システムＬＳＩ設計技術　96－8
（2000. 5. 12）

誤り追跡入力と論理関数処理を併用したFPGA対応論理診断手法

井上 宏，皆見 利行，水谷 浩二，黒木 修隆，沼 昌宏

神戸大学

〒657-8501 神戸市灘区六甲台町 1-1

E-mail: {hiroshi, kaimi, kuroki, numa}@cas.eedept.kobe-u.ac.jp

あらまし　本稿では，LUT型FPGAで実現された回路に含まれる多重設計誤りの修正を行うEXL法を提案する。本手法では，誤り追跡入力と呼ばれるパターンに基づく処理と，BDDに基づく論理関数処理を組み合わせている。パターンに基づく処理では，従来のゲート回路を対象とする拡張 X-伝搬法と同様に，考慮すべき組合せ箇所数を大幅に削減することを目的とする。一方，論理関数処理においては，残存する各組合せ箇所について，LUT 内部の機能を正しく決定することを目的とする。これら2種類の手法を併用することにより，現実的な処理時間で，100 %の限定率が達成された。

キーワード　論理診断，設計誤りの修正，FPGA，LUT，誤り追跡入力

An Error Diagnosis Technique for LUT-Based FPGA Designs Combining Pattern-Based Technique and BDD-Based Formal Technique

Hiroshi Inoue, Toshiyuki Kaimi, Koji Mizutani,

Nobutaka Kuroki and Masahiro Numa

Kobe University

1-1, Rokkodai, Nada, Kobe, Hyogo 657-8501, Japan

E-mail: {hiroshi, kaimi, kuroki, numa}@cas.eedept.kobe-u.ac.jp

Abstract　This paper presents an approach to rectify multiple logic design errors in LUT-based FPGA designs. This approach, called EXL-algorithm, combines two kinds of techniques: a pattern-based technique for locating errors like in the EXM-algorithm, and a BDD-based formal technique to fix LUT function at each location. This combination allows the EXL-algorithm to rectify errors of LUT functions within practical processing time at 100 % hit ratio.

key words　Error diagnosis, Rectification of design errors, FPGA, LUT, PLEM

1. はじめに

論理回路の設計時間短縮のために，論理合成ツールを利用した設計が一般的となっている。しかし，合成された回路がタイミング等の仕様を満足しない場合，人手で回路に変更を加えることがある。変更にともなって回路中に設計誤りが混入する可能性があり，論理検証が不可欠となる。検証により設計誤りの存在が確認された際に，その修正の自動化を目的とするのが論理診断である。論理診断手法は設計誤りの修正に限らず，仕様変更（ECO: Engineering Change Order）に対応したインクリメンタル合成 [1], [2] にも応用できる。すなわち，以前の仕様に基づいて設計・合成された回路が誤りを含むとみなして論理診断手法を適用することにより，最小の修正で仕様変更に対応しようとする。著者らはこれまでに，ゲートレベルの回路に含まれる多重の論理設計誤りを自動的に診断する拡張 X-伝搬法 [3] を提案している。

一方で，内部の論理や配線を電気的に書き替えることが可能な LUT（Look-Up Table）型 FPGA（Field Programmable Gate Array）の利用が急速に広がっている。これまでに，FPGA を対象とした論理診断手法については，限られた手法 [4], [5] が提案されている。

Kukimoto ら [4] はブール関係に基づく手法を提案している。この手法は，LUT の内部機能のみの変更で設計誤りを修正しようとすることで，タイミング設計のやり直しを避ける点に特徴がある。その一方で，同一パス上に存在する複数の誤りには対応できない点に問題があった。

Pomeranz ら [5] は，複数の入力組合せに対する回路の出力値を理想出力値と比較することで，診断を試みる手法を提案している。しかし，各入力組合せに対して回路出力値を調べるため，診断の最終段階で設計誤りと虚報を区別するのが困難である。外部入力数 n に対して最悪の場合，2^n 個の組合せについて調べる必要があり，現実的ではない。

本稿では，ゲート回路を対象とした有力な論理診断手法である拡張 X-伝搬法をもとに，FPGA に対応する論理診断を行う EXL（Extended X-algorithm for LUT-based circuits）法を提案する。図 1 に示すように，EXL 法は次の処理からなる。

i) 誤り追跡入力を用いた組合せ箇所の抽出と絞り込み
ii) 論理関数処理に基づく LUT 機能修正

まず i) により，誤り追跡入力 [6] と呼ぶ入力ベクトルを用いた組合せ箇所の抽出と絞り込みを行う。修正方法を特定しない組合せ箇所の段階で，考慮すべき組合せ箇所数を大きく削減する。残存する各組合せ箇所について，該当する各 LUT の機能を ii) の論理関数処理に基づいて修正する。

このように，誤り追跡入力に基づく処理と論理関数処理を併用することによって，効率的に組合せ箇所を削減し，現実的な処理時間内での論理診断を実現する。

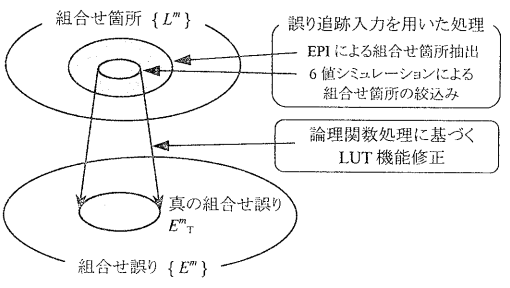

図 1　EXL 法の概念

2. 用語と問題の定義

必要な用語と，対象とする論理診断問題を定義する。

定義 1　$f_s = (f_{s1}, \cdots, f_{sp})$ および $f_g = (f_{g1}, \cdots, f_{gp})$ を，n 入力変数ベクトル $x = (x_1, \cdots, x_n)$ に対する p 出力ブール関数ベクトルとする。ここで f_s は機能仕様に，f_g は設計誤りを含む回路の外部出力関数に対応するとする。一つのブール変数 X または $\overline{X}$ を含む入力ベクトル $\alpha = (a_1, \cdots, a_{k-1}, X / \overline{X}, a_{k+1}, \cdots, a_n)$ に対して

$$f_{sj}(\alpha) = X \text{ かつ } f_{gj}(\alpha) = a \quad (a \text{ は定数 0 or 1}) \qquad (1)$$

