設 計 自 動 化 60−20
（1991．12．13）

二分決定グラフからの組合せ論理回路の合成

石浦 菜岐佐
大阪大学工学部情報システム工学科

論理関数の二分決定グラフ表現から直接組合せ論理回路を合成する一手法を提案する．本手法は積和形表現を経由しないため，積和形表現が入力変数の指数規模となってしまう，パリティ関数や対称関数の一部に対しても実用的な計算量で合成が可能である．入力変数の数を n，二分決定グラフの最大幅を w とすると，合成される組合せ回路（ファンイン制限あり）の段数，素子数はそれぞれ $O(\log n \log w)$、$O(nw^3)$ である．合成される回路は二線入力二線出力の論理回路であるが、この回路はそのまま用いれば、パス遅延故障ロバストテスト容易であり，かつ単一縮退故障に対してトータリセルフチェッキングであるという，テスト容易化設計の立場からも好ましい性質を持つ．

Synthesis of Combinational Logic Circuits from Binary Decision Diagrams

Nagisa ISHIURA
Department of Information Systems Engineering
Faculty of Engineering, Osaka Univeristy

We propose in this paper a new method of synthesizing combinational logic circuits directly from binary decision diagrams. Since the synthesis process in this method does not go by way of two-level (sum-of-product) representations, we can synthesize combinational circuits for parity functions and symmetric functions within practical time, which was impossible in the conventional methods because the size of the two-level representations of these functions becomes exponential of the number of input variables. Let n be the number of input variables of Boolean function f, and w be the *maximum width* of the binary decision diagram that represents f. Then the depth and the size of the circuit synthesized by our method are $O(\log n \log w)$ and $O(nw^3)$, respectively. The synthesized circuit is a 2-rail-input 2-rail-output circuit. This circuit has good properties from the standpoint of testability; the circuit is *robustly path-delay fault testable* and also *totally self-checking* for single stuck-at faults.

1 はじめに

論理合成は現在 VLSI の計算機援用設計におけるもっとも重要な技術の一つであり, 多くの研究がなされている [Cho89, Dar81, Dev90, Mat27, Mur89]. 本論文の主題である多段論理合成 [Fuj91] は, 何らかの形で与えられた論理関数の仕様記述から多段の組合せ論理回路を合成するものである. 現在, 多段論理合成手法の主流は, 論理関数の仕様記述から一旦二段論理回路 (積和形論理回路) を導出し, これを多段化した後に最適化を行なうというものである. この方法では, 回路の局所変換による方法 [Dar81] に比べ, 仕様の与え方に左右されにくい大域的な冗長性の除去が行なえるため, 合成される回路の品質が高く, 積和形表現が大きくならない回路では, ほぼ実用になる回路が合成できる. しかし, 実用上重要な論理関数の中にも積和形表現が入力変数の数の指数に比例して大きくなるものが存在する. パリティ関数, 対称関数, および算術演算に現れる関数などがその例であるが, このような関数に対しては積和表現を経由しない合成法を考える必要がある. 本稿では, この問題に対する一つの解決策として, 論理関数の二分決定グラフ (BDD) 表現 [Bry86] から組合せ論理回路を合成する一手法を提案する.

論理関数の BDD 表現は有向グラフを用いる表現法で, 多くの実用的な論理関数が入力変数の多項式に比例する大きさで表現できる [Ish90]. 例えば, n 入力パリティ関数を表現する BDD のサイズは $O(n)$ であり, 一般に n 入力の対称関数は $O(n^2)$ のサイズの BDD で表現できる. その他, 算術演算に現れる関数の中にも BDD によれば表現が小さくなるものがあり, BDD から組合せ回路を合成する方法は, 積和形表現を経由する方法の相補的技術として今後重要になると考えられる.

BDD 表現から組合せ論理回路を合成する最も簡単な方法は, BDD の各節点にセレクタを割り付ける方法である. この方法を用いれば, 容易に組合せ回路が合成でき, その素子数も, もとの BDD の節点数を N としたとき $O(N)$ で済む. しかし問題になるのは段数で, 入力変数の数を n としたとき, 合成される回路の段数は $O(n)$ 段になってしまう. むろん, これに対して多段最適化手法 [Mur89, Mat27, Cho89] を適用すれば段数は削減できるが, 必ずしも大域的な最適化になるとは限らない. これに対し, 本稿で提案する手法は BDD から直接段数の小さな回路を合成することを目ざすものである. 本手法では, BDD の表現する論理関数はブール行列の乗算の繰り返しによって計算できることに着目し, これを計算する樹状の組合せ回路を構成するというものである. 入力変数の数を n, 二分決定グラフの最大幅を w とすると, 本手法により合成される組合せ回路 (ファンイン制限あり) の段数, 素子数はそれぞれ $O(\log n \log w)$, $O(nw^3)$ である. セレクタを用いた方法に比べ素子数は大きくなるが, w が小さいときには段数が大幅に改善される.

さらに, 本手法により合成される回路はテスト容易性の観点からも優れた性質を持っている. 合成される回路は二線入力二線出力論理回路 (二線論理回路) である. 回路規模や入力線数の削減のために, これを一線入力一線出力の回路に作り変えることもできるが, このまま二線論理の回路として用いれば, 1) パス遅延故障ロバストテスト容易であり, かつ 2) 単一縮退故障に対してトータリセルフチェッキングである. とくに 1) のパス遅延故障ロバストテスト容易であることから, 回路中のすべてのパス遅延故障, CMOS 開放故障, 多重縮退故障がテスト可能であり, この回路は強力なテスト容易性を持つといえる.

