-B 木は複数存在するが，本稿では特に最小の総交換ビット数を持つ X-B 木を生成する手法について説明を行う．本手法ではまず可能な X-B 木の網羅を行い，次にその中から最小のものを選択することによって，最小 X-B 木の生成を行う．

提案手法は建設的なアプローチであり，$n-1$ 個の葉ノードを持つ X-B 木から n 個の葉ノードを持つ X-B 木を生成する．手続きはまず 1 つの内部ノードと 2 つの葉ノードからなる二分木から始める．ある X-B 木が与えられると，複数の葉ノードとそれらの親ノードの集合を決定する．次に図 5 に示すように葉ノードと親ノードの間に交換ノードを挿入する．

挿入個所は以下のようにして決定する．まず X-B 木中のある葉ノードを内部ノードと 2 つの葉ノードに置き換える．次に，すべての交換番号の割り当てを網羅することで，この X-B 木に含まれるすべての二分木のビット文字列表現を求める．これをすべての葉ノードに対して行った後，行がビット文字列表現に，列が葉ノードに対応する被覆表を構築する．この被覆問題を解くことによって，葉ノードの集合が得られる．この葉ノードの集合とそれらの親ノードの間に交換ノードが挿入される．表 1 に最小 X-B 木の主な性質を示す．

表1　最小 X-B 木の主な性質

内部ノード数	葉ノード数	二分木総数	交換ノード数	総交換ビット数
1	2	1	0	0
2	3	1	0	0
3	4	2	1	1
4	5	3	2	2
5	6	6	3	3
6	7	11	4	5
7	8	23	5	7
8	9	46	6	9
9	10	98	7	11
10	11	207	8	13
11	12	451	9	15
12	13	983	10	17
13	14	2179	11	20
14	15	4850	12	22
15	16	10905	13	25

3.　因数分解手法

3.1　問題の定式化

リテラルはある変数またはその否定である．因数分解形論理式は論理関数の一表現であり，以下のように帰納的に定義できる．1) リテラルは因数分解形である，2) 因数分解形の論理和は因数分解形である，3) 因数分解形の論理積は因数分解形である．一般的に，ある論理関数の因数分解形は一意に決まらない．例えば，以下の 3 つの論理式

$$abc + abd + cd$$
$$ab(c + d))cd$$
$$abc + (ab + c)d$$

はいずれも同じ論理関数の因数分解形である．ある論理関数の因数分解形がすべての可能な因数分解形の中で最

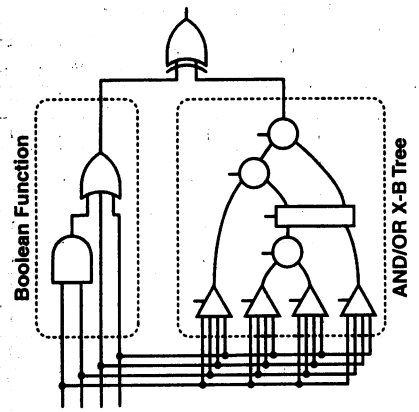

図 6　因数分解問題の miter 構造によるモデル化

も少ないリテラル数を持つとき，この因数分解形を最小因数分解形という．

AND/OR 二分木はそれぞれの内部ノードが 2 入力 AND ゲートまたは 2 入力 OR ゲートに対応する根付き二分木である．葉ノードをリテラルとみなすことにより，任意の因数分解形論理式は AND/OR 二分木で表現できる．

本提案手法では，$V = \{v_1, \ldots, v_{|V|}\}$ を変数とする不完全定義論理関数 (f, d, r) を実現する最小因数分解形論理式を発見することをその目的とする．実際には，(f, d, r) を実現する最小数の葉ノードを持つ AND/OR 二分木を発見することによってこの目的を達成する．

3.2 QBF の構築

上で定式化した因数分解問題は miter 構造 [15] によってモデルすることができる (図 6)．このモデルでは，与えられた不完全定義論理関数と AND/OR X-B 木の等価性を検証している．ここで，AND/OR X-B 木は X-B 木に以下の変更を施したものである．まず内部ノードは演算ノードと呼ばれ，2 入力 AND ゲートまたは 2 入力 OR ゲートに対応する．それぞれの演算ノードは変数 $c_o \in \{0, 1\}$ を持っており，これによってノードの種類 (2 入力 AND ゲートまたは 2 入力 OR ゲート) を決定する．図 7 に演算ノードとその等価論理回路を示す．葉ノードはリテラルノードと呼ばれ，あるリテラルに対応する．それぞれのリテラルノードは変数 $c_l \in \{1, \ldots, 2|V|\}$ を持っており，これによって対応するリテラル $l \in \{v_1, \overline{v_1}, \ldots, v_{|V|}, \overline{v_{|V|}}\}$ を決定する．図 8 にリテラルノードとその等価論理回路を示す．この c_o と c_l と前述の交換番号 c_x の 3 種類の変数を構成変数と呼ぶ．ある数の葉ノードを持つ任意の AND/OR X-B 木は，ある構成変数の割り当てを持つ AND/OR X-B 木によって表現できる．

