順序回路のソフトエラー耐性評価手法の状態数削減による高速化

赤峰, 悠介
九州大学大学院システム情報科学府

吉村, 正義
九州大学大学院システム情報科学研究院

松永, 裕介
九州大学大学院システム情報科学研究院

出版情報：電子情報通信学会技術研究報告. 109 (462), pp.163-168, 2010-03. 電子情報通信学会
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耐性評価手法の状態数削減による高速化

赤峰　悠介†　　吉村　正義††　　松永　裕介††

† 九州大学 大学院システム情報科学府
†† 九州大学 大学院システム情報科学研究院
〒 819-0395 福岡県福岡市西区元岡 744
E-mail: †akamine@c.csce.kyushu-u.ac.jp, ††{yosimura,matsunaga}@ait.kyushu-u.ac.jp

あらまし　ソフトエラー耐性を考慮した論理回路の設計では，ソフトエラー耐性評価手法が必要となる．著者らは，順序回路を対象とした評価手法として，マルコフモデルを用い状態遷移の振る舞いを厳密に解析するものを提案している．この手法は計算時間の点で問題があり，その大部分を占める処理の一つが，状態数を元数とする連立方程式の計算である．そこで，本稿では，計算を複数回に分けて行うことで一度に扱う状態数を削減し，また，計算に必要のない状態を削除することによる高速化手法を提案する．

キーワード　ソフトエラー，順序回路，有限状態機械，吸収確率

An Acceleration of Soft Error Torelance Estimation Method for Sequential Circuits by Reducing the Number of States

Yusuke AKAMINE†, Masayoshi YOSHIMURA††, and Yusuke MATSUNAGA††

† Graduate School of Information Science and Electrical Engineering, Kyushu University
†† Faculty of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishiku, Fukuoka 819-0395 JAPAN
E-mail: †akamine@c.csce.kyushu-u.ac.jp, ††{yosimura,matsunaga}@ait.kyushu-u.ac.jp

Abstract　Soft error tolerance estimation method is necessary for the soft error aware logic design. We proposed an estimation method with Markov model for sequential circuits, which can analyze the behavior of the state transition strictly. This method has an issue that it is difficult to apply for large scale circuits, and solving simultaneous equations is one of bottleneck processes. In this paper, we propose acceleration methods by dividing simultaneous equations into small problems and removing unnecessary unknowns.

Key words　soft error, sequential circuit, finite state machine, absorption probability

1. はじめに

近年，LSI の信頼性においてソフトエラーが問題として認識されつつある．ソフトエラーとは，トランジスタへの中性子の衝突に起因し発生する電荷により，記憶素子に保持している値の反転や，論理ゲートの出力値の一時的な反転が起きることである．値の反転に必要な最小の電荷量 (臨界電荷量) は，半導体デバイスの微細化に伴い減少しており，中性子が衝突した際にソフトエラーが発生する確率は増大している．これまで，メモリ回路におけるソフトエラーに関しては多くの解析がなされ，誤り訂正符号 [1] などの比較的低コストで施せる有効な対策が提案されている．一方，論理回路に関しては，メモリ回路と比較して回路構造に規則性がないことなどから，効果的な対策や解析手法は確立されていない [2]．そのため，論理回路のソフトエラーが無視できない問題となる可能性がある [3].

論理回路の設計時にソフトエラー耐性を考慮する場合，回路が所望のソフトエラー耐性を持つか評価するために，ソフトエラー耐性の評価手法が必要である．ここで，ソフトエラーが発生した回路の出力における振る舞いが正常な回路の振る舞いと異なるとき，回路はソフトエラーによって誤動作したといえる．よって，ソフトエラーが発生し外部出力に伝搬する確率をソフトエラー耐性の評価指標として用いるのが適当である．論理回路に対するソフトエラー耐性評価手法はいくつか提案されているが [4] [5] [6] [7] [8] [9]，それらの手法の多くは，ソフトエラーが発生したクロックサイクルにおける組み合わせ回路部の解析を行うものである．しかしながら，順序回路においては，ソフ

