# 順序回路のソフトエラー率解析手法の非明示的列挙による高速化について

松永, 裕介
九州大学大学院システム情報科学研究院

赤峰, 悠介
九州大学大学院システム情報科学府



出版情報：電子情報通信学会技術研究報告. VLD. 109 (201), pp.31-36, 2009-09. The Institute of Electronics, Information and Communication Engineers
バージョン：
権利関係：

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

# 順序回路のソフトエラー率解析手法の非明示的列挙による高速化について

松永　裕介†　　赤峰　悠介††

† 九州大学大学院システム情報科学研究院
〒 819–0395 福岡市西区元岡 744
†† 九州大学大学院システム情報科学府
〒 819–0395 福岡市西区元岡 744
E-mail: †matsunaga@ait.kyushu-u.ac.jp

**あらまし**　中性子線などの影響で論理回路中の論理値に一時的な誤りが発生することをソフトエラーと呼ぶ．順序回路中において論理素子やフリップフロップで発生したソフトエラーが外部出力まで伝搬する確率を計算するためには順序回路が実現している有限状態機械を解析する必要があり，単純な方法では多くの計算量を必要とする．本稿では順序回路の等価性検証で用いられる，2 分決定グラフを用いた非明示的列挙手法を応用して高速に順序回路のソフトエラー耐性を評価する手法について述べる．

**キーワード**　ソフトエラー, 順序回路, 有限状態機械, 2 分決定グラフ, 非明示的列挙手法

# On accelleration of SER analysis for sequential circuits using implicit enumeration

Yusuke MATSUNAGA† and Yusuke AKAMINE††

† Faculty of Information Science and Electorical Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka, 819–0395 Japan
†† Graduate School of Information Science and Electorical Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka, 819–0395 Japan
E-mail: †matsunaga@ait.kyushu-u.ac.jp

**Abstract**　Soft-error is a phenomena that the output value of a logic gate flips transiently because of neutron particle strike, etc. Exactly evaluating the soft-error rate for sequential circuits requires analysis of finite state machine corresponding to the given circuit, which is known to be inefficient if using a naive method. This paper proposes an efficient algorithm to analyze propagation effect caused by soft-errors, which utilizes binary decision diagrams for implicitly enumeration.

**Key words**　soft error, sequential circuit, finite state machine, binary decision diagram, implicit enumeration method

## 1. はじめに

近年，LSI の信頼性において中性子起因のソフトエラーが問題として認識されつつある．従来は特に SRAM のようなメモリ素子の値が反転してしまう現象について多くの解析がなされ，その対策として多重化やエラー検出・訂正符号の導入などが提案されている．一方，ランダムロジックに対するソフトエラーはメモリに比べると確率が小さいことなどからあまり重点的には研究されておらず，また，メモリに比べると規則性がないことから単純で効果的な解析手法や対策は知られていない．

ランダムロジックに用いられる回路素子の中でも特にフリップフロップは値を保持しつづけるためソフトエラーの影響を受けやすい．フリップフロップに対してソフトエラー耐性を高める手法はいくつか提案されている [1], [2] がそのどれもが面積，遅延，消費電力などの点でオーバーヘッドを持つ．そのため，必要なソフトエラー耐性を確保しつつ，できる限り上記のオーバーヘッドを抑える設計がもっとも望ましい．

そのような設計を行うために必要な技術の一つはソフトエ

ラー耐性の評価手法である．フリップフロップや論理ゲートなどの回路素子単体におけるソフトエラーの発生確率はデバイスシミュレーションや回路シミュレーションおよび，実チップに対する中性子照射実験などからある程度値を推定することは可能である．一方，フリップフロップを含んだ論理回路は順序回路として動作するため，回路の出力値は現在の入力値だけでなく，内部に保持している状態の影響を受ける．そのため，ソフトエラーによって回路が誤動作したとしてもそれが直ぐに外部出力に現れるとは限らず，何回かの状態遷移を経た後でエラーが顕在化する場合もありうる．論理回路におけるソフトエラーの振る舞いを評価する手法はいくつか提案されているが [3]〜[8]，その多くは組み合わせ回路を対象としている．唯一 [8] は順序回路を対象としているが，組み合わせ回路部分を数時刻分時間展開しただけで本質的には順序回路を扱っているとは言い難い．これに対して著者らは順序回路におけるソフトエラーの振る舞いを確率的に遷移する状態機械 (マルコフモデル) としてモデル化することで，ソフトエラー耐性を解析する手法を提案している [9].