以下では, まず 2 節で BDD と論理関数の記法を定義し, 3 節で BDD の表現する論理関数のブール行列乗算による計算法を述べる. 4 節では 3 節の計算法に基づく組合せ回路の合成法について述べ, 5 節でそのテスト容易性を議論する.

2 二分決定グラフとその準既約形

2.1 二分決定グラフ (BDD)

$\mathcal{B} = \{0,1\}$ 上の二分決定グラフ (BDD) [Bry86] は次のように定義される.

Def 1 $\mathcal{B}$ 上の二分決定グラフは 6 つ組 $B = (X, N_V, N_C, I, e^0, e^1, l)$ である. ただし,

$X = \{0, 1, \cdots, n-1\}$ は変数のインデクスの集合.
N_V は**変数節点の集合**.
$N_C = \{c_0, c_1\}$ は**定数節点の集合**.
$I = \{i_1, i_2, \cdots, i_m\} \subset (N_V \cup N_C)$ は初期節点の集合.
$e^0, e^1 : N_V \to (N_V \cup N_C)$ はそれぞれ節点から出る 0 枝と 1 枝を表す.
$l : (N_V \cup N_C) \to (X \cup \{n\})$ は節点のレベルを表し, 次を満たす.

$$l(v) = n \text{ iff } v \in N_C,$$
$$l(u) < l(e^0(u)),\ l(u) < l(e^1(u)). \qquad \square$$

c_0, c_1 はそれぞれ **0 節点**, **1 節点**と呼ばれる. 節点の対 $(v, e^0(v))$, $(v, e^1(v))$ はそれぞれ節点 v の **0 枝**, **1 枝** と呼ばれる. **0 枝**と **1 枝**は合わせて**枝**と呼ばれる. BDD B のサイズは, B の節点数と定義され, size(B) と表記される. すなわち, $|S|$ を集合 S の要素数とすると, size(B) $= |N_V \cup N_C|$ である. 上記の定義ではどの節点からも到達できない節点が初期節点以外に存在するものも BDD となるが, このようなものは意味がないので, 本稿では除外して考える. すなわち, 節点 v は, $v \in I$ または $\exists u[e^0(u) = v \vee e^1(u) = v]$ を満たすものと仮定する.

BDD の各節点 v は, 次に定義される論理関数 $f_v : \mathcal{B}^n \to \mathcal{B}$ を表現する.

$$f_{c_0} = 0 \text{ (恒偽関数)} \qquad f_{c_1} = 1 \text{ (恒真関数)}$$
$$f_v = \overline{x_{l(v)}} \cdot f_{e^0(v)} + x_{l(v)} \cdot f_{e^1(v)} \text{ (if } v \in N_V)$$

BDD の初期節点集合の表現する関数 $\{f_{i_1}, f_{i_2}, \cdots, f_{i_m}\}$ を, その BDD が表現する関数という.

2.2 BDD の既約形と準既約形

変数節点 v が $e^0(v) = e^1(v)$ を満たすとき, v は冗長であるという. また, 二つの変数節点 v_1 と v_2 が $e^0(v_1) = e^0(v_2)$ かつ $e^1(v_1) = e^1(v_2)$ を満たすとき, v_1

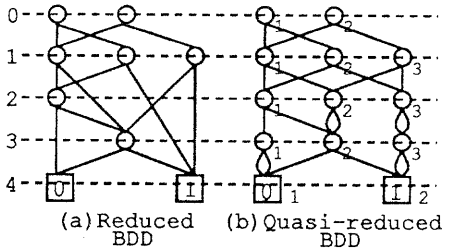

図1. 既約な BDD と準既約な BDD.

と v_2 は等価であるという. 冗長な節点と等価な節点は, BDD が表現する論理関数を変更することなく除去することができる. この操作を**既約化** (reduction) と呼ぶ. 与えられた変数の全順序の下で, ある論理関数 f を表す BDD は複数存在するが, これらに既約化を行なって得られるものは, 唯一であることが知られている [Bry86].

BDD のうち, すべての変数節点 v が $l(e^0(v)) = l(e^1(v)) = l(v)+1$ を満たすものを**密な BDD** と呼ぶことにする. 任意の BDD は, 冗長な節点を追加することにより, 密な BDD に変形することができる. 密な BDD から, 等価な節点の削除を行なって得られるものを**準既約な BDD** と定義する. 図1 (a) は既約な BDD であり, (b) は同じ関数を表現する準既約な BDD である.

Prop 1 準既約な BDD は次の性質を持つ.

(1) 同じ論理関数を表す準既約な BDD は唯一である.
(2) 準既約な BDD のサイズは同じ関数を表す既約な BDD のサイズの高々 n 倍である. □

本稿では, $f_{i_1}, f_{i_2}, \cdots, f_{i_m}$ のいずれもが依存しない入力変数は存在しないと仮定する. すなわち, 冗長な節点のみからなるレベルは存在しないと仮定する. すると, 次の補題が成立する.

Lem 2 準既約な BDD においては, $0 < j$ なる各 j について, $l(v) = j$ を満たす節点は 2 つ以上存在する.

[証明] 帰納法による.