トエラーが発生したとしても，その影響がすぐに外部出力に現れるとは限らず，何度かの状態遷移を経た後に顕在化する可能性があり，組み合わせ回路部の解析だけでは不十分であると考えられる．文献 [9] の手法は，ソフトエラー発生後の状態遷移の振る舞いを解析するものであるが，計算の結果得られる値がソフトエラー耐性の評価指標としては不適切なものである．

これに対し著者らは，順序回路におけるソフトエラー発生後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) としてモデル化し，ソフトエラーが発生し外部出力に伝搬する確率を計算する手法を提案している [10]．この手法は，与えられた入力値の確率分布の仮定のもとでは厳密な計算を行うものであるが，計算時間の点から実用的とはいえない．この手法では，マルコフモデル上にエラーが外部出力に伝搬したことを表わす仮想的な状態を設け，その状態に到達する確率 (吸収確率と呼ぶ) を求めることでソフトエラーが発生し外部出力に伝搬する確率を求める．吸収確率を求める際の連立方程式の計算が，実行時間の大部分を占める処理の一つとなっている．

そこで本稿では，吸収確率の計算における連立方程式の元数に着目した二つの高速化手法を提案する．一つ目の手法は，連立方程式の計算を複数回に分けて行うことで，一度の計算で扱う状態数を削減することを狙いとしている．また，二つ目の手法は，本来吸収確率を求める必要のない状態を削除することにより，連立方程式の元数を削減するものである．実験により，高速化手法を用いることで，単純な実装と比較して最大 10 倍程度の高速化を確認した．

本稿の構成を以下に示す．まず，2 節でマルコフモデルを用いたソフトエラー耐性評価手法を説明する．次に，3 節で提案する高速化手法について説明する．4 節で高速化手法の評価実験の結果を示し，5 節で本稿をまとめる．

2. マルコフモデルを用いたソフトエラー耐性評価手法 [10]

本節では，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を説明する．なお，以降では，本手法を厳密手法と呼ぶ．

2.1 順序回路と回路対

順序回路とは，有限状態機械を論理素子およびフリップフリップ (以降，FF と略す) などの記憶素子を用いて実現したもので，出力値が過去の入力系列に依存するという特徴を持つ．過去の入力系列に依存した「状態」を記憶素子の値として保持することで，過去の入力系列を出力に反映させる．

順序回路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 (以降，正常回路と呼ぶ) と，ソフトエラーの発生を仮定した回路 (以降，故障回路と呼ぶ) に同一の入力を与え，出力が異なるかを調べればよい．このような二つの回路を並べた回路を，本稿では回路対と呼ぶ．回路対の例を，図 1 に示す．図 1 において，正常回路は評価対象の回路である．故障回路は，回路構成は正常回路と同様であるが，FF でのソフトエラーの発生を仮定した回路である．故障回路

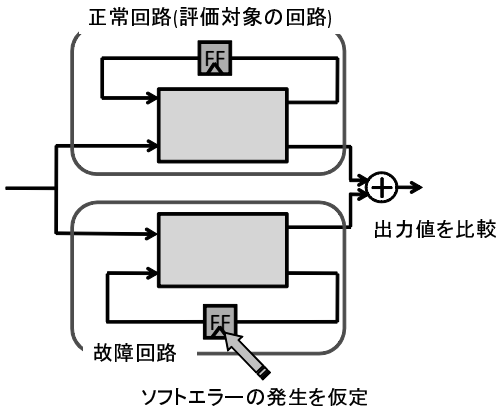

図 1 回 路 対

はソフトエラー発生の直前までは正常回路と等しい振る舞いをする．正常回路と故障回路の出力値が異なるとき，エラーは外部出力に伝搬したといえる．

2.2 マルコフモデル

以上のように，回路対を用いることで，正常な回路の振る舞いとエラーを含む振る舞いの比較を行うことができる．本来，順序回路の振る舞いは入力と状態によって一意に定まる，決定的なものである．ただし，回路の動作と中性子が衝突するタイミングには関連がないため，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，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ので，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，回路対の状態遷移の振る舞いを，確率的に状態遷移する状態機械であるマルコフモデルであると考えることができる．