この手法は与えられる入力値の出現確率などの仮定の元では厳密な解析を行うものであるが，20〜30 状態の順序回路の解析に数 10 分を要しており，計算時間および摘要可能な回路規模の点から実用的は言えない．そこで，本稿では，順序回路における状態集合や状態遷移関数を 2 分決定グラフ (Binary Decision Diagram: BDD) [10] を用いて非明示的に表すことによって処理を高速化する手法について述べる．以降，2. 節でマルコフモデルを用いた順序回路のソフトエラー耐性評価法を概説し，3. 節で 2 分決定グラフを用いた非明示的手法について述べる．4. で実験結果を示し考察を行う.

## 2. マルコフモデルを用いた順序回路のソフトエラー耐性評価法 [9]

### 2.1 有限状態機械

順序回路とは，有限状態機械 (Finite State Machine: FSM) を論理素子およびフリップフロップなどの記憶素子を用いて実現したもので，出力の値が現在の入力だけでなく，過去の入力系列に依存するという特徴を持つ．記憶素子の値 (の組合せ) は「状態」と呼ばれ，過去の入力系列を出力に反映させるために用いられる.

形式的には有限状態機械 $\mathcal{M}$ は次のような 6 つ組で表される.

$$\mathcal{M} = (I, O, S, S^0, \lambda, \delta)$$

ここで,

- $I$ は入力記号 (入力アルファベット) 集合．$I$ は有限かつ空ではない.
- $O$ は出力記号 (出力アルファベット) 集合．$O$ も有限かつ空ではない．$I$ に含まれる記号と $O$ に含まれる記号は (見た目が同じでも) 区別される.
- $S$ は状態集合．$S$ も有限かつ空ではない.
- $S^0 \subseteqq S$ は初期状態の集合.
- $\lambda : S \times I \to O$ は出力関数，すなわち，現状態と入力記号から出力記号への写像である.
- $\delta : S \times I \to S$ は次状態関数，すなわち，現状態と入力記号から次状態への写像である.

つまり，有限状態機械とは内部に状態を持ち，与えられた入力記号 (と現状態) に従って状態遷移を行い，かつ，入力記号に従った出力記号を出力するものである.

$N$ 個のフリップフロップを持つ順序回路の実現している状態機械は便宜上，$2^N$ 個の状態を持つ．しかし，その全てが初期状態 $S^0$ から到達可能とは限らない.

以降，特にことわりがない限り，順序回路と有限状態機械を同一のものとして扱う.

### 2.2 直積機械

機械 $\mathcal{M}$ がソフトエラーによって通常とは異なる動作をした結果，外部出力に誤った値が出力されるかどうかを調べるためには，正しい動作をしている回路 (以後，正常回路と呼ぶ) と，ソフトエラーによって誤った状態に遷移してしまった回路 (以後，故障回路と呼ぶ) に同一の入力を加えて，出力が異なるかどうか調べれば良い．ここで，正常回路の状態 $S$ を正常状態，故障回路の状態 $S'$ を故障状態と呼ぶことにする．このように 2 つの回路を並べて構成された有限状態機械を直積機械と呼ぶ．直積機械 $P(\mathcal{M})$ の形式的な定義は以下の通り.

$$P(\mathcal{M}) = (I, 0, 1, \Pi, \Pi^0, \lambda^P, \delta^P)$$

ここで,

- $I$ は M の入力記号と同一.
- 出力記号は 0 と 1 の 2 つ.
- 出力関数は $\lambda^P(i, s, s') = \lambda(i, s) \oplus \lambda(i, s')$．つまり，正常回路の出力と故障回路の出力が異なったときに直積機械の出力が 1 となる.
- 状態集合 $\Pi = S \times S' \cup \Pi_f$．$\Pi_f$ は出力にエラーが伝搬したことを表す仮想的な状態.
- 初期状態の集合は，ある状態 $s \in S$ に対して，あるソフトエラーが起こったときに変化した状態を $s_e \in S'$ としたときに，$(s, s_e)$ の全ての組合せとなる.
- 次状態関数 $\delta^P$ は少し複雑で，通常の状態と入力に関しては，$(\delta(i, s), \delta(i, s'))$ となるが，$\lambda^P(i, (s, s')) = 1$ となる入力と状態に対しては，$\delta^P(i, (s, s')) = \pi_f$ となる．また，いかなる入力に対しても $\delta^P(i, \pi_f) = \pi_f$ となる.