(1) $j = 1$ のとき: $l(v) = 1$ である節点が v だけであったと仮定する. 初期節点が 1 つ (i とする) の場合, この BDD が準既約であることから, $e^0(i) = e^1(i) = v$ である. これは, レベル 0 の唯一の節点である i が冗長であることを意味し, どの関数にも寄与しない変数は存在しないという仮定に反する. 初期節点が 2 つ以上ある場合, これを $i_1, i_2, \cdots, i_m$ とすると, $k = 1, 2, \cdots, m$ に対して $e^0(i_k) = e^1(i_k) = v$ となり, これらの初期節点は全て等価となる. これは, この BDD が準既約であることに反する.

(2) $j > 1$ のとき: 帰納法の仮定よりレベル $j-1$ の節点は 2 つ以上ある. すると, (1) の初期節点が 2 つ以上ある場合と同様の議論により, レベル j の節点は 2 つ以上ある. □

2.3 節点間の到達可能性

$a = (a_0, a_1, \cdots, a_{n-1}) \in \mathcal{B}^n$ $(n = |X|)$ なるベクトルを, X に対する**割当て**と呼ぶ. また, その要素 a_i を変数 x_i への割当てと呼ぶ.

Def 2 割当て a のもとで, 節点 u から v に到達可能であるとは次のように定義され, $u \xrightarrow{a} v$ と表記される.

1) $u \xrightarrow{a} u$.
2) $e^b(u) = v \wedge b = a_{l(u)}$ のとき $u \xrightarrow{a} v$.
3) $u \xrightarrow{a} w$ かつ $w \xrightarrow{a} v$ のとき $u \xrightarrow{a} v$. □

なお, 簡単のため $u \xrightarrow{a} v$ かつ $v \xrightarrow{a} w$ のとき, これを $u \xrightarrow{a} v \xrightarrow{a} w$ と表記することにする. BDD の表現する論理関数は, 次に示されるとおり, 定数節点への到達可能性と密接な関係を持っている.

Prop 3 $v \in (N_V \cup N_C)$ に対し, 次が成立する.

$$f_v(a) = 1 \quad \text{iff} \quad v \xrightarrow{a} c_1,$$
$$f_v(a) = 0 \quad \text{iff} \quad v \xrightarrow{a} c_0.$$

3 レベル間の到達可能性行列

本稿では, 初期節点から 1 節点への到達可能性を, ブール行列とその乗算を用いて定式化することにより, 組合せ回路の合成を考える.

Def 3 B を準既約な BDD とする. B のレベル l, k の節点数をそれぞれ m_l, m_k とし, レベル l の節点を $\{v_l^1, v_l^2, \cdots, v_l^{m_l}\}$, レベル k の節点を $\{v_k^1, v_k^2, \cdots, v_k^{m_k}\}$ と表記することにする. B の l から k へのレベル間到達可能性行列 $R_{l,k}$ は $m_l \times m_k$ 行列であり, その (i,j) 成分 $r^{i,j} : \mathcal{B}^n \to B$ は次のように定義される.

$$r^{i,j}(a) = 1 \text{ iff } v_l^i \xrightarrow{a} v_k^j. \qquad \square$$

すなわち, $R_{l,k}$ の (i,j) 成分は, レベル l の i 番目の節点からレベル k の j 番目の節点へ到達できる割当てに対して 1 となる論理関数である. 例えば, 図1 (b) の BDD において,

$$R_{0,1} = \begin{bmatrix} \overline{x_0} & x_0 & 0 \\ \overline{x_0} & 0 & x_0 \end{bmatrix}, R_{1,2} = \begin{bmatrix} \overline{x_1} & x_1 & 0 \\ \overline{x_1} & 0 & x_1 \\ 0 & \overline{x_1} & x_1 \end{bmatrix},$$

である. 一般に, 隣接するレベル間の到達可能性を表す $R_{l,l+1}$ の要素 $r^{i,j}$ は, 変数 x_l のみに依存し, 次のようになる.

$$r^{i,j} = \begin{cases} \overline{x_l} & (e^0(v_l^i) = v_{l+1}^j \wedge e^1(v_l^i) \neq v_{l+1}^j) \\ x_l & (e^0(v_l^i) \neq v_{l+1}^j \wedge e^1(v_l^i) = v_{l+1}^j) \\ 0 & (e^0(v_l^i) \neq v_{l+1}^j \wedge e^1(v_l^i) \neq v_{l+1}^j) \\ 1 & (e^0(v_l^i) = v_{l+1}^j \wedge e^1(v_l^i) = v_{l+1}^j) \end{cases} \quad (1)$$

また, $R_{l,h}$ と $R_{h,k}$ が既知の時, $R_{l,k}$ は,

$$R_{l,k} = R_{l,h} \cdot R_{h,k} \quad (2)$$

により計算できる. ここに $\cdot$ はブール行列の乗算を表す. 例えば,

$$\begin{aligned} R_{0,2} &= R_{0,1} \cdot R_{1,2} \\ &= \begin{bmatrix} \overline{x_0}\overline{x_1} + x_0\overline{x_1} & \overline{x_0}x_1 & x_0x_1 \\ \overline{x_0}\overline{x_1} & \overline{x_0}x_1 + x_0\overline{x_1} & x_0x_1 \end{bmatrix}, \end{aligned}$$

である.