順序回路において，ソフトエラーが発生し外部出力に伝搬する確率を求めるためには，外部出力へのエラーの伝搬を考慮しつつ状態遷移の振る舞いを解析する必要がある．そこで，厳密手法では，回路対の状態遷移を表わすマルコフモデル上に，外部出力にエラーが伝搬したことを表わす仮想的な状態を設け，状態遷移の振る舞いを解析する．このように状態遷移の振る舞いを表したとき，ソフトエラーが発生し外部出力に伝搬する確率は，この仮想的な状態に到達する確率を求めることで計算できる．

外部出力にエラーが伝搬した以降の状態を，failure 状態と呼ぶ．本手法の目的は，ソフトエラーが発生し外部出力に伝搬する確率を求めることであるから，エラーが伝搬した以降の振る舞いに関して解析する必要はない．よって，failure 状態は，自分自身に確率 1 で遷移する状態とする．このような状態に到達すると，以降は永久にその状態にとどまり続けることになる．ここで，確率 1 で自分自身に遷移する状態を吸収状態と呼び，吸収状態に到達することを，その状態に吸収されると表現する．また，一旦正常回路と故障回路の FF の値が等しい状態に遷移すると，それ以降の時刻で外部出力の値が誤る可能性はない．よって，正常回路と故障回路の値が等しい状態に関しても，以降の振る舞いを解析する必要はなく，吸収状態とみなせ

る．正常回路と故障回路の FF の値が等しい状態を masked 状態と呼ぶ．

以上より，ソフトエラー発生後の状態を以下のように三つに分類することができる．

failure 状態
エラーが外部出力まで伝搬した以後の状態

masked 状態
正常回路と故障回路の FF の値が等しい状態

一時状態
正常回路と故障回路の FF の値は異なるが，エラーは外部出力に伝搬していない状態

図 2 は，ソフトエラー発生後の状態遷移をマルコフモデルで表わした例である．図 2 において，一時状態は正常回路と故障

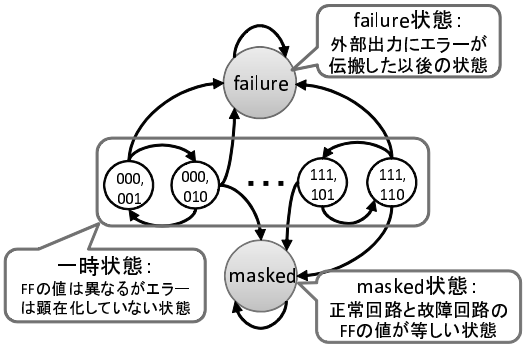

図 2　マルコフモデルと状態の分類

回路の FF の値のペアで表わしている．

結局，順序回路においてソフトエラーが発生し外部出力に伝搬する確率を求めることは，上記のマルコフモデル上で，failure 状態に吸収される確率を求めることに他ならない．

2.3　ソフトエラーが発生し外部出力に伝搬する確率の計算

ソフトエラーが発生し外部出力に伝搬する確率を求めることは，failure 状態 π_f に吸収される確率を求めることと等しい．ソフトエラーが発生し外部出力に伝搬する確率 $P_{abs}(\pi_f)$ は，以下の式で表わされる．

$$P_{abs}(\pi_f) = \sum_{\pi \in \Pi_{init}} P_{init}(\pi) \cdot q_{\pi\pi_f} \tag{1}$$

式 (1) において，$P_{init}(\pi)$ は状態 π の初期状態確率であり，ソフトエラー発生直後の状態 (初期状態と呼ぶ) が π である確率である．Π_{init} は初期状態の集合である．また，$q_{\pi\pi_f}$ は，状態 π を出発して π_f に吸収される確率であり，吸収確率と呼ぶ．以下に，初期状態確率及び，吸収確率の計算方法を示す．