が成立するとき，α を **誤り追跡入力** [6] と呼ぶ。 □

定義 2　$B = \{0, 1\}$, $T = \{0, 1, *\}$ とする。ここで，$*$ は**ドントケア**を表す。LUT は，真理値表で示される任意の機能を実現する論理素子である。その機能は，**LUT 入力ベクトル** $y = (y_1, \cdots, y_k)$ に対する **LUT 関数** $h : B^k \rightarrow B$ で表現される。本稿では，最大 4 入力（$k \leq 4$）の LUT を扱う。さらに，修正法に関する自由度を表すため，LUT 関数を不完全記述関数 $h : B^k \rightarrow T$ として扱う。 □

定義 3　対応する LUT 関数に誤りを想定する LUT のことを，**誤り箇所** l_i と呼ぶ。m 個の誤り箇所からなる空でない集合を，**多重度 m の組合せ箇所** $L^m = \{l_i \mid i = 1 \cdots m\}$ と呼ぶ。 □

定義 4　**誤り種類** t_i として，LUT 機能誤りを扱う。t_i はその修正法，すなわち正しい LUT 機能（LUT 関数）の情報をも含む。 □

定義 5　**誤り** e_i は，誤り箇所 l_i と誤り種類 t_i の組 (l_i, t_i) で表される。**多重度 m の組合せ誤り** $E^m = \{e_i \mid i = 1 \cdots m\}$ は，m 個の誤りからなる空でない集合である。ここで，すべての i ($= 1 \cdots m$) について，$e_i \in E^m$ の要素 l_i が L^m に含まれるとき，E^m を **L^m における組合せ誤り**という。この場合，

L^m は E^m に含まれるともいい，$L^m = L(E^m)$ と表す。 □

定義 6 その修正によって機能仕様を満たす回路が得られる多重度 m の組合せ誤りを，**真の組合せ誤り** ${E^m}_T$ と呼ぶ。多重度 m に対して存在するすべての ${E^m}_T$ からなる集合を，**真の組合せ誤り集合** $C_{{E^m}_T} = \{{E^m}_T\}$ と呼ぶ。 □

定義 7 真の組合せ誤りに含まれる組合せ箇所を，**真の組合せ箇所** ${L^m}_T$ と呼ぶ。一つ以上の真の組合せ誤り ${E^m}_T \in C_{{E^m}_T}$ に対して，${L^m}_T = L({E^m}_T)$ が成り立つ。 □

EXL 法では，次の論理診断問題を対象とする。

論理診断問題：LUT 型 FPGA 回路と機能仕様をもとに，$C_{{E^m}_T} \neq \phi$ を満たす最小の多重度 m に対して真の組合せ誤り集合 $C_{{E^m}_T}$ を求める。 □

3. 誤り追跡入力を用いた組合せ箇所の抽出

EXL 法では，考慮すべき組合せ箇所数が多い初期段階において，誤り追跡入力を用いた EPI および 6 値シミュレーションによって，効率的に組合せ箇所の抽出と絞り込みを行う。これらの手法については文献 [3] で述べられているため，本節ではおもに LUT 型 FPGA に対応するための変更点について述べる。

3.1 EPI(Error Possibility Index)

本手法では，EPI と呼ぶ誤り可能性の指標を用いることにより，考慮すべき組合せ箇所を抽出する。

論理診断においては，すべての不一致外部出力，すなわち機能仕様と異なる値を示す外部出力について，正しい出力値に変化するような修正法を求める必要がある。EPI は，不一致外部出力に対する可制御性を表す。ある誤り追跡入力に対して発生する各不一致外部出力毎に，各 LUT の EPI を設定する。いかなる修正を施しても着目する不一致外部出力の信号値を変化させない組合せ箇所に対しては，EPI の和が 1 未満となるように設定する。EPI の和が 1 以上となる組合せ箇所のみを抽出する。

ある LUT における LUT 出力の EPI を epi_{out}，各 LUT 入力 y_i の EPI を $epi(y_i)$ とすると，以下の手順により epi_{out} から $epi(y_i)$ を計算する。

Step 1 各 i $(1 \le i \le k)$ について $epi(y_i) = 0$ とする。

Step 2 $S_n = 0$ とする。

Step 3 $S_n = S_n + 1$ とし，$S_n > k$ ならば終了。

Step 4 S_n 本の LUT 入力線の信号値を同時に変更することで LUT 出力値が変化するとき，その各入力線を要素とする集合を活性化入力線集合 Y_s とする。

Step 5 Y_s を $Y_a = \{y_{a_i} \mid epi(y_{a_i}) > 0,\ y_{a_i} \in Y_s\}$ (EPI 割当済) と $Y_u = \{y_{u_i} \mid epi(y_{u_i}) = 0,\ y_{u_i} \in Y_s\}$ (EPI 未割当) に分割する。