因数分解問題の QBF への定式化はこのモデルに基づいて行われる．QBF のそれぞれの論理和節は，モデル中の対応するノードの種類に応じて関数制約，演算ノード制約，交換ノード制約，リテラルノード制約の 4 種類に分類できる．以下にそれぞれの制約の構築手法を示す．

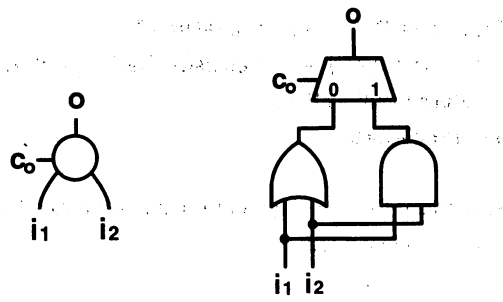

図 7　(a) 演算ノードと (b) 等価論理回路

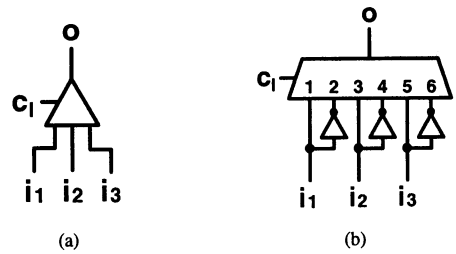

図 8　(a) リテラルノードと (b) 等価論理回路

関数制約: AND/OR X-B 木の根における演算ノードの出力変数を o_{root} とする．関数制約では与えられた不完全定義論理関数と AND/OR X-B 木の出力の等価性を検証する．

$$\xi_f = (f \equiv o_{root}) + d \tag{3}$$

ここで f と d はそれぞれ不完全定義論理関数のオンセット関数とドントケア関数を表す．

演算ノード制約: 演算ノード制約は AND/OR X-B 木中の演算ノードを表現する．それぞれの演算ノードに対して，以下の式が構築される．

$$\xi_o = \overline{c_x}\big(o \equiv (i_1 + i_2)\big) + c_x\big(o \equiv (i_1 \cdot i_2)\big) \tag{4}$$

ここで i_1 と i_2 はそれぞれ演算ノードの 1 番目と 2 番目の子ノードの出力変数に対応する．

交換ノード制約: n を正の整数 $1 \leq n \leq m$ とすると，n の論理積表現は以下のようにして与えられる．

$$\mathcal{CUBE}(x, m, n) = \prod_{i=1}^{\lceil \log_2 m \rceil} (x_i \equiv b_i) \tag{5}$$

ここで $b_1 b_2 \ldots b_{\lceil \log_2 m \rceil}$ は $n-1$ を 2 進数列で表現したものである．例えば，

$$\mathcal{CUBE}(4, 1) = \overline{x_1} \cdot \overline{x_2}$$
$$\mathcal{CUBE}(4, 2) = x_1 \cdot \overline{x_2}$$
$$\mathcal{CUBE}(4, 3) = \overline{x_1} \cdot x_2$$
$$\mathcal{CUBE}(4, 4) = x_1 \cdot x_2$$

である．

交換ノード制約は AND/OR X-B 木中の交換ノードを表現する．それぞれの演算ノードに対して，以下の式が構築される．

$$\xi_o = \sum_{i=1}^{n}\left(\mathcal{CUBE}(c_o, n, i)\prod_{j=1}^{n}(o_j \equiv i_{((i+j-2) \bmod n)+1})\right)\cdot \overline{\sum_{i=n+1}^{2^{\lceil \log_2 n\rceil}} \mathcal{CUBE}(c_o, n, i)} \quad (6)$$

ここで n は交換ノードの入力数，o_j は交換ノードの j 番目の出力に対応する変数，i_j は交換ノードの j 番目の入力の子ノードの出力に対応する変数である．

リテラルノード制約: リテラルノード制約は AND/OR X-B 木中のリテラルノードを表現する．それぞれのリテラルノードに対して，以下の式が構築される．

$$\xi_l = \sum_{i=1}^{|V|}\left(\mathcal{CUBE}(c_l, 2|V|, 2i-1)(o \equiv v_i) + \mathcal{CUBE}(c_l, 2|V|, 2i)(o \equiv \overline{v_i})\right)\cdot \overline{\sum_{i=2|V|+1}^{2^{\lceil \log_2 2|V|\rceil}} \mathcal{CUBE}(c_l, 2|V|, i)} \quad (7)$$