$R_{0,n}$ は, 初期節点から定数節点への到達可能性を表す行列であり, 各要素は BDD が表す論理関数とその

否定からなる．すなわち，$f_{i_1}, f_{i_2}, \cdots, f_{i_m}$ を B が表す論理関数とすると，

$$R_{0,n} = \begin{bmatrix} \overline{f_{i_1}} & f_{i_1} \\ \overline{f_{i_2}} & f_{i_2} \\ \cdots & \cdots \\ \overline{f_{i_m}} & f_{i_m} \end{bmatrix}.$$

となる．この行列は，各隣接レベル間の到達可能性行列の乗算により求めることができる．

$$R_{0,n} = R_{0,1} \cdot R_{1,2} \cdot \cdots \cdot R_{n-1,n}. \quad (3)$$

本稿では，上式を計算する組合せ回路を構成することにより，BDD で表現される論理関数を実現する回路を合成する．この際，行列の乗算が任意の順序で評価できる (結合律を満たす) ことに注目し，これを二分木状に行なうことによって段数の削減を図る．

4 組合せ回路の合成

4.1 ブール行列の乗算の実現

$P = [p^{i,j}]$, $Q = [q^{i,j}]$ をそれぞれ $p \times r$, $r \times q$ のブール関数行列とする．すると，P と Q の乗算結果 $R = [r^{i,j}]$ は $p \times q$ 行列であり，その (i,j) 成分は次のようになる．

$$\begin{aligned} r^{i,j} &= p^{i,1} \cdot q^{1,j} + p^{i,2} \cdot q^{2,j} + \cdots + p^{i,r} \cdot q^{r,j} \\ &= \textstyle\sum_{k=1}^{r} p^{i,k} \cdot q^{k,j}. \end{aligned}$$

ここで，$\cdot$ は論理積 (and)，$+$ は論理和 (or) を表す．すなわち，ブール関数行列の乗算は and-or 二段の組合せ回路で実現できる．乗算の結果得られる行列の各成分は最大 r 個の積項を持つ．また，各積項のリテラル数は高々 2 である．従って，$p \times r$ 行列と $r \times q$ 行列の乗算を二段組合せ回路で実現した場合，その積項の総数は pqr 個以下である．また，ファンイン制限 2 の and, or ゲートでこの組合せ回路を実現した場合の段数，素子数はそれぞれ，$1 + \lceil \log r \rceil$, $2pqr$ 以下である．

4.2 BDD の表現するブール関数の実現

$R_{l,l+1}$ の各成分は式 (1) のとおり，$0, 1, \overline{x_l}, x_l$ のいずれかであるから，回路としては結線だけで実現できる．ブール行列の乗算は上述のとおり and-or 二段の組合せ回路で実現できるので，式 (2) に従って組合せ回路を構成すれば BDD の表現するブール関数が実現できる．ただし，この際次の制限を設ける．

制限 1 式 (2) の乗算は樹状に結合する．例えば，$n = 4$ の場合には，$R_{0,4} = ((R_{0,1} \cdot R_{1,2}) \cdot (R_{2,3} \cdot R_{3,4}))$ という結合に基づいて組合せ回路を合成する．

制限 2 ブール行列乗算の組合せ回路を構成する際に，次の 4 つの規則に対応する回路の簡単化を行なう．

1) $a \cdot 1 \to a$.　2) $a \cdot 0 \to 0$.
3) $a + 1 \to 1$.　4) $a + 0 \to a$.

さらに，2), 3) の簡単化を行なった際に a を計算する回路のうち不要な部分は削除する．

制限 1 は回路の段数を削減するために重要である．また，制限 2 は次節で述べる回路のテスト容易性ために必要である．図 2 は n=4 の場合の回路のブロック図である．ただし，出力 f_k は f_{i_k} を表す．この回路の，段数，素子数は次の定理のとおりである．

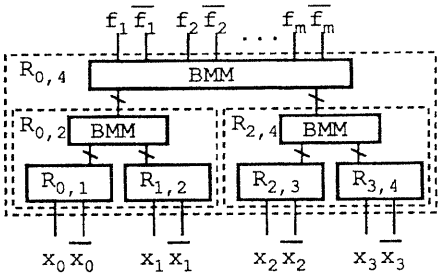

図 2. 回路のブロック図

Th 4 与えられた BDD B の入力変数の数を n, 最大幅を w とする．B から合成される回路の段数 (定数ファンイン)，素子数はそれぞれ，$O(\log n \log w)$, $O(nw^3)$ である．□

5 合成された回路のテスト容易性

合成される回路は図 2 に示したとおり二線入力二線出力 (二線論理) の組合せ回路である．回路規模や入力線数の削減のために，これを一線入力一線出力の回路に作り変えることもできるが，このまま二線論理の回路として用いることができれば，次に示すような優れたテスト容易性を持つ．

(1) 合成された二線論理回路は，パス遅延故障ロバストテスト容易 [Dev90] である．

(2) 合成された二線論理回路は，単一縮退故障に対してトータリセルフチェッキング [Nan91] である．

(1) は，この回路中のすべてのパス遅延故障，CMOS 開放故障，多重縮退故障がテスト可能であるという，強力な結果である．また，(2) は，回路がオンラインのテストにも有効であることを示している．これらの定理は，準既約な BDD の性質と，合成された二線論理回路が単調であることに依っている．