Step 6 各 LUT 入力 $y_{u_i} \in Y_u$ に対して，$epi(y_{u_i})$ を次のように計算する。

$$epi_{sub} = \max\left(epi_{out} - \sum_{y_{u_i} \in Y_a} epi(y_{a_i}),\ 0\right),$$

$$epi(y_{u_i}) = \frac{epi_{sub}}{|Y_u|}.$$

Step 7 Step 3 に戻る。

3.2 6 値シミュレーション

EPI を用いた判断は，外部出力に接続するゲートの出力値変化のみに着目している。次の段階では，変化後の外部出力値が機能仕様を満たす理想回路の出力値と等しくなる可能性に着目し，より厳密な判断を行う。網羅的な処理を回避するため，修正後の LUT 機能を特定せずに信号値を評価している。誤りを想定する各箇所について，可能性のある誤り種類のそれぞれが存在する場合をまとめて評価するために，不定信号値 D と E を導入する。D は $\{0, 1\}$，E は $\{0, 1, X, \overline{X}\}$ のいずれかをとる可能性を表す。これらの信号値に，誤り追跡入力の 0, 1, X, $\overline{X}$ の 4 値を加えた 6 値を用いて，外部出力まで X が伝搬する可能性の有無を判断する。理想回路で X が出力される外部出力について，実回路に関する 6 値シミュレーションの結果が X または E とならない場合は修正不可能と判断して，誤りを想定した組合せ箇所を削除する。

図 2 に，6 値シミュレーションにおける LUT 出力信号値の計算方法を示す。

(a) 誤りを想定しない箇所 $l_i \notin L^m$

a1) 入力信号値 {0, 1} のみで出力値 v_o が決定される場合
v_o を LUT の出力信号値とする

a2) 入力信号値に X または $\overline{X}$ が含まれる場合
次表を用いて v_o を計算する

$f_{X=0}$ \ $f_{X=1}$	0	1	–
0	0	$\overline{X}$	E
1	X	1	E
–	E	E	E

–：未定義

a3) 入力信号値に X または $\overline{X}$ が含まれない場合
入力に E が含まれるなら $v_o = E$ とする
入力に E が含まれないなら $v_o = D$ とする

(b) 誤りを想定する箇所 $l_i \in L^m$

b1) 入力に $\{X, \overline{X}, E\}$ のいずれかが含まれる場合 $v_o = E$，それ以外は $v_o = D$ とする

図 2 6 値シミュレーションにおける LUT 出力信号値 v_o の計算

4. 論理関数処理に基づくLUT機能修正

本節では，誤り追跡入力を用いて抽出された各組合せ箇所に対して，論理関数処理に基づいて LUT 機能を修正する方法について述べる。

4.1 用語の定義

まず，必要となる用語を定義する。ブール演算子に関する定義は，[7], [8] に基づいている。

定義 8 集合 A, E に対して，$A \subseteq E$ が成り立つとする。このとき，A の**特徴関数** $f_A : E \rightarrow B$ は，

$$f_A(x) = \begin{cases} 1 & (x \in A) \\ 0 & (x \notin A) \end{cases} \tag{2}$$

を満たす関数である。 □

定義 9 n 入力ブール関数 $f(x_1, \cdots, x_i, \cdots, x_n)$ に対して，$f|_{x_i=1} = f(x_1, \cdots, 1, \cdots, x_n)$, $f|_{x_i=0} = f(x_1, \cdots, 0, \cdots, x_n)$ を，それぞれ f の x_i に関する**コファクタ**，$\bar{x}_i$ に関するコファクタと呼ぶ。 □

定義 10 n 入力ブール関数 $f(x_1, \cdots, x_n)$ のブール関数 c に関する**一般化コファクタ**(*generalized cofactor*)は $f|_{c=1}$ で表され，次の条件を満足する。

$$c \cdot f = c \cdot f|_{c=1} \tag{3}$$

$$\therefore c \cdot f \le f|_{c=1} \le \bar{c} + f \tag{4}$$

定義 11 n 入力ブール関数 $f(x_1, \cdots, x_n)$ の $x_q = (x_{i_1}, \cdots, x_{i_q})$ $(q < n)$ による**スムース演算** $S_{x_q} f$ は，

$$S_{x_q} f = S_{x_{i_1}} S_{x_{i_2}} \cdots S_{x_{i_q}} f ,$$

$$S_{x_{ij}} f = f|_{x_{ij}=0} + f|_{x_{ij}=1}$$

で表される。 □

定義 12 ある誤り箇所 l_i の出力を内部変数に置き換えるとき，それを**LUT 出力変数**と呼び，誤り箇所と同一の記号 l_i で表す。ある組合せ箇所 $L^m = \{l_1, \cdots, l_m\}$ に含まれる各箇所の出力変数 l_i から成るベクトルを，**LUT 出力変数ベクトル** $\boldsymbol{l}$ と呼ぶ。 □

定義 13 ある外部出力 j に関して，**外部入力ベクトル** $x = (x_1, \cdots, x_n)$ と $\boldsymbol{l}$ で表される論理関数を，**外部出力関数** $f'_{gj}(x, \boldsymbol{l})$ と呼ぶ。各出力に対応した f'_{gj} から成るベクトルを，**外部出力関数ベクトル** $\boldsymbol{f}'_g = (f'_{g1}, \cdots, f'_{gp})$ と呼ぶ。 □