2.3.1　初期状態確率の計算

状態 π の初期状態確率 $P_{init}(\pi)$ は，以下の式で表わされる．

$$P_{init}(\pi) = \sum_{s \in S} P_{steady}(s) \cdot e_{s\pi} \tag{2}$$

ここで，正常回路の状態の集合を S とする．$s \in S$ の定常確率 $P_{steady}(s)$ は，正常回路のある時刻の状態が s である確率である．また，$e_{s\pi}$ は，ソフトエラーにより状態 s から状態 $\pi \in \Pi$ に遷移する確率を表す．

正常回路の状態 s の定常確率 $P_{steady}(s)$ は以下のように計算できる．まず，正常回路において，状態 s_i から s_j に遷移する確率を $r_{s_i s_j}$ とする．このとき，s_j の定常確率は以下の式で表わされる．

$$P_{steady}(s_j) = \sum_{s_i \in S} P_{steady}(s_i) \cdot r_{s_i s_j} \tag{3}$$

また，全状態の定常確率の総和は 1 であるので，以下の式が成り立つ．

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

式 (3),(4) により，各状態間の遷移確率が既知であれば，定常確率を未知数とする連立方程式を解くことにより，定常確率は計算可能である．

一方，$e_{s\pi}$ は，FF 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 FF に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

回路対の状態の集合を Π とする．$\pi_i \in \Pi$ を出発し failure 状態 π_f へ吸収される確率 $q_{\pi_i\pi_f}$ は以下のように表される．

$$q_{\pi_i\pi_f} = p_{\pi_i\pi f} + \sum_{\pi_j \in \Pi_{tmp}} p_{\pi_i\pi_j} q_{\pi_j\pi_f} \tag{5}$$

Π_{tmp} は一時状態の集合である．各状態間の遷移確率が既知であれば，$q_{\pi_i\pi_f}$ を未知数とする連立方程式を解くことで吸収確率を計算できる．

2.4　処理の流れと問題点

本節では，厳密手法の処理の流れを説明する．厳密手法の処理の流れを，図 3 に示す．まず，正常回路の定常確率を求める

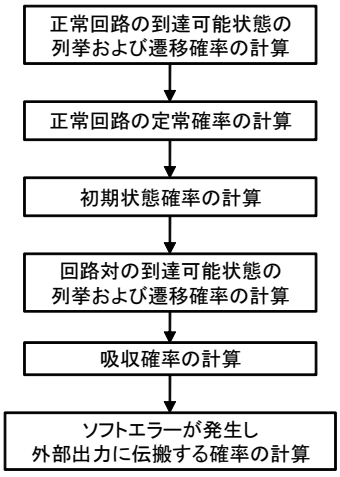

図 3　処理の流れ

ために，正常回路の遷移確率を求める必要がある．遷移確率を計算する方法としては論理シミュレーションなどが考えられる．到達不可能な状態の定常確率を求める必要はないため，遷移確

率の計算は到達可能な状態に対してのみ行う．次に，正常回路の遷移確率から，定常確率を計算する．定常確率は，連立方程式を解くことで求める．次に，求めた定常確率から，初期状態確率を求める．本手法では，$e_{s\pi}$ は与えられるものとし，正常回路の各状態の定常確率との積をとることで初期状態確率を求める．次に，吸収確率計算のために，回路対の遷移確率を求める．正常回路の場合と同様，論理シミュレーションなどの方法が考えられる．次に，回路対の遷移確率から，吸収確率を求める．吸収確率は，連立方程式を解くことで求める．最後に，各初期状態の初期状態確率と吸収確率の積の総和を計算することで，ソフトエラーが発生し外部出力に伝搬する確率を求める．