以下での議論で，元の論理関数の入力変数に対する割当と，二線論理回路に対する入力ベクトルの対応をとるため，写像 $d : \mathcal{B}^n \to \mathcal{B}^{2n}$ を以下のように定義する．ただし，回路の入力ベクトルは先頭から順に $x_0, \overline{x_0}, x_1, \overline{x_1}, \cdots, x_{n-1}, \overline{x_{n-1}}$ に対応するものとする．

$$\begin{aligned} d(a) &= d_0 d_2 \cdots d_{n-1}, \\ d_i &= \begin{cases} 01 \text{ (if } a_i = 0) \\ 10 \text{ (if } a_i = 1) \end{cases}. \end{aligned}$$

5.1 パス遅延故障のロバストテスト容易性

組合せ回路のパスの遅延が大きくなる故障をパス遅延故障という [Lin87]．パス遅延故障のテストは通常 2 ベクトル (p_1, p_2) からなる．入力変化 $p_1 \to p_2$ に対する信号値変化が規定時間までにパスに沿って伝播し出力に現れるかどうかを検査する．この際に，ハザードが発生すると，この検査が困難になる．回路内の遅延の分布や他のパスの遅延故障の存在によらず故障検査ができる時，このテストをロバストテストといい，回路中の

全てのパス遅延故障に対してロバストテストが存在するとき，この回路を**パス遅延故障ロバストテスト容易** (robustly path-delay fault testable) という [Dev90].

本稿の手法によって合成された回路がパス遅延故障ロバストテスト容易であることを証明するために，まず，BDD の枝とパスの活性化を定義する．

Def 4 BDD の枝 $e=(u,v)$ が次を満たすとき，e は割当 a のもとで**活性化されている**，あるいは a は e を**活性化する**という．

$$(e^0(u)=v \wedge a_{l(u)}=0) \vee (e^1(u)=v \wedge a_{l(u)}=1) \quad \square$$

Def 5 BDD の枝の系列 $(v_1,v_2)(v_2,v_3)\cdots(v_{k-1},v_k)$ を BDD の**パス**と呼ぶ． $\square$

Def 6 パス p を構成する全ての枝が割当 a のもとで活性化されているとき，p は割当 a のもとで活性化されている，あるいは a は p を活性化するという． $\square$

パスの活性化に関しては，次の補題が成立する．

Lem 5 節点 v から u に至るパスのうち，一つの割当によって活性化されるパスは唯一である．また，節点 v から u に至るパスが存在するなら，そのパスを活性化する割当が存在する．

[証明] v から u に至るパス p は，適当な定数 $b_0,b_1,b_2,\cdots,b_{k-1}$ (ただし $b_i\in\{0,1\}$) を用いて次のように書くことができる．

$$\begin{aligned} p &= (v_0,v_1)(v_1,v_2)(v_2,v_3)\cdots(v_{k-1},v_k),\\ &\quad v_0=v, v_k=u,\\ v_{i+1} &= e^{b_i}(v_i)\ (i=0,1,2,\cdots,k-1), \end{aligned}$$

$l_v=l(v), l_u=l(u)$ とする．割当 a が与えられたとき，v から u に至るパスで活性化されるものは，上式において $b_0=a_{l_v}, b_1=a_{l_v+1}, b_2=a_{l_v+2},\cdots,b_{k-1}=a_{l_u-1}$ としたものであり，唯一である．逆に，u から v に至る任意のパス p が与えられたとき，これに対する b_i を求め，$a_{l_v}=b_0, a_{l_v+1}=b_1, =a_{l_v+2}=b_2,\cdots,a_{l_u-1}=b_{k-1}$ とすれば，p を活性化することができる． $\square$

Cor 6 初期節点 i から定数節点へ至るパスのうち，一つの割当によって活性化されるものは唯一である．また，i から定数節点へ至る任意のパス p に対して，p を活性化する割当が存在する． $\square$

BDD 中のある節点 u から ある節点 v へ到達できる条件を $r_{u,v}$ と表記することにする．$r_{u,v}:B^n\to B$ であり，$r_{u,v}(a)=1$ iff $u\xrightarrow{a}v$ である．さて，ここで合成された回路を考える．BMM 回路の各出力信号線は，節点 u から節点 v への到達可能条件 $r_{u,v}$ を計算するものである．$r_{u,v}$ を与える信号線を $s_{u,v}$ と表記することにする．and ゲートと or ゲートの 出力信号線が y，入力信号線が $a_1,a_2,\cdots,a_m$ であることを，それぞれ $y\leftarrow and(a_1,a_2,\cdots,a_m), y\leftarrow or(a_1,a_2,\cdots,a_m)$ で表すことにすると，ブール行列の乗算の定義より，$s_{u,v}$ は次のような部分回路によって計算されている．

$$\begin{aligned} s_{u,v} &\leftarrow or(s^1_{u,v},s^2_{u,v},\cdots,s^k_{u,v})\\ s^i_{u,v} &\leftarrow \begin{cases} s_{u,w_i} & (r_{w_i,v}\text{ が恒真})\\ s_{w_i,v} & (r_{w_i,v}\text{ が恒真})\\ and(s_{u,w_i},s_{w_i,v}) & (\text{それ以外}) \end{cases} \end{aligned}$$

ここで，w_i はレベル l_w $(l(u)<l_w<l(v))$ の節点である．$s^i_{u,v}$ は，u から w_i を経由して v に到達できる条件を与えるものである．この部分回路に対して，次の補題が成立する．

Lem 7 ある入力ベクトル $d(a)$ のもとで，$s_{v,w}$ の値が 1 のとき，or ゲートの入力 $s^1_{u,v},s^2_{u,v},\cdots,s^k_{u,v}$ のうち値が 1 のものは唯一であり，残りの値は全て 0 となる．逆に，任意の $s^i_{u,v}$ だけを 1 に，残りを全て 0 にする入力ベクトルが存在する．