定義 14 ある LUT l_i における入力 y_j が表す関数を，**LUT 入力関数** $g_{l_i j} : B^n \rightarrow T$ と呼ぶ。各出力に対応した $g_{l_i j}$ から成るベクトルを，**LUT 入力関数ベクトル** $\boldsymbol{g}_{l_i} = (g_{l_i 1}, \cdots, g_{l_i k})$ と呼ぶ。**LUT 入力割当て** $y_a \in B^k$ は，k 個の論理定数から成るベクトルである。このとき，**LUT 入力関数一致条件** $g_{l_i eq}(y_a)$ は，$g_{l_i} = y_a$ となる条件

$$g_{l_i eq}(y_a) = \prod_j \left(g_{l_i j} = y_{aj} \right) \tag{5}$$

を表す。 □

定義 15 ある LUT l_i の入力ベクトル $y = (y_1, \cdots, y_k)$ に対する出力関数を，**LUT 関数** $h_{l_i} : B^k \rightarrow T$ で表す。また，ある組合せ箇所 L^m に対応した LUT 関数から成るベクトル $\boldsymbol{h}_l = (h_{l_1}, \cdots, h_{l_m})$ を，L^m **における LUT 関数ベクトル**と呼ぶ。 □

定義 16 **LUT 出力関数** $f_{l_i} : B^n \rightarrow T$ は，ある LUT 出力 l_i の外部入力 $x = (x_1, \cdots, x_n)$ に対する関数を表す。また，$\boldsymbol{f}_l = (f_{l_1}, \cdots, f_{l_m})$ を，L^m における**LUT 出力関数ベクトル**と呼ぶ。 □

図 3 に示すように，各 LUT 出力関数は $f_{l_i} = h_{l_i}(g_{l_i})$ によって求められる。

定義 17 すべての外部出力が機能記述と一致する条件を示す**一致関数** $c(x, l)$ は，

$$c(x, \boldsymbol{l}) = \prod_{j=1}^{p} \left(f'_{gj}(x, \boldsymbol{l}) \equiv f_{sj} \right) \tag{6}$$

で表される。 □

補題 1 着目する組合せ箇所が真の組合せ誤り $E^m{}_T$ となるための，LUT 出力関数ベクトル $\boldsymbol{f}_l$ に関する必要十分条件は，

$$c(x, \boldsymbol{f}_l) \equiv 1 \tag{7}$$

$$\forall i \in \{1, \cdots, m\} : \; f_{l_i} = h_{l_i}(g_{l_i}) \tag{8}$$

である。 □

式 (7) 単独では必要条件であり，十分条件ではない。LUT 入力関数ベクトル $\boldsymbol{g}_{l_i} = (g_{l_i 1}, \cdots, g_{l_i k})$ により k 入力で f_{l_i} を実現するために，式 (8) も必要となる。

定義 14～16 において，関数 $g_{l_i j}, h_{l_i}, f_{l_i}$ はドントケアを含む $T = \{0, 1, *\}$ のいずれかの値をとると想定している。

また本手法ではドントケア条件を表現するため，以下に示すドントケア変数を導入する。

定義 18 **ドントケア変数** $d_{l_i j}$ は，ある LUT l_i の出力値

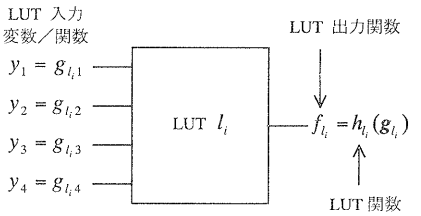

図 3 誤り箇所 l_i の入出力関数

表1　LUT 入力割当て y_a と LUT 入力関数一致条件

入力割当て $y_a = y_1 y_2 y_3 y_4$	LUT 入力関数一致条件 $g_{l_i eq}(y_a)$
0 0 0 0	$\overline{g_{l_i 1}}\ \overline{g_{l_i 2}}\ \overline{g_{l_i 3}}\ \overline{g_{l_i 4}}$
0 0 0 1	$\overline{g_{l_i 1}}\ \overline{g_{l_i 2}}\ \overline{g_{l_i 3}}\ g_{l_i 4}$
⋮	⋮
1 1 1 1	$g_{l_i 1}\ g_{l_i 2}\ g_{l_i 3}\ g_{l_i 4}$

表2　LUT 関数 $h_{l_i}(y_a)$ の決定

$g_{l_i eq}(y_a)\cdot c_{0only}$	$g_{l_i eq}(y_a)\cdot c_{1only}$	$h_{l_i}(y_a)$
$\not\equiv 0$	$\equiv 0$	0
$\equiv 0$	$\not\equiv 0$	1
$\equiv 0$	$\equiv 0$	*
$\not\equiv 0$	$\not\equiv 0$	競合

を，ある LUT 入力割当てについては定数(0 or 1)に決定する必要がない場合に割り当てる。**ドントケア変数ベクトル** $\boldsymbol{d}_{l_i} = (d_{l_i 1}, \cdots, d_{l_i n_i})$ は，LUT l_i に含まれるドントケア変数から成るベクトルである。また，ある組合せ箇所に対応した**ドントケア変数連結ベクトル** $\boldsymbol{d}_{l_1 \to l_i}$ は，

$$\boldsymbol{d}_{l_1 \to l_i} = (\boldsymbol{d}_{l_1}, \cdots, \boldsymbol{d}_{l_i}) = (d_{l_1 1}, \cdots, d_{l_1 n_1}, \cdots, d_{l_i 1}, \cdots, d_{l_i n_i}) \quad (9)$$