最後の次状態関数の定義より明らかなように，$P(\mathcal{M})$ の状態が一旦，$\pi_f$ に遷移すると以降，永久に $\pi_f$ に留まりつづける．以降，$\pi_f$ に遷移することを $\pi_f$ に吸収されると表現することにする.

### 2.3 ソフトエラー耐性の評価

以上のように，正常な回路の振る舞いと，ソフトエラーが起こったあとの回路の振る舞いはそれぞれ有限状態機械としてモデル化することができる．しかし，回路の動作と中性子線が衝突するタイミングにはなんの関連もないため，順序回路の動作から見ればソフトエラーはランダムに起こるものとみなすことができる．また，ソフトエラー発生後に与えられる入力に関しても，ソフトエラー発生のタイミングとは無関係に決まるため,

なんらかの確率分布に従ったランダムなものとみなすことができる．そこで，上記の有限状態機械から入力記号を取り除き，次状態関数および出力関数の入力部分に確率変数を代入することでマルコフモデルを考えることができる．結局，順序回路においてソフトエラーが発生し，その影響が出力まで伝搬する確率を求めることは，この直積機械のマルコフモデルにおいて $\pi_f$ に吸収される確率を求めることに他ならない．

具体的には，次の 2 つのステップで計算を行う．

（1） 直積機械の初期状態 $\pi$ の存在確率 $P_{init}(\pi)$.

（2） 直積機械の各状態 $\pi \in \Pi$ が $\pi_f$ に吸収される確率 $P_{abs}(\pi)$

順序回路 $\mathcal{M}$ においてソフトエラーが発生し，出力に伝搬する確率 $SER(\mathcal{M})$ は次式で与えられる．

$$SER(\mathcal{M}) = \sum_{\pi \in \Pi} P_{init}(\pi) \times P_{abs}(\pi) \tag{1}$$

#### 2.3.1 初期状態の存在確率の計算

初期状態 $\pi = (s, s_e)$ の存在確率は，

- $\pi$ の元となっている正常状態 $s$ の存在確率 $P_{steady}(s)$
- $s$ においてソフトエラー $e$ の起こる確率 $P_{error}(e)$

を用いて，次式で表される．

$$P_{init}(\pi) = P_{steady}(s) \times P_{error}(e) \tag{2}$$

正常状態 $s$ の存在確率 $P_{steady}(s)$ は正常回路の状態遷移確率から以下の様に計算できる．まず，正常回路において状態 $s_i$ から状態 $s_j$ へ遷移する確率を $P_{trans}(s_i, s_jj)$ とする．また，状態 $s_i$ の存在確率を $P_{steady}(s_i)$ とすると，次式が成り立つ．

$$P_{steady}(s_j) = \sum_{s_i \in S} P_{steady}(s_i) \times P_{trans}(s_i, s_j) \tag{3}$$

さらに，全ての状態の存在確率の和は 1 であるので，

$$\sum_{s_i \in S} P_{steady}(s_i) = 1 \tag{4}$$

が成り立つ．状態遷移確率を既知とすれば，これらの式 (3) および (4) から存在確率 $P_{steady}(s_i)$ を求めることができる．

一方，$P_{error}(e)$ はフリップフロップにおいてソフトエラーが発生する確率や，論理素子におけるソフトエラーの影響がフリップフロップまで到達する確率などから計算することができる．

#### 2.3.2 吸収確率の計算

直積機械において，状態 $\pi_i$ から状態 $\pi_j$ へ遷移する確率を $P_{trans}(\pi_i, \pi_j)$ とし，状態 $\pi_i$ から 1 回もしくは複数回の遷移を経て $\pi_f$ に吸収される確率を $P_{abs}(\pi_i)$ とすると，次式が成り立つ．

$$\begin{aligned} P_{abs}(\pi_i) = & \sum_{\pi_j \in \Pi} P_{trans}(\pi_i, \pi_j) \times P_{abs}(\pi_j) \\ & + P_{trans}(\pi_i, \pi_f) \end{aligned} \tag{5}$$