[証明] 入力ベクトル $d(a)$ のもとで $s^i_{u,v}$ の値が 1 であることは，$r_{u,w_i},r_{w_i,v}$ がともに 1 であり，v から w_i を経由して u へ到達できる，すなわち，$v\to w_i\to u$ なるパスが活性化されていることと同値である．今仮に，入力ベクトル $d(a)$ のもとで $s_{v,w}$ の値が 1 のとき，$s^1_{u,v},s^2_{u,v},\cdots,s^k_{u,v}$ のうち値が 1 のものが 2 つあったとする．一般性を失うことなく，これを $s^1_{u,v},s^2_{u,v}$ とする．これらの値が 1 であることは，$v\to w_1\to u$ なるパスと $v\to w_2\to u$ なるパスが同時に活性化されていることを意味し，Lem 5 に反する．逆に，Lem 5 より v から任意の w_i を経て u に至るパスを活性化する割当 a が存在するので入力ベクトルとして $d(a)$ を用いれば，$s^i_{u,v}$ (のみ) を 1 にすることができる． $\square$

この補題から，次が成立する．

Lem 8 ある入力ベクトル $d(a)$ に対してある外部出力 s の値が 1 のとき，外部入力から s に至るクリティカルパス が存在し，その集合は，and ゲートを節点とする二分木 $C^s_{d(a)}$ を構成する．

[証明] ある BMM 回路の出力 $s_{u,v}$ が入力ベクトル $d(a)$ のもとで値が 1 であり，かつクリティカルであったとする．Lem 7 より or ゲートの入力のうち 1 であるもの ($s^i_{u,v}$ とする) は唯一であるから，$s^i_{u,v}$ はクリティカルである．また，$s^i_{u,v}$ に接続している信号線もクリティカルであるから，$r_{w_i,v}$ が恒真の場合には s_{u,w_i} が，$r_{w_i,v}$ が恒真の場合には $s_{w_i,v}$ が，それ以外の場合には and ゲートの入力である $s_{u,w_i},s_{w_i,v}$ がともにクリティカルとなる．値が 1 である外部出力 s より始めて外部入力に向かってBMM 回路の中をたどっていくと，上の議論より，クリティカルパスは and ゲートを節とする二分グラフを構成し，終端は外部入力に到達する．このグラフが再収れんを持たない二分木であることは次のようにして示せる．クリティカルパスの (出力からたどった場合の) 分岐点 となる and ゲートの入力を $s_{u,w_i},s_{w_i,v}$ とすると，これらの信号線に到達できる外部入力の入力変数の集合はそれぞれ $\{x_{l(u)},\cdots,x_{l(w_i)-1}\},\{x_{l(w_i)},\cdots,x_{l(v)-1}\}$ であり，共通部分を持たない．従って，このグラフは再収れんを持たない． $\square$

p を外部入力 i から外部出力 s に至る再収れんのないクリティカルパスとする．i に $0\to1, 1\to0$ 変化を加えれば，それに対する信号値変化が s で観測できる．p には再収れんがないので，他のパスの遅延の影響を受けず，ハザードも発生しない．従って，p の遅延故障にはロバストテストが存在する．前述の補題の $C^s_{d(a)}$ に含まれるパスは全て再収れんのないクリティカルパ

スであり, 従ってその遅延故障にはロバストテストが存在する.

もし, 外部入力から外部出力に至る任意のパスが再収れんなく活性化できれば, この回路はパス遅延故障ロバストテスト容易といえる. この条件は, 次の補題により保証される.

Lem 9 外部入力から外部出力 s に至る任意のパス p に対し, $p \in C^s_{d(a)}$ なる活性化回路の二分木 $C^s_{d(a)}$ を与える入力ベクトル $d(a)$ が存在する.

[証明] BMM 回路の出力信号線および $R_{l,l+1}$ を計算する回路の出力信号線のうち, p に含まれるものを $t_1, t_2, \cdots, t_k$ $(k \leq \lceil \log n \rceil)$ とする. 各 t_j は BDD 中のある節点 u_j からある節点 v_j への到達可能性に対応している. $T = \{u_1, v_1\} \cup \{u_2, v_2\} \cup \cdots \cup \{u_k, v_k\}$ とすると, T 中には同じレベルの節点は一つずつしかない. q を T の全ての節点を含む BDD のパスとし, a を q を活性化する割当とする. a の存在は Cor 6 により保証される. a のもとでは全ての j について u_j から v_j に到達できるので, $d(a)$ を印加すれば $t_1, t_2, \cdots, t_k$ は 1 となる. Lem 8 の証明より, これらを含むパス p はクリティカルである. □

Th 10 合成された二線論理回路は, パス遅延故障ロバストテスト容易である. □

5.2 トータリセルフチェッキング性

合成された二線論理回路は $2n$ 入力 $2m$ 出力であるが, 正常動作時には $D = \{01, 10\}^n$ の要素を入力として受け付け, $R = \{01, 10\}^m$ の要素を出力する. 以下では, D および R の要素を**符号語**, $B^{2n} - D$ および $B^{2m} - R$ の要素を**非符号語** と呼ぶ. 回路のトータリセルフチェッキング性は次のように定義される [Nan91].