を表す。 □

定義 19　LUT 入出力に関連する関数 $g_{l_i j}, h_{l_i}, f_{l_i}$ は，ドントケア変数ベクトルを用いて，

$$g_{l_i j} = g'_{l_i j}(\boldsymbol{x}, \boldsymbol{d}_{l_1 \to l_{i-1}}) \quad (10)$$

$$h_{l_i} = h'_{l_i}(\boldsymbol{y}, \boldsymbol{d}_{l_i}) \quad (11)$$

$$f_{l_i} = f'_{l_i}(\boldsymbol{x}, \boldsymbol{d}_{l_1 \to l_i}) \quad (12)$$

と表される。 □

本稿では，関数を区別する必要がない限り，$g'_{l_i j}, h'_{l_i}, f'_{l_i}$ を $g_{l_i j}, h_{l_i}, f_{l_i}$ と表現する。

ドントケア変数は，ある入力ベクトルに対する真理値が 0 または 1 に決定できない場合に割り当てられるが，他の箇所の修正にともない，いずれかに決定されることがある。

EXL 法が対象とする論理診断問題のうち，LUT 機能を修正する部分については次のように表現できる。

LUT 機能修正問題：与えられた組合せ箇所 L^m が，真の組合せ箇所 $L^m{}_T$ かどうかを判断する。$L^m = L^m{}_T$ の場合には，L^m に対するすべての $E^m{}_T$ を求める。すなわち，式 (7), (8) を満たすすべての $\boldsymbol{h}_l$ を求める。 □

4.2　一致関数と充足条件

LUT 出力変数ベクトル $\boldsymbol{l}$ に論理定数（0 または 1）を要素とする割当て $\boldsymbol{a} = (a_1, \cdots, a_m)$ を与えた時の一致関数 $c(\boldsymbol{x}, \boldsymbol{l})$ を，$c_{\boldsymbol{a}}$ または $c_{a_1 a_2 \cdots a_m}$ と表記する。例えば，$m = 3$ に対して，$\boldsymbol{l}$ に $\boldsymbol{a} = (0, 0, 0)$ を与えた時の一致関数を c_{000} と表記する。すべての m 次元ブールベクトル $\boldsymbol{a} \in B^m$ に対する $c_{\boldsymbol{a}}$ の和 c_{sum} について

$$c_{\text{sum}} = \sum_{\boldsymbol{a} \in B^m} c_{\boldsymbol{a}} = c_{0\cdots 00} + c_{0\cdots 01} + \cdots + c_{1\cdots 11} \equiv 1 \quad (13)$$

を満足しない組合せ箇所に関しては，LUT 出力変数ベクトル $\boldsymbol{l}$ にいかなる値を設定しても，機能記述と一致しない。よって，そのような組合せ箇所 $L^m = \{l_1, \cdots, l_m\}$ は候補から除く。式 (13) を**一致関数充足条件**，この条件を満足する組合せ箇所を**一致関数充足組合せ箇所**と呼び，記号 $L^m{}_c$ で示す。

4.3　単一設計誤りを想定した LUT 機能修正

まず多重度 $m = 1$ の場合，すなわち単一設計誤りを想定した場合の LUT 関数の修正方法について述べる。

ある組合せ箇所 $L^1 = \{l_1\}$ に対して，一致関数 c を求め，$c_{\text{sum}} = c_0 + c_1 \equiv 1$ が成立するかどうかを判定する。もし，成立しないなら，その組合せ箇所を削除する。また，成立する場合には以下の手順により修正を行う。

$c_{\text{sum}} = c_0 + c_1 \equiv 1$ が満足されるなら，c_0 と c_1 の関係は図 4 のようになる。ここで，$c_0 + c_1 \equiv 1$ であることを用いて次の 2 種類の関数 $c_{0\text{only}}$ と $c_{1\text{only}}$ を導入する。

$$c_{0\text{only}} = c_0 \bar{c}_1 = (c_0 + \bar{c}_0)\bar{c}_1 = \bar{c}_1 \quad (14)$$

$$c_{1\text{only}} = \bar{c}_0 c_1 = (c_1 + \bar{c}_1)\bar{c}_0 = \bar{c}_0 \quad (15)$$

これら二つの関数により，LUT 入力ベクトル $\boldsymbol{y}_a \in B^k$ に対する l_1 の LUT 関数 $h_{l_1}(\boldsymbol{y}_a)$ が以下の方法で決定される。

Step 1　l_1 の LUT 入力関数 $g_{l_1 1}, \cdots, g_{l_1 k}$ を求める。

Step 2　$c_{0\text{only}} = \bar{c}_1$，$c_{1\text{only}} = \bar{c}_0$ を求める。

Step 3　表 1 に示すように，各入力割当て $\boldsymbol{y}_a$ に対して LUT 入力関数一致条件 $g_{l_i eq}(\boldsymbol{y}_a)$ を求める。

Step 4　表 2 に従って，各入力割当て $\boldsymbol{y}_a \in B^k$ に対応する LUT 関数 $h_{l_1}(\boldsymbol{y}_a)$ を決定する。

$g_{l_i eq}(\boldsymbol{y}_a)\cdot c_{0\text{only}} \not\equiv 0$ となる LUT 入力割当て $\boldsymbol{y}_a$ は，$c_{0\text{only}}$ の条件下で入力関数ベクトル $\boldsymbol{g}_{l_i}$ が入力割当て $\boldsymbol{y}_a$ と一致する外部入力変数ベクトルが存在することを示す。そのような入力割当て $\boldsymbol{y}_a$ に対して，一致関数 $c = 1$ とする