$P_{trans}(\pi_i, \pi_j)$ が既知であれば，式 (6) から $P_{abs}(\pi_i)$ を求めることができる．

### 2.4 順序回路のソフトエラー耐性評価法の実装および問題点

前述の手法で順序回路においてソフトエラーが伝搬する確率を計算することは可能だが，このままでは計算効率の点で大きな問題点がある．たとえば，フリップフロップ数が $N$ の場合，単純には $2^N$ 個の状態が存在するが，その全てが初期状態から到達可能とは限らない．このことは特に直積機械においては顕著であり，$2^{2N}$ 個の状態のうち，実際に到達可能なのはほんのわずかである．もちろん，到達不可能な状態の存在確率は 0 であるので，それらに関係する状態遷移確率を計算する必要はない．そこで，文献 [9] では，正常回路および直積機械のそれぞれにおいて状態遷移確率を求める前に，初期状態から到達可能な状態の列挙を行った後に，必要な状態に関する状態遷移確率のみを求めている．

しかし，直積機械の到達可能状態の列挙を単純に行うと多大な計算時間を必要とする．実際，文献 [9] の実験においては，30 状態程度の順序回路の解析に 1〜数時間を要しており，さらに計算時間の大部分が直積機械の到達可能状態の列挙に費やされていることが分かっている．

## 3. 非明示的列挙を用いた高速化

2 つの順序回路の等価性を判定する手法として，2 つの状態機械から直積機械を合成し，その直積機械の到達可能状態の列挙を行って，到達可能な状態のもとで 2 つの順序回路の出力が異なる場合があるかを調べる方法が知られている [11]. この手法の特徴は，状態集合や状態遷移関数をその特性関数 (characteristic function) の形で表わし，その特性関数を 2 分決定グラフで保持するというものである．本節ではこの手法で用いられている 2 分決定グラフを使った非明示的列挙手法を，順序回路のソフトエラー耐性評価に応用する手法について述べる．

### 3.1 集合と特性関数

状態の集合 $S = \{s_1, s_2, s_3, \ldots, s_n\}$ を考える．この要素の部分集合 $S_1 = \{s_{k_1}, s_{k_2}, \ldots, s_{k_m}\}, 1 \leqq k_i \leqq n$ は次のような関数 $\chi_{S_1} : S \to \{0, 1\}$ で表現することができる．

$$\chi_{S_1}(s) = \begin{cases} 1 & \text{if } s \in S_1 \\ 0 & \text{otherwise} \end{cases}$$

つまり，$s$ が $S_1$ に含まれる場合には $\chi_{S_1}(s) = 1$ となり，それ以外の場合には $\chi_{s_1}(s) = 0$ となる．このような関数 $\chi_{S_1}$ を集合 $S_1$ に対する特性関数 (characteristic function) と呼ぶ.

状態遷移関数 $\delta : S \times I \to S$ は定義域が $S \times I$，値域が $S$ の写像であるが，見方を変えると遷移を表わす，現状態 $s_p$，入力記号 $i$，次状態 $s_n$ の 3 つ組 $(s_p, i, s_n)$ の集合と見なすことができる．これはすなわち，$S \times I \times S$ を定義域とする 3 項関係であり，特性関数 $\chi_\delta : S \times I \times S \to \{0, 1\}$ を用いて表現することができる．

先の例のように状態集合 $S$ の各要素について 2 値符合化がされており，$k$ ビットの変数 $x_0, x_1, \ldots, x_{k-1}$ を用いて表わされ

るものとすると，状態遷移関数 $\delta : S \times I \to S$ は，$k$ 個の論理関数 $\delta_i : \{0,1\}^k \times I \to \{0,1\}$ の束で表わされる．次状態を表わす変数を $y_0, y_1, \ldots, y_{k-1}$ とすると，状態遷移を表わす特性関数は次式のようになる．

$$\chi_\delta = \prod_{i=0}^{k-1} (y_i \equiv \delta_i)$$

ここで，$a \equiv b$ は $a \cdot b + \overline{a} \cdot \overline{b}$ を表わす．

状態集合の場合と同様に，2 つの状態遷移関係の直積を表わす特性関数はもとの状態遷移関係の表わす特性関数の論理積となる．

### 3.2 2 分決定グラフを用いた到達可能状態の非明示的列挙手法

以上のように，状態集合，および状態遷移関数を 2 分決定グラフを用いて非明示的に表現できるので，それらを用いて到達可能状態の列挙を行う．図 1 に与えられた有限状態機械の到達可能状態を求めるアルゴリズムを示す．

```
    BFS_FSM(S, I, δ, λ, S_0) {
1:      Reached = From = New^0 = S_0;
2:      k = 0;
3:      do {
4:          k = k + 1;
5:          To = Img(δ, From);
6:          New^k = To − Reached;
7:          From = New^k;
8:          Reached = Reached ∪ New^k;
9:      } while (New^k ≠ 0);
    }
```