演算ノード，交換ノード，リテラルノードの出力に対応する変数の集合を $O = \{o_1, \ldots, o_{|O|}\}$ とし，構成変数の集合を $C = \{c_1, \ldots, c_{|C|}\}$ とする．因数分解問題モデルに対応する最終的な QBF ξ はすべての制約の論理積に，普遍限量子 $\forall$ と存在限量子 $\exists$ を導入したものとなる．

$$\xi = \exists C \forall V \exists O\, \xi_f\left(\prod_{i=1}^{L-1}\xi_{o_i}\right)\left(\prod_{i=1}^{X}\xi_{x_i}\right)\left(\prod_{i=1}^{L}\xi_{l_i}\right) \quad (8)$$

ここで ξ_{o_i}，ξ_{x_i}，ξ_{l_i} はそれぞれ i 番目の演算ノード，交換ノード，リテラルノードに対応し，L と X はそれぞれ AND/OR X-B 木中のリテラルノードと交換ノードの総数である．

3.3 最小因数分解形発見手法

L 個のリテラルノードを持つ AND/OR X-B 木に対して，式 (8) によって QBF ξ が構築される．もし ξ が充足可能であれば，リテラル数 L 以下の因数分解形論理式が存在することを意味する．提案手法ではリテラルノード数 $L = |V|$ から始め，L を増加させていくことによって最小因数分解形を発見する．ある L において，もし ξ が充足可能であれば，そのときの構成変数の割り当てから最小因数分解形を求める．充足可能でない場合には，L を 1 増加し ξ を計算する．この手続きを最小解が発見されるまで繰り返す．図 9 に最小因数分解形発見の基本的手続きを示す．ここで，$|V|$ より小さいリテラル数を持つ因数分解形は存在しないことに注意したい．これはどのような因数分解形においてもすべての変数が少なくとも 1 度は出現しなければならないためである．

```
Procedure ExactFactor(f, d)
  L=|V|
  do
    ξ=ConstructQBF(f, d, L)
    ExecuteQBFSolver(ξ)
    if ξ is satisfiable
      A=GetSatisfiableAssignment(ξ)
      F=ConstructFactoredForm(A)
      return F
    end if
    L=L+1
  while true
end Procedure
```

図 9 最小因数分解形発見の基本的手続き

4. 計算機実験結果

提案手法 Exact Factor を我々が開発した論理操作クラスライブラリ Logica を用いて C++言語によって実装し，例題に対する計算機実験を行った．まず，我々は QBF の充足可能性判定手法として ssolve [9]，SEMPROP [10]，sKizzo [11]，Quantor [12] の 4 つの汎用ソルバの評価を行った．この結果より，これらのツールの中から複数の例題を最も高速に判定可能であった sKizzo を選択した．

今回は例題として，我々が作成した 9 個の関数と MCNC91 ベンチマークから `majority` を選択した．`majority` 以外の関数は `algebraic` グループと `Boolean` グループに分類される．`algebraic` グループに属する関数はブール代数固有の性質 ($a \cdot a = a$ など) を用いることなく最小積和形論理式から最小因数分解形を求めることができる関数である．一方で，`Boolean` グループに属する関数はブール代数固有の性質を用いた場合においてのみ最小積和形論理式から最小因数分解形を求めることができる関数である．

実験結果を表 2 に示す．参考として，二段論理最小化ツール ESPRESSO [16] による結果とヒューリスティック因数分解アルゴリズム Good Factor [1] による結果を示している．表において左端の 2 列は関数名と関数の変数の総数を示す．次の 3 列はそれぞれ，ESPRESSO によって生成された積和形論理式のリテラル数，Good Factor による因数分解形のリテラル数，提案手法による因数分解形のリテラル数を表す．6 番目から 8 番目の 3 列はそれぞれ，充足可能な最終的な QBF の変数の総数，論理和項の数，リテラル数を表す．最後の列は最小因数分解形を求めるために要した計算時間を秒で表す．

提案手法は厳密最小解を求めることができるため，実験結果においては結果の品質と実行時間が重要である．実

表2 計算機実験結果

関数名	変数の総数	*ESPRESSO* [16]	*Good Factor* [1]	提案手法 (*Exact Factor*)				
		リテラル数	リテラル数	リテラル数	*QBF*			計算時間
					変数の総数	論理和項数	リテラル数	[秒]
majority	5	13	10	**9**	90	251	825	2.6
algebraic1	8	14	8	8	81	245	991	1.8
algebraic2	6	36	11	11	115	458	2014	483.9
algebraic3	6	15	11	**10**	103	409	1796	200.7
algebraic4	9	19	9	9	103	299	1046	12.0
boolean1	5	11	8	**6**	54	194	813	0.5
boolean2	6	26	16	**10**	103	413	1817	65.0
boolean3	5	13	10	**8**	78	287	1176	10.2
boolean4	6	22	18	**11**	115	458	2014	466.3
boolean5	6	30	20	**12**	127	383	1349	319.9