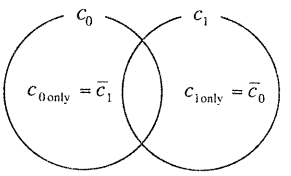

図 4　c_0 と c_1 の関係の例

には $l_1 = 0$ とする必要があるので，$h_{l_1}(y_a) = 0$ を割り当てる。同様に $g_{l_1eq}(y_a) \cdot c_{1\text{only}} \neq 0$ の場合には，$h_{l_1}(y_a) = 1$ を割り当てる。

一方，$g_{l_1eq}(y_a) \cdot c_{0\text{only}} \equiv 0$ かつ $g_{l_1eq}(y_a) \cdot c_{1\text{only}} \equiv 0$ となる y_a については，次の場合が考えられる。

i) いかなる外部入力ベクトルに対しても，y_a の組合せが発生しない（$g_{l_1eq}(y_a) \equiv 0$）。

ii) y_a の組合せを生じるすべての外部入力ベクトルに対して $c_{0\text{only}} = c_{1\text{only}} = 0$ となり，$h_{l_1}(y_a) = 0$ または 1 のいずれでも構わない。

いずれの場合も h_{l_1} の値は一致関数 c に影響を及ぼさないので，ドントケア $*$ とする。

ある入力割当て y_a に対して $g_{l_1eq}(y_a) \cdot c_{0\text{only}} \neq 0$ かつ $g_{l_1eq}(y_a) \cdot c_{0\text{only}} \neq 0$ と競合が発生した場合，同一の入力割当て y_a に対して，外部入力ベクトル x によって異なる LUT 出力とする必要が生じる。よって，その箇所での修正は不可能となり，注目している組合せ箇所 $L^m{}_c$ を候補から除く。

4.4 多重設計誤りへの対応

4.4.1 多重設計誤りに対する LUT 機能修正法の概要

多重度 m の設計誤りを含む回路に対する，LUT 関数の決定方法について述べる。一致関数充足組合せ箇所 $L^m{}_c = \{l_1, \cdots, l_m\}$ に対して，外部入力に近い方から順に，$l_1, l_2, \cdots, l_m$ と並んでいるとする。すなわち，l_i から外部出力へのパスに l_j $(j > i)$ が含まれることはあるが，その逆はないものとする。ある誤り箇所 l_i よりも出力側の各箇所 l_i に関する一致関数のスムース演算結果

$$c_{S_i} = S_{l_{i+1}} S_{l_{i+2}} \cdots S_{l_m} c \qquad (16)$$

を **l_i に関する一致関数**と呼ぶ。

多重設計誤りの場合は，式 (16) を用いて，入力側の

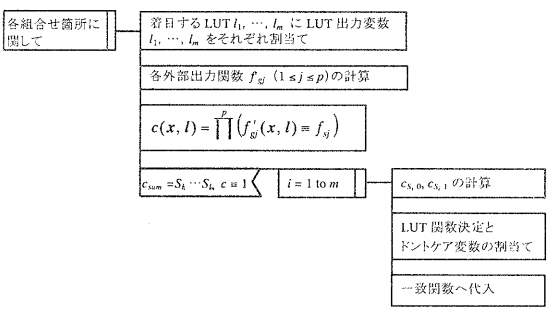

図 5 多重度 m の設計誤りを想定した LUT 機能修正

LUT，すなわち l_1 から順に LUT 関数を決定する。図 5 に処理内容を示す。各誤り箇所 l_i に対する繰り返し処理部について次に示す。

Step 1 $i = 1$ とする。

Step 2 着目する l_i に関する一致関数 c_{S_i} に $l_i = 0, 1$ を代入し $c_{S_i 0} = c_{S_i}\big|_{l_i=0}$，$c_{S_i 1} = c_{S_i}\big|_{l_i=1}$ をそれぞれ計算する。この時，一致関数充足条件は，

$$c_{S_i 0} + c_{S_i 1} \equiv 1 \qquad (17)$$

と表される。$c_{S_i 0}$ と $c_{S_i 1}$ を 4.3 節で述べた手法における c_0, c_1 としてそれぞれ扱い，LUT 関数の決定とドントケア変数の割当てを行う。なお LUT 関数の決定方法については，次節で述べる。

Step 3 外部入力ベクトル x とドントケア変数連結ベクトル $d_{l_1 \to l_i}$ による l_i の出力関数 $f'_{l_i d}(x, d_{l_1 \to l_i})$ を求める。

Step 4 一致関数 c_{S_i} に，$l_i = f'_{l_i d}(x, d_{l_1 \to l_i})$ を代入。

Step 5 $i < m$ なら $i = i + 1$ として，Step 2 へ戻る。

Step 6 $i = m$ なら $c \equiv 1$ であることを確認し，LUT 関数ベクトル h_l を解として出力する。

4.4.2 LUT 関数の決定

本節では，多重設計誤りに対応した LUT 関数の決定方法について述べる。

前節 Step 2 においてドントケア変数が割り当てられ，かつ LUT 入力関数にドントケア変数が含まれる場合は，ドントケア変数を考慮した処理を行う必要がある。よって，LUT 入力関数にドントケア変数が含まれる場合と含まれない場合の二通りについて述べる。

(1) LUT 入力関数にドントケア変数が含まれない場合

この場合は，4.3 節で述べた単一設計誤りにおける LUT 機能修正と同様の方法を用いる。決定された LUT 関数にドントケアが含まれる場合は，各ドントケアにそれぞれ異なるドントケア変数を割り当てる。