図 1　到達可能状態の列挙アルゴリズム

このアルゴリズムは有限状態機械の状態集合 $S$，入力記号 $I$，状態遷移関数 $\delta$，出力関数 $\lambda$，および初期状態 (集合)$S_0$ を入力として受取り，到達可能状態 Reached を計算する．アルゴリズム中で From は次に到達可能な状態を探すための遷移元の状態集合を表わす．$\mathrm{New}^k$ は $k$ ステップ目にはじめて到達可能となった状態集合を保持するために用いられる．1 行目においてこれらの変数はすべて $S_0$ で初期化される．その後，5 行目の Img 演算 (後述) を用いて From から到達可能な状態集合 To が求められる．To に含まれる状態のうち，Reached に含まれていないものが $\mathrm{New}^k$ の要素となる (6 行目)．続いて $\mathrm{New}^k$ を From とし，Reached を更新して今までの処理を繰り返す．このループは新たに到達可能となる状態が無くなるまで続けられる (9 行目)．

また，アルゴリズム中の関数 Img はスムージング (smoothing) 演算と呼ばれる論理演算を用いて以下のように定義される．

$$\mathrm{Img}(\delta, \mathrm{From}) = \exists S \exists I (\chi_{\mathrm{From}(s)} \wedge \chi_{\delta(s,i,t)})$$

$n$ 変数論理関数 $f(x_1, x_2, \ldots, x_n$ に対して，$x_i$ によるスムージング演算 $\exists x_i f$ は次のように定義される．

$$\exists x_i f = f_{x_i} + f_{\overline{x_i}}$$

ここで，$f_{x_i}(f_{\overline{x_i}})$ は $f$ の変数 $x_i$ に対して，1(0) を代入して得られる論理関数を表わす．複数の変数 $X^s = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ によるスムージングも次のように定義される(注1)．

$$\exists X^s f = \exists x_{i_1} (\exists x_{i_2} (\ldots (\exists x_{i_k} f)))$$

まず，$\chi_{\mathrm{From}(s)} \wedge \chi_{\delta(s,i,t)}$ は現状態が From である状態遷移 $\langle s, i, t \rangle$ の集合を表わす特性関数である．そこからスムージング演算によって $s$ と $i$ の変数を削除すると，次状態を表わす変数 $t$ のみからなる特性関数が得られる．これが $S$ の $\delta$ による像 (image) となる．

### 3.3 2 分決定グラフを用いた状態遷移確率計算

順序回路に対するソフトエラー耐性評価のためには到達可能状態の列挙につづいて，状態間の遷移確率を求める必要がある．残念ながら，遷移確率を表す行列は明示的に表現しなければならないため，最終的には 2 分決定グラフを用いて非明示的に表現された状態遷移関係から明示的に表現された遷移確率を計算する必要がある．

今，現状態は変数 $X = (x_1, x_2, \ldots, x_n)$ で符号化されているものとし，次状態も同様に変数 $Y = (y_1, y_2, \ldots, y_n)$ で符号化されているものとする．入力変数を $Z = (z_1, z_2, \ldots, z_m)$ として，状態遷移関係を表す特性関数を $\chi_\delta$ とする．今，現状態 $s_i$ と次状態 $s_j$ が与えられたとすると，$s_i \to s_j$ の遷移を引き起こす入力の集合は次式で与えられる．

$$\exists X \exists Y \chi_\delta \wedge \chi_{s_i} \wedge \chi_{s_j} \qquad (6)$$

ここで，$\chi_{s_i}$，$\chi_{s_j}$ はそれぞれ状態 $s_i$ と状態 $s_j$ を表す特性関数である．いま，外部入力の値が一様に分布していると仮定すると，(6) 式の結果の特性関数の真理値表密度がすなわち，状態 $s_i$ から状態 $s_j$ への遷移確率となる(注2)．このように，状態遷移を表す特性関数と $\chi_{delta}$ および，現状態 $s_i$ および次状態 $s_j$ を表す特性関数を与えることで状態遷移確率の計算を行うことは可能であるが，効率化のために以下の工夫を行う．

- 特性関数を 2 分決定グラフで表現する際に，入力を表す変数がもっとも下部 (定数節点に近い方) になるように変数順を決める．
- 以下のアルゴリズム calc_trans_prob を用いて状態遷移確率を求める (図 2).