本手法の問題点は，多大な実行時間を要するという点であり，特に，回路対の遷移確率の計算と，吸収確率の計算が実行時間の大部分を占める処理である．FF 数を k とすると，最悪の場合回路対の状態数は 2^{2k} である．回路対の遷移確率を論理シミュレーションにより求める場合，入力パターン数を I とすると，論理シミュレーションの実行回数は $2^{2k} \times I$ 回である．また，吸収確率の計算においては連立方程式を解く．連立方程式の解法の一つであるガウスの消去法を用いる場合，実行時間は元数の 3 乗に比例することが知られている．元数は回路対の状態数であるから，吸収確率の計算は，最悪の場合 $(2^{2k})^3 = 2^{6k}$ に比例した時間を要する．

3. 高速化手法

本節では，二つの高速化手法を提案する．厳密手法においては，吸収確率を求めるための連立方程式の計算が実行時間の大部分を占める処理の一つである．提案する二つの高速化手法は，いずれも，連立方程式の元の数に着目し，高速化を狙うものである．まず，一つ目の手法は，連立方程式の計算を複数回に分けて行うことで，一度の計算で扱う状態数を削減することを狙いとしている．また，二つ目の手法は，本来吸収確率を計算する必要のない状態を削除することで連立方程式の元数を削減するものである．

3.1 状態集合の分割

2.3 節で示したように，ソフトエラーが発生し外部出力に伝搬する確率は，ある初期状態の初期状態確率と，その初期状態からの吸収確率の積の総和をとることにより求められる．初期状態からの吸収確率は，連立方程式を解くことで求められるが，この際に，取りうる全ての状態を元として持つ必要はない．初期状態 π_i からの吸収確率を求めるためには，π_i からの到達可能状態のみを元とする連立方程式を解けばよい．当然，状態集合ごとの連立方程式の元数は，全到達可能状態数以下となる．連立方程式を解く回数自体は増えるものの，個々の実行時間が短ければ，全体の実行時間を短縮できる可能性がある．

図 4 に，状態集合の分割の例を示す．図 4 において，π_1 および π_2 は初期状態である．一度の連立方程式の計算で π_1, π_2 の吸収確率を求める場合，元数は 9 となる．π_1 の吸収確率を求めるためには，到達可能状態の各状態の吸収確率を元とする連立方程式を解けばよい．π_1 からの到達可能状態の集合は，$\{\pi_1, \pi_3, \pi_4, \pi_6, \pi_7, \pi_8\}$ であるから，元数は 6 となる．π_2 に関しても，同様に元数は 6 である．

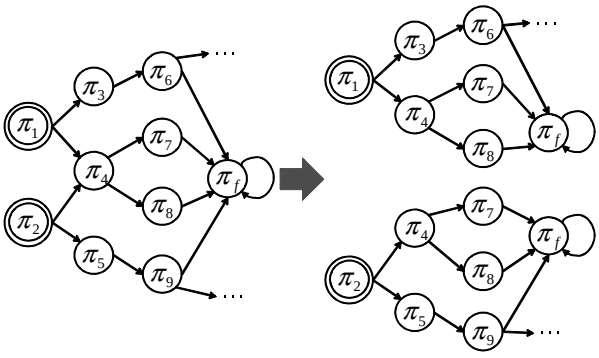

図 4　状態集合の分割の例

状態集合を分割して到達可能状態を列挙する場合，異なる状態集合間に同じ状態が現れる可能性がある．他の状態集合に対する計算において既に表れている状態に関しては，その状態からの遷移確率，到達可能状態の集合，また吸収確率は既に得られているため，再度計算する必要はない．

ある初期状態からの吸収確率の計算に必要な状態集合は，その初期状態からの到達可能状態の集合である．ただし，個々の初期状態毎の到達可能状態の集合に対し連立方程式を解く場合，当然，初期状態の数だけの連立方程式を解く必要がある．連立方程式を解く回数自体が増えると，高速化の効果が小さくなる可能性があるため，適切な状態数に分割する必要があると考えられる．

3.2 pre-failure 状態の削除

pre-failure 状態とは，一時状態のうち，確率 1 で failure 状態へ遷移する状態のことである．すなわち，pre-failure 状態の次状態は failure 状態のみである．pre-failure 状態から一時状態への遷移確率は 0 であることと，式 (5) から，ある pre-failure 状態 π_{pf} からの吸収確率は以下の式で表わされる．

$$q_{\pi_{pf}\pi_f} = p_{\pi_{pf}\pi_f} = 1 \tag{6}$$

π_{pf} から π_f への遷移確率 $p_{\pi_{pf}\pi_f}$ は，pre-failure 状態の定義から明らかに 1 である．

吸収確率を求める連立方程式においては，当然，既知である pre-failure 状態からの吸収確率を元として持つ必要はない．また，pre-failure 状態への遷移は，failure 状態への遷移と同様のものとして扱える．pre-failure 状態の集合を $\Pi_{pf} \subseteqq \Pi_{tmp}$ とすると，ある状態 π の吸収確率は以下のように表わせる．

$$q_{\pi_i\pi_f} = p_{\pi_i\pi_f} + \sum_{\pi_k \in \Pi_{pf}} p_{\pi_i\pi_k} + \sum_{\pi_j \in \Pi_{tmp} - \Pi_{pf}} p_{\pi_i\pi_j} q_{\pi_j\pi_f} \tag{7}$$

以上のように，pre-failure 状態の吸収確率は既知であるため連立方程式の元に含める必要はない．よって，吸収確率を求める連立方程式の元数は，pre-failure 状態の集合に含まれる状態数だけ削減できることになり，単純な実装と比較して高速化が期待できる．

表 1　実 行 時 間

benchmark	#input	#FF	time[s]							
			base		partition		pre-failure		partition+pre-failure	
			total	abs prob	total	abs prob	total	abs prob	total	abs prob
s27	4	3	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
s820	18	5	93.44	<0.01	92.16	<0.01	93.27	<0.01	92.37	<0.01
s386	7	5	0.03	<0.01	0.02	<0.01	0.02	<0.01	0.03	<0.01
s510	19	6	416.80	<0.01	416.93	<0.01	414.34	<0.01	410.24	<0.01
s1488	8	6	0.28	<0.01	0.28	<0.01	0.33	<0.01	0.34	<0.01
s208.1	10	8	7.98	0.03	7.87	<0.01	7.98	0.03	7.85	<0.01
mm4a	7	12	4.77	0.30	4.60	0.06	4.80	0.30	4.59	0.06
s298	3	14	0.49	0.11	0.46	0.03	0.44	0.03	0.45	0.02
s344	9	15	12646.09	11522.14	5865.96	4750.53	1111.54	3.79	1109.52	2.16
s1196	14	18	4076.59	372.44	3863.48	0.91	3873.84	19.64	3865.78	0.86
s382	3	21	89398.83	87834.66	65208.42	64923.48	63091.25	61536.03	42788.79	41180.84

表 2　連立方程式の元数

benchmark	#input	#FF	#unknown					
			base	partition		pre-failure	partition+pre-failure	
				max	sum		max	sum
s27	4	3	18	6	18	18	6	18
s820	18	5	186	94	280	108	79	196
s386	7	5	91	30	144	77	25	126
s510	19	6	452	97	486	97	24	100
s1488	8	6	350	72	374	63	16	71
s208.1	10	8	3840	512	3840	3840	512	3840
mm4a	7	12	10577	1328	12661	10577	1328	12661
s298	3	14	6332	1645	7730	3192	1102	4144
s344	9	15	678160	266901	806399	33145	14125	41590
s1196	14	18	47656	4422	51681	42346	4422	45958
s382	3	21	1502857	777881	2217277	1150087	642249	1738535

4. 実　　験

本節で示した二つの高速化手法の効果を確認するための実験を行った．各手法は C++言語により実装した．実験に用いた計算機は，CPU が Intel Xeon 3.3GHz，メモリが 32GB である．ベンチマーク回路として