(2) LUT 入力関数にドントケア変数が含まれる場合

ドントケア変数ベクトル $d = d_{l_1 \to l_{i-1}}$ とし，d に含まれるドントケアの数を n_d とする。

$g_{l_ieq}(y_a) \cdot c_{0\text{only}}\big|_d \neq 0$ かつ $g_{l_ieq}(y_a) \cdot c_{1\text{only}}\big|_d \neq 0$ となる d については y_a に対する競合が発生する。このことを利用して，2^{n_d} 通りのドントケア変数ベクトルのうち競合が起こらないドントケア変数ベクトル集合 D_r を求める。D_r の特徴関数 f_{D_r}，l_i に関する LUT 入力関数一致条件 $g_{l_ieq}(y_a)$，$c_{0\text{only}} = \bar{c}_1$，$c_{1\text{only}} = \bar{c}_0$ を用いて，$y_a \in B^k$ に対する LUT 関数 $h_{l_i}(y_a)$ を求める。

図 6 に手順を示す。まず，y_a に対して，$\boldsymbol{d}$ のもとで $g_{l_i eq}(y_a)\cdot c_{0\text{only}}|_{\boldsymbol{d}} \neq 0$ となる集合 $D^0_{y_a}$ を求める。このような $\boldsymbol{d} \in D^0_{y_a}$ については LUT 入力割当て y_a に対する l_i の論理を 0 とする必要がある。次に，$\boldsymbol{d}$ のもとで $g_{l_i eq}(y_a)\cdot c_{1\text{only}}|_{\boldsymbol{d}} \neq 0$ となる集合 $D^1_{y_a}$ を求める。同様に，$\boldsymbol{d} \in D^1_{y_a}$ については l_i の論理に 1 を割り当てる必要がある。ここで，$\boldsymbol{d} \in D^0_{y_a} \cap D^1_{y_a}$ については LUT 入力割当て y_a に対して競合が発生するため，$f_{D_r} = \overline{f_{D^0_{u_a}} \cdot f_{D^1_{u_a}}} \cdot f_{D_r}$ とすることにより $\boldsymbol{d}$ を D_r の要素から取り除く。

$\forall y_a \in B^k$ に対して上記の処理を行うことにより，許容されるドントケア変数の組合せを限定できる。続いて，各 y_a に対して LUT 関数 $h_{l_i}(y_a)$ を以下のように定める。

(1) $f_{D^0_{y_a}}\Big|_{f_{D_r}=1} \equiv 1$ ならば， 0

(2) $f_{D^1_{y_a}}\Big|_{f_{D_r}=1} \equiv 1$ ならば， 1

(3) その他の場合，ドントケア変数 d_a ($a = n_d + 1$, …)

(3) の場合には，y_a に対する出力論理をこの段階では 0 または 1 に決定できないのでドントケア変数 d_a を割り当てるが，以後の処理でいずれかに決定される場合がある。また，d_a の追加にともない，f_{D_r} を次式により拡張する。

$$f'_{D_r} = \overline{d_a} \cdot \overline{f_{D^1_{y_a}}} \cdot f_{D_r} + d_a \cdot \overline{f_{D^0_{y_a}}} \cdot f_{D_r} \qquad (18)$$

5. 実験評価

本稿で提案した EXL 法を C 言語 (gcc ver. 2.7.2.1) を用いて計算機 (Sun Ultra2 / model 2200) 上に実装するとともに，実験評価を行った。

実験には，表 3 に示す ISCAS'85 ベンチマーク回路 [9] に対して無作為に誤りを 1 個から 3 個挿入した回路例を用いた。1 種類の回路に対して多重度ごとに 20 例，合計 3 ×20×8 = 480 例について実験を行った。その際，BDD ノード数を 200 万に制限した。なお，その制限を越える

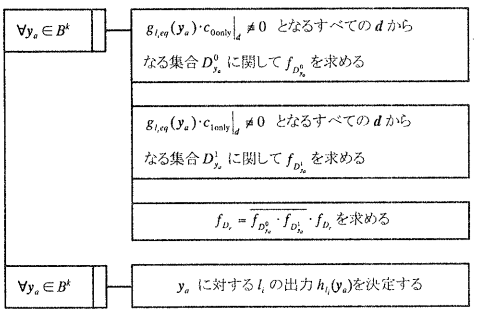

図 6 ドントケア変数を考慮した LUT 関数の決定

表 3 ISCAS'85 ベンチマーク回路

回路名	ゲート数	LUT 数	外部入力数	外部出力数
C432	160	109	36	7
C499	202	94	41	32
C880	383	184	60	26
C1355	546	166	41	32
C1908	880	346	33	25
C2670	1,193	598	233	140
C3540	1,669	742	50	22
C5315	2,307	1,146	178	123

回路例に対しては，ノード制限を 800 万に拡張し，再実験を行った。C2670 に対する実験では，7 例がノード制限を越えたものの同一パス上に複数の設計誤りを含む回路例への対応が可能となった。

各多重度に対する実験結果を，表 4 から表 6 に示す。なお，表中の値は，すべて幾何平均による結果である。“組合せ箇所数”の欄は，考慮すべき組合せ箇所が各処理によって削減された過程を示している。表中の “1st hit” は最初の解が発見されるまでの処理時間を表している。また，“真の組合せ誤り数”，すなわち解の個数はドントケアを展開して数えた結果である。

実験結果より，EXL 法により実用的な処理時間での論理診断が可能となることが確認された。その要因として，誤り追跡入力を用いた処理による組合せ箇所の削減効果が挙げられる。とりわけ最初の段階で，EPI によって組合せ箇所の抽出が効率的に行われている。さらに，論理関数処理に基づいて LUT 機能を修正するため，100 % の限定率を達成することができた。その一方で，最大処理時間に示されるように，長い処理時間を必要とする回路も数例存在した。この点を改善するためには，ドントケア変数の BDD における順序について考慮する必要があると考えられる。さらに 6 値シミュレーション後に別の手法を導入することで組合せ箇所のさらなる削減が可能となれば，診断可能な設計誤りの多重度向上に結び付くと考えられる。