図 2 中の index($v$) は 2 分決定グラフの節点 $v$ の変数インデックスを返す関数，child0($v$) および child1($v$) は $v$ の 0 枝および 1 枝の子供の節点を返す関数である．このアルゴリズムは 2 分決定グラフをたどりながら，状態変数に関する部分はたどった経路を配列 st_vec に記録してゆく，入力変数を表すレベルまでたどり着いたらその節点より下にある 2 分決定グラフの表し

(注1)：スムージング演算の定義から，複数の変数によるスムージングはその変数の適用順に依らず一意に定まる．

(注2)：入力が一様でない確率分布を持つ場合も，その分布が適切に表現されていれば (6) 式の特性関数から計算できる．

```
    calc_trans_prob(v) {
1:      if ( v = 0 ) {
2:          return;
        }
3:      i ← index(v);
4:      if ( i は状態を表す変数 ) {
5:          st_vec[i] ← 1;
6:          calc_trans_prob(child1(v));
7:          st_vec[i] ← 0;
8:          calc_trans_prob(child0(v));
        } else {
9:          d ←  の真理値表密度;
10:         cur ← st_vec から現状態を取り出す;
11:         next ← st_vec から次状態を取り出す;
12:         P_trans(cur, next) ← d;
        }
    }
```

図 2　状態遷移確率の計算

ている関数の真理値表密度を求める．この関数は (6) 式の表すものと同一である．実際の実装の詳細に関してはいくつか注意するべき点がある．

- 真理値表密度の計算に関しては同一の節点に対する計算結果は常に等しいので，演算結果テーブルを用いることで冗長な計算を省くことができる．
- 10 行めで唯一の現状態が得られるとは限らない．つまり，複数の現状態に対して，同一の入力で同一の次状態に遷移する場合には，そのようなことがおこる可能性がある．その場合，考えられる全ての現状態に対して同一の遷移確率を登録する．11 行目の次状態に関しては遷移が決定的である限り唯一の状態しかありえない．

このアルゴリズムは (6) 式の計算と同様の処理を行っているが，既存の 2 分決定グラフをたどるだけなので，AND 演算とスムージング演算を行う (6) 式の計算よりははるかに処理効率がよい．

### 3.4　直積機械の遷移関係の生成

順序回路のソフトエラー耐性評価のために直積機械を構成する場合，$\pi_f$ に関する遷移を考慮する必要がある．また，正常回路の状態と故障回路の状態が等しくなった場合には，以降，どのような入力に対しても常に等しい出力が得られる (直積機械としては常に 0 の出力が得られる) ので，そのような状態に対する遷移を調べる必要はない．そこで，以下のような手順で，直積機械の状態遷移関係を生成する．

（1） 外部入力を表す変数，外部出力を表す変数，正常回路の現状態および次状態を表す変数，故障回路の現状態および次状態を表す変数を用意する．

（2） $\pi_f$ を表す現状態および次状態用の 1 ビットの変数を用意する．この変数インデックスを $c_f$ および $n_f$ とする．通常の状態の場合には $c_f$ および $n_f$ ビットは 0 で符号化しておく．一方，$\pi_f$ を表す場合には通常の状態用の変数ビットをすべて 0 にして $c_f$ もしくは $n_f$ のみを 1 とする．

（3） 正常回路の状態遷移関係 $\chi_\delta$ を表す 2 分決定グラフを作る．

（4） $\chi_\delta$ に用いられている変数のうち，現状態および次状態を表す変数を故障回路用に付け替えた $\chi_{\delta'}$ を表す 2 分決定グラフを作る(注3)．

（5） $\chi_{cur\_normal} \leftarrow \overline{c_f}$ とする．
現状態の集合を通常の状態のみに制限する際に用いる．

（6） $\chi_{next\_normal} \leftarrow \overline{n_f}$ とする．
次状態の集合を通常の状態のみに制限する際に用いる．

（7） $\chi_{next\_error} \leftarrow n_f$ とする．
次状態の集合を $\pi_f$ のみに制限する際に用いる．

（8） $\phi \leftarrow \chi_\delta \wedge \chi_{\delta'} \wedge \chi_{cur\_normal}$ とする．
$\Phi$ は単純に入力のみを共有した直積機械の状態遷移関係である．

（9） 正常回路の出力関係の特性関数 $\chi_\lambda$ を表す 2 分決定グラフを作る．

（10） $\chi_\lambda$ から状態変数を故障回路ように付け替えた $\chi_{\lambda'}$ を作る．

（11） $\chi_{equiv} \leftarrow \exists O(\chi_\lambda \wedge \chi_{\lambda'})$ とする．
つまり，正常回路と故障回路の出力が等しくなる入力と現状態の集合を表す特性関数である．ただし，$O$ は出力を表す変数の集合を表す．