Def 7 回路 C が, 故障集合 F に属する任意の故障が存在しても, 符号語入力に対しては決して誤った符号語を出力しないならば, C は F に関してフォールトセキュア (fault secure) であるという. また, C が F に属する任意の故障が存在するときには, 少なくとも一つの符号語入力に対して非符号語を出力するならば, C は F に関してセルフテスティング (self testing) であるという. C が Fに関してフォールトセキュアかつセルフテスティングであるとき, C は Fに関してトータリセルフチェッキング (totallyself-checking) であるという. □

Lem 11 合成された回路は単一縮退故障に関してフォールトセキュアである.

[証明] $R_{l,l+1}$ を計算する回路は結線のみ, BMM 回路は and ゲートと or ゲートのみから構成されているので, 合成された回路は単調である. 従って, 回路中に単一縮退故障が発生して複数の出力にその影響が伝播した場合でも, 信号の誤り方は一方向であり, $0 \to 1$, $1 \to 0$ の両方の誤り方を同時に含むことはない. 従って, この回路は決して不正な符号語を出力しない. □

合成された回路はパス遅延故障テスト容易であるから, 全ての単一縮退故障に対してこれを検出するテストが存在する. しかし, このテストの中には x_i と $\overline{x_i}$ の両方ともが 0 となる, すなわち符号語でないものが含まれており, これだけではセルフテスティングとは言えない. 合成された回路のセルフテスティング性は, 次の証明による.

Lem 12 合成された回路は単一縮退故障に関してセルフテスティングである.

[証明] 回路で発生する各単一縮退故障に対し, これを検出する符号語が存在することを示す. まず BMM 回路で発生する故障について考える. Lem 7 の記法

$$
\begin{aligned}
s_{u,v} &\leftarrow or(s^1_{u,v}, s^2_{u,v}, \cdots, s^k_{u,v}) \\
s^i_{u,v} &\leftarrow \begin{cases} s_{u,w_i} & (r_{w_i,v} \text{ が恒真}) \\ s_{w_i,v} & (r_{w_i,v} \text{ が恒真}) \\ and(s_{u,w_i}, s_{w_i,v}) & (\text{それ以外}) \end{cases}
\end{aligned}
$$

に従えば, $s_{u,v}, s^i_{u,v}, s_{u,w_i}, s_{w_i,v}$ それぞれの s-a-0, s-a-1 故障が考えられる. このうち, 故障の等価性, 回路の対称性を考慮すると, 1) $s_{u,v}$ の s-a-0 故障, 2) $s_{u,v}$ の s-a-1 故障, 3) $s^i_{u,v}$ の s-a-0 故障, 4) s_{u,w_i} の s-a-1 故障を考えれば良いことになる.

1) $s_{u,v}$ の s-a-0 故障:

この故障の存在は $s_{u,v}$ を常に 0 にするので, 元の BDD 上で考えれば, 節点 u から v に到達できなくなることに対応する. u に到達できる初期節点 i と適当な定数節点 c_b $(b \in B)$ を選び, $i \xrightarrow{a} u \xrightarrow{a} v \xrightarrow{a} c_b$ を満たす割当 a を求める. a の存在は Lem 6 より保証される. 故障がない場合には, a のもとで i から c_b に到達でき $c_{\overline{b}}$ には到達できない. これに対して故障の影響によって u から v に到達できない場合には, i から c_b にも $c_{\overline{b}}$ にも到達することはできなくなってしまう. これは, 回路に $d(a)$ を印加したときに, $f_i, \overline{f_i}$ に対応する出力がいずれも 0 となることに対応する. 従って, $d(a)$ (符号語である) はこの故障のテストとなる.

2) $s_{u,v}$ の s-a-1 故障:

BDD 上で考えれば, 節点 u から v に常に到達可能であることを対応する. u に到達できる初期節点 i および v と同じレベルの別の節点 v' を選ぶ. v' の存在は Lem 2 により保証される. $v \neq v'$ であるから, $f_v \neq f_{v'}$. 従って, 入力変数 $x_{l(v)}, x_{l(v)+1}, \cdots x_{n-1}$ に適当な値を割り当てれば, v からは $c_{\overline{b}}$ に, v' からは c_b に到達可能とすることができる. これを利用し, 割当 a として, $i \xrightarrow{a} u \xrightarrow{v} v' \xrightarrow{a} c_b$ かつ $v \xrightarrow{a} c_{\overline{b}}$ なるものを選ぶ. 故障がない場合には, a のもとで i から c_b に到達でき $c_{\overline{b}}$ には到達できない. これに対して故障の影響がある場合には, u から v にも到達でき, 従って, i から c_b にも $c_{\overline{b}}$ にも到達できてしまう. これは, 回路に $d(a)$ を印加したときに, $f_i, \overline{f_i}$ に対応する出力がいずれも 1 となることに対応する. 従って, $d(a)$ はこの故障のテストとなる.

3) $s^i_{u,v}$ の s-a-0 故障:

BDD 上で考えれば, 節点 u から w_i 経由では v に到達できないことを対応する. 従って, a として $i \xrightarrow{a} u \xrightarrow{a} w_i \xrightarrow{a} v \xrightarrow{a} c_b$ (i と c_b は適当なもの) を満たすものを選べば, 1) と同様の議論によって, $d(a)$ がこの故障のテストとなる.