6. まとめ

本稿では，誤り追跡入力と論理関数処理の併用によって，LUT 型 FPGA を対象とする論理診断を行う EXL 法を提案した。誤り追跡入力を用いた処理によって効率的に組合せ箇所を削減し，論理関数処理に基づく LUT 機能修正を行うことで，誤りを見逃すことなく現実的な処理時間で論理診断を行うことが可能となった。

今後の課題として，ドントケア変数の変数順序を考慮することによる処理時間の削減，また論理関数処理に基づく LUT 機能修正前の段階でさらに組合せ箇所を削減する手法の導入が挙げられる。

表 4　単一設計誤りに対する実験結果 (m = 1)

回路名	組合せ箇所数				平均処理時間 (s)		最大処理時間 (s)	真の組合せ誤り数
	Total	aft. EPI	aft. 6-val sim.	Final	1st hit	Total time		
C432	109	2.93	1.96	1.09	2.9	2.9	3.7	5.20
C499	94	3.55	2.46	1.13	7.0	7.8	13.8	2.95
C880	184	2.32	1.61	1.11	3.4	3.4	7.8	3.37
C1355	166	2.30	2.10	1.06	7.9	8.5	14.1	3.09
C1908	346	7.89	5.21	1.40	8.6	9.0	12.3	9.36
C2670	598	7.11	3.62	1.15	12.2	12.7	92.0	21.97
C3540	742	6.55	2.49	1.04	24.1	25.5	43.6	5.39
C5315	1,146	4.26	2.73	1.25	14.4	14.6	18.7	5.79
平均	293	4.14	2.60	1.15	8.2	8.5	29.1	5.64

表 5　多重度 2 の設計誤りに対する実験結果 (m = 2)

回路名	組合せ箇所数				平均処理時間 (s)		最大処理時間 (s)	真の組合せ誤り数
	Total	aft. EPI	aft. 6-val sim.	Final	1st hit	Total time		
C432	5,995	13.32	3.99	1.07	3.1	3.3	5.7	4.86
C499	4,465	20.16	17.72	1.30	15.1	26.3	1,322.2	3.80
C880	17,020	6.22	2.77	1.27	4.4	4.9	11.3	16.98
C1355	13,861	3.34	3.09	1.14	15.4	19.1	1,287.1	33.47
C1908	60,031	80.50	16.06	1.98	16.0	22.0	2,128.5	21.03
C2670	179,101	52.94	12.59	1.47	31.7	85.7	12,990.4	305.62
C3540	275,653	65.99	16.10	1.21	34.9	38.0	5,463.1	58.56
C5315	657,231	25.81	11.15	1.52	14.4	18.9	445.8	105.10
平均	43,275	21.18	8.28	1.35	12.9	17.9	541.4	28.33

表 6　多重度 3 の設計誤りに対する実験結果 (m = 3)

回路名	組合せ箇所数				平均処理時間 (s)		最大処理時間 (s)	真の組合せ誤り数
	Total	aft. EPI	aft. 6-val sim.	Final	1st hit	Total time		
C432	2.16×10^5	1,054	27.9	1.24	5.3	6.1	87.2	27.98
C499	1.39×10^5	134	84.8	1.37	33.9	92.4	2,080.6	11.97
C880	1.04×10^6	8.25	2.63	1.29	7.6	8.6	63.4	20.17
C1355	7.63×10^5	51.1	39.5	1.45	38.7	99.1	5,328.6	68.60
C1908	6.90×10^6	23,230	1,790	2.68	152.2	713.8	16,976.5	39.70
C2670	3.56×10^7	784.7	56.5	1.41	41.5	528.2	36,526.2	9,752.1
C3540	6.81×10^7	258.6	26.6	1.11	47.5	78.0	102.7	974.6
C5315	2.51×10^8	200.2	28.8	1.96	38.7	43.7	163.1	2,996.9
平均	4.21×10^6	294.3	45.7	1.51	29.8	70.6	945.2	164.1

参考文献

[1] Y. Watanabe and R. K. Brayton, "Incremental synthesis for engineering changes," ICCD-92, pp. 40-43, 1991.

[2] M. Fujita, Y. Tamiya, Y. Kukimoto, and K. C. Chen, "Application of Boolean unification to combinational logic synthesis," ICCAD-91, pp. 510-513, 1991.

[3] M. Tomita, T. Yamamoto, F. Sumikawa, and K. Hirano, "Rectification of multiple logic design errors in multiple output circuits," Proc. 31st DAC, pp. 212-217, 1994.

[4] M. Fujita, Y. Tamiya, Y. Kukimoto, and K. C. Chen, "Application of Boolean unification to combinational logic synthesis," ICCAD-91, pp. 510-513, 1991.

[5] I. Pomeranz and S. M. Reddy, "On error correction in macro-based circuits," IEEE Trans. CAD, vol. 16, no. 10, pp. 1088-1100, 1997.

[6] M. Tomita, N. Suganuma, and K. Hirano, "Pattern generation for locating logic design errors," IEICE Trans., vol. E77-A, no. 5, pp. 881-893, May 1994.

[7] H. J. Touati, H. Savoj, B. Lin, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Implicit state enumeration of finite state machines using BDD's," ICCAD-90, pp. 130-133, 1990.

[8] G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994.

[9] F. Brglez and H. Fujjiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translation in FORTRAN," ISCAS'85, 1985.