（12） 正常回路と故障回路の次状態が等しくなる入力と現状態の集合を表す特性関数を $\chi_{ident}$ とする．

（13） $\psi \leftarrow \phi \wedge \chi_{eq} \wedge \overline{\chi_{ident}} \wedge \chi_{next\_normal}$
正常回路と故障回路の出力は等しいが，次状態が等しくないような遷移のみを表す特性関数

（14） $\xi \leftarrow \exists N(\phi \wedge \overline{\chi_{eq}}) \wedge \chi_{next\_error}$
つまり，$\pi_f$ への遷移のみを表す特性関数である．ただし，$N$ は次状態を表す変数の集合を表す．

（15） $\gamma \leftarrow \psi \vee \xi$ とすると，$\gamma$ が求める直積機械の状態遷移関係となる．

## 4. 実験結果

上記手法を C++ 言語で実装しベンチマーク回路に対して実験を行った．実験内容は文献 [9] に示したものとほぼ同じものであり，実験結果も計算時間を除いてまったく等しいものとなっている．今回の実験の目的は 2 分決定グラフを用いた非明示的列挙手法の効果を見ることにある．

実験に用いた計算機の CPU は Intel Xeon 3.3GHz である．ベンチマーク回路として MCNC ベンチマークセット [12] の有限状態機械 (fsmex) を用いた．ベンチマークの各データは状態符号化プログラム JEDI [13] および one-hot 符号化により符号化したものを論理合成ツール SIS [14] を用いて多段化している．文献 [9] で述べているように，SIS による論理合成処理で重要なことは，未使用な符号に対する状態遷移を (適当に) 作り出すこ

(注3)：正常回路の状態変数のインデックスに適当なオフセットを付加したものを故障回路の状態変数のインデックスとしておけば，変数の付け替えは同形で状態変数のインデックスのみがオフセット分ずれた 2 分決定グラフを生成すれば良い．

とにある．本来，回路が正常な動作しかしない場合には，そのような未使用な符号に対する状態遷移は起こらないが，ソフトエラーによってフリップフロップの値がランダムに反転する仮定のもとでは未使用な符号に対する遷移が起こる可能性がある．

JEDI は最短符号長で符号化を行うため未使用な符号は少ないが，one-hot 符号化では状態数に対して指数乗の未使用符号が存在する．そのため，ソフトエラー耐性の評価という目的ではもっとも困難な符号割り当ての一つと考えることができる．

実験結果を表 1 に示す．

表 1　実 験 結 果

| 回路名 | 状態数 | FF 数 | A | B | C |
|---|---|---|---|---|---|
| ex5-JEDI | 9 | 4 | 62 | 167 | 0.01 |
| ex5-ONEHOT | 9 | 9 | 90 | 240 | 0.02 |
| ex7-JEDI | 10 | 4 | 66 | 183 | 0.01 |
| ex7-ONEHOT | 10 | 10 | 110 | 297 | 0.02 |
| bbara-JEDI | 10 | 4 | 59 | 146 | 0.01 |
| bbara-ONEHOT | 10 | 10 | 100 | 370 | 0.02 |
| ex3-JEDI | 10 | 4 | 82 | 234 | 0.02 |
| ex3-ONEHOT | 10 | 10 | 110 | 308 | 0.02 |
| opus-JEDI | 10 | 4 | 41 | 43 | 0.01 |
| opus-ONEHOT | 10 | 10 | 100 | 120 | 0.02 |
| ex4-JEDI | 14 | 4 | 66 | 76 | 0.02 |
| ex4-ONEHOT | 14 | 14 | 210 | 255 | 0.06 |
| dk512-JEDI | 15 | 4 | 83 | 113 | 0.01 |
| dk512-ONEHOT | 15 | 15 | 225 | 360 | 0.07 |
| bbsse-JEDI | 16 | 4 | 84 | 126 | 0.01 |
| bbsse-ONEHOT | 16 | 16 | 272 | 442 | 0.12 |
| cse-JEDI | 16 | 4 | 122 | 426 | 0.03 |
| cse-ONEHOT | 16 | 16 | 272 | 799 | 0.14 |
| keyb-JEDI | 19 | 5 | 153 | 350 | 0.02 |
| keyb-ONEHOT | 19 | 19 | 361 | 836 | 0.30 |
| s1-JEDI | 20 | 5 | 100 | 106 | 0.02 |
| s1-ONEHOT | 20 | 20 | 400 | 500 | 0.35 |
| ex1-JEDI | 20 | 5 | 117 | 142 | 0.04 |
| ex1-ONEHOT | 20 | 20 | 420 | 483 | 0.35 |
| styr-JEDI | 30 | 5 | 206 | 458 | 0.06 |
| styr-ONEHOT | 30 | 30 | 930 | 2015 | 2.88 |
| sand-JEDI | 32 | 5 | 260 | 448 | 0.20 |
| sand-ONEHOT | 32 | 32 | 1056 | 1254 | 4.14 |