験結果より既存手法の Good Factor は algebraic グループに属する関数に対しては最小に近い解を生成していることが確認できる．しかしながら，Boolean グループに属する関数については，Good Factor の結果は十分に最小に近いとはいえない．例えば，例題関数 boolean4 のそれぞれの手法によって生成された論理式を以下に示す．

ESPRESSO: $ab\bar{c}\bar{d} + ab\bar{e} + \bar{a}\bar{b}cd + \bar{a}\bar{b}ef + cd\bar{e} + \bar{c}\bar{d}ef$

Good Factor: $cd\bar{e} + \bar{c}\bar{d}ef + \bar{a}\bar{b}(ef + cd) + ab(\bar{e} + \bar{c}\bar{d})$

Exact Factor: $(ab + cd + ef)(\bar{a}\bar{b} + \bar{c}\bar{d} + \bar{e})$

また13以上のリテラルを持つ例題に対しては，これを提案手法は1時間以内で解くことができなかった．これはQBFの大きさが増大してしまったためである．QBFの大きさは主にリテラル数(AND/OR X-B木のリテラルノード数)と関数の変数の総数によって決定される．QBFの充足可能性判定アルゴリズムは現在も活発な研究が行われているため，将来にはより大きな問題が解決可能になることが期待される．

5. ま と め

論理式の因数分解は多段論理合成技術の基礎である一方，最も困難な問題の一つである．本稿では，不完全定義論理関数を実現する最小因数分解形論理式を求める厳密手法を提案した．提案手法では因数分解問題を限量子付ブール式によって定式化し，これを汎用のソルバによって解くことにより最小解を求める．また我々はX-B木と呼ばれる新しいグラフ構造を提案した．このX-B木を用いることで，因数分解問題を効率的にQBFに表現することが可能になっている．例題を用いた計算機実験では，提案手法によって12リテラル以下の最小因数分解形を求めることが可能であったことを示した．解決可能な問題の大きさは比較的小さいものの，提案手法はスタンダードセルの回路・論理設計などの応用に有用であると考えられる．

文 献

[1] R. K. Brayton *et al.*, "MIS: A Multiple-level Logic Optimization System," *IEEE Trans. on Computer-Aided Design*, vol. 6, no. 6, pp. 1062–1081, Nov. 1987.

[2] E. L. Lawler, "An Approach to Multilevel Boolean Minimization," *Journal of ACM*, pp. 283–295, 1964.

[3] E. Davidson, "An Algorithm for NAND Decomposition under Network Constraints," *IEEE Trans. on Computers*, vol. 18, pp. 1098–1109, 1969.

[4] R. Drechsler and W. Gunther, "Exact Circuit Synthesis," *Int. Workshop on Logic Synthesis*, 1998.

[5] J. P. M. Silva and K. A. Sakallah, "GRASP—A New Search Algorithm for Satisfiability," in *Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 220–227, Nov. 1997.

[6] M. Moskewicz *et al.*, "Chaff: Engineering an Efficient SAT Solver," in *Proc. of ACM/IEEE Design Automation Conference*, pp. 530–535, Jun. 2001.

[7] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[8] A. Biere *et al.*, "Symbolic Model Checking without BDDs," in *Proc. of ACM/IEEE Design Automation Conference*, pp. 193-207, Jun. 1999.

[9] R. Feldmann *et al.*, "A Distributed Algorithm to Evaluate Quantified Boolean Formulas," in *Proc. National Conf. on Artificial Intelligence*, pp. 285–290, 2000.

[10] R. Letz, "Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas," in *Proc. TABLEAUX2002, vol. 2381 of LNAI*, pp. 160–175, 2002.

[11] M. Benedetti, "sKizzo: a QBF decision procedure based on Propositional Skolemization and Symbolic Reasoning," *Tech. Rep. TR04-11-03*, ITC-Irst, Nov. 2004.

[12] A. Biere, "Resolve and Expand," in *Proc. Intl. Conf. Theory and Applications of Satisfiability Testing*, LNCS, Springer 2005.

[13] A. Proskurowski, "On the Generation of Binary Trees," *Journal of ACM*, vol. 27, no. 1, pp. 1–2, Jan. 1980.

[14] S. Zaks, "Lexicographic Generation of Ordered Trees," *Theoretical Computing Science*, vol. 10, pp. 63–82, 1980.

[15] D. Brand *et al.*, "Verification of Large Synthesized Designs," in *Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 534–537, Nov. 1993.

[16] R. K. Brayton *et al.*, Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Boston, 1984.