4) s_{u,w_i} の s-a-1 故障:

BDD 上で考えれば, w_i から v に到達可能であるときには必ず u から w_i (従って u から v に) 到達可能であることを意味し, 2) と同様の議論によって, 符号語でかつこのテストとなるものが存在する.

次に, $R_{l,l+1}$ を与える回路の故障を考える. この回路は式 (1) に従って構成されており, 節点 u から v ($l(v)=l(u)+1$) への到達可能性を与える信号線を $s_{u,v}$, 変数 $x_i, \overline{x_i}$ に対応する信号線をそれぞれ $s_{x_i}, s_{\overline{x_i}}$ とすると,

$$s_{u,v} \leftarrow \begin{cases} s_{\overline{x_i}} & \text{if } e^0(u) = v \wedge e^1(u) \neq v \\ s_{x_i} & \text{if } e^0(u) \neq v \wedge e^1(u) = v \end{cases}$$

と書ける. $s_{u,v}$ の s-a-0 故障と s-a-1 故障は, それぞれ BMM 回路の 1), 2) の場合と同様である. $s_{x_i}, s_{\overline{x_i}}$ の s-a-0, s-a-1 故障に関しては, 対称性より 5) s_{x_i} の s-a-0 および 6) s_{x_i} の s-a-1 故障を考えればよい.

5) s_{x_i} の s-a-0 故障:

BDD 上では, レベル i のどの節点 u からも $e^1(u)$ に到達できないことを対応する. 従って, 割当として $i \xrightarrow{a} u \xrightarrow{a} e^1(u) \xrightarrow{a} c_b$ (i と c_b は適当な節点) を満たすものを選べば, この故障は検出できる.

6) s_{x_i} の s-a-1 故障:

BDD 上では, レベル i のどの節点 u からも $e^1(u)$ に常に到達できることを対応する. 従って, u として $e^0(u) \neq e^1(u)$ なる節点を選び, 割当として $i \xrightarrow{a} u \xrightarrow{a} e^0(u) \xrightarrow{a} c_b$ かつ $e^1(u) \xrightarrow{a} c_{\overline{b}}$ (i と c_b は適当な節点) を満たすものを選べば, この故障は検出できる.

以上より, 全ての故障が符号語で検出できることが証明された. □

Th 13 合成された二線論理回路は, 単一縮退故障に関して TSC である. □

6 むすび

BDD から直接組合せ回路を合成する手法を提案した. 本手法は, BDD の幅が小さい場合には有効であるが, そうでない場合には合成された回路規模が大き過ぎるという問題点があり, これを削減する手法を考えることが, 今後の課題である. また, 回路を二線論理にすれば, パス遅延故障ロバストテスト容易性, 単一故障に関するトータリセルフチェッキング性の実現はさほど困難ではなく, 重大な成果とはいえない. 従って, 一線論理でこのテスト容易性を実現できるように, 合成法を改良することも今後の課題である. さらに, don't care をどう扱うかに関しても検討が必要であると考える.

謝辞

本研究にあたり, 御助言, 御討論いただきました, 富士通研究所の藤田昌宏氏, NTT の湊真一氏, 京都大学の出口豊氏に感謝致します. また, 御討論頂いた京都大学矢島研究室および大阪大学白川研究室の皆様に感謝します.

参考文献

[Bry86] R. E. Bryant: "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691 (Aug. 1986).

[Cho89] H. Cho, G. Hatchtel, R. Jacoby and P. Moceyunas: "Test Patten Generation is Logic Optimization", *Proc. IEEE International Conference on Computer-Aided Design (ICCAD-89)*, (Nov. 1989).

[Dar81] J. Darringer, W. Joyner, L. Berman and L. Trevillyan: "Logic Synthesis through Local Transformation", *IBM Journal of Research and Development*, vol. 24, no. 4, pp. 272-280 (Jul. 1981).

[Dev90] S. Devadas and K. Keutzer: "Synthesis and Optimization Procedure for Robustly Delay-Fault Testable Combinational Logic Circuits", *Proc. ACM/IEEE 27th Design Automation Conference*, pp. 221–227 (Jun. 1990).

[Fuj91] 藤田昌宏: 多段論理合成技術の動向, 電子情報通信学会論文誌 A, vol. J74-A, no. 2, pp. 152–161 (Feb. 1991).

[Ish90] N. Ishiura and S. Yajima: "A Class of Logic Functions Expressible by Polynomial-Size Binary Decision Diagrams", *Proc. Synthesis and Simulation Meeting and International Interchange (SASIMI 90)*, pp. 48–54 (Oct. 1990).

[Lin87] C. J. Lin and S. M. Reddy: "On Delay Fault Testing in Logic Circuits", *Proc. IEEE International Conference on Computer-Aided Design (ICCAD-87)*, pp. 694–703 (Nov. 1987).

[Mat27] 松永裕介, 藤田昌宏: 順序付き 2 分決定グラフと許容関数を用いた多段論理回路簡単化手法, 電子情報通信学会論文誌 A, vol. J74-A, no. 2, pp. 196–205 (Feb. 1991).

[Mur89] S. Muroga, Y. Kambayashi, H. C. Lai and J. N. Culliney: "The Transduction Method – Design of Logic Networks based on Permissible Functions", *IEEE Trans. Computers*, vol. C38, vol. 10, pp. 1404–1424 (Oct. 1989).

[Nan91] 南谷崇: フォールトトレラントコンピュータ, 電子情報通信学会 (編), コンピュータアーキテクチャシリーズ, オーム社 (Feb. 1991).