表 1 において，A の列は直積機械の状態数を表している．B の列は直積機械の状態遷移確率の非ゼロの要素数を表している．C の列は計算時間を秒単位で表したものである．

文献 [9] の実験では sand-JEDI で約 3400 秒，sand-ONEHOT で約 10,700 秒かかっていることと比較して極めて高速に処理を行っている．また，多くの場合，直積機械の状態数と状態遷移確率の非ゼロ要素数の比は高々3〜4 倍程度に収まっており，状態遷移確率の行列が非常に疎であることがわかる．

## 5. おわりに

本稿では，順序回路のソフトエラー耐性を評価する非明示的列挙手法について述べた．状態数が数十程度の例題に対しては極めて高速に処理を行っており，手法の有効性が確認できた．実際にはフリップフロップ数が数千〜数十万程度の回路を扱う必要があり，本手法をそのまま適用することは不可能と思われるので，大規模な回路に適用可能な近似手法の開発が今後の課題である．

## 謝　辞

本研究の一部は，独立行政法人科学技術振興機構 CREST-DVLSI の研究補助金および，科学研究費補助金の助成によるものである．

## 文　献

[1] T. Calin, M. Nicolaidis and R. Velazco: “Upset hardened memory design for submicron cmos technology”, IEEE Transactions on Nuclear Science, **Vol. 43**, No. 6 (1996).

[2] S. Mitra, M. Zhang, N. Seifert, T. Mak and K. S. Kim: “Built-in soft error resilience for robust system design”, ICICDT ’07, pp. 1–6 (2007).

[3] S. Krishnaswamy, S. Plaza, I. Markov and J. Hayes: “Enhancing design robustness with reliability-aware resynthesis and logic simulation”, Proceedings of IEEE International Conference on Computer Aided Design, pp. 149–154 (2007).

[4] Y. Lin and L. He: “Device and architecture concurrent optimization for fpga transient soft error rage”, Proceedings of IEEE International Conference on Computer Aided Design, pp. 194–198 (2007).

[5] B. Zhang, W. Wang and M. Orshansky: “Faser: Fast analysis of soft error susceptibility for cell-based designs”, ISQED, pp. 755–760 (2006).

[6] M. Zhang and N. Shanbhag: “A soft error rate analysis (sera) methodology”, Proceedings of IEEE International Conference on Computer Aided Design, pp. 111–118 (2004).

[7] N. Miskov-Zivanov and D. Marculescu: “Mars-c: Modeling and reduction of soft erros in combinational circuits”, Proceedings of 43nd IEEE/ACM Design Automation Conference, pp. 767–772 (2006).

[8] N. Miskov-Zivanov and D. Marculescu: “Mars-s: Modeling and reduction of soft errors in sequential circuits”, ISQED ’07, pp. 893–898 (2007).

[9] 赤峰, 吉村, 松永：“マルコフモデルを用いた順序回路のソフトエラー耐性評価手法”, DA シンポジウム 2009 情報処理学会 (2009).

[10] R. E. Bryant: “Graph-based algorithms for boolean function manipulation”, IEEE Trans. Comput., **C-35**, no. 8, pp. 677–691 (1986).

[11] O. Coudert and J. C. Madre: “A unified framework for the formal verification”, Proceedings of International Conference on Computer Aided Design, pp. 126–129 (1990).

[12] S. Yang: “Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0” (1991).

[13] B. Lin and A. R. Newton: “Synthesis of multiple level logic from symbolic highlevel description languages”, Proceedings of IFIP International Conference on VLSI, pp. 187–196 (1989).

[14] E. M. Sentovich: “Sis: A system for sequential circuit synthesis”, Technical report, UC Berkeley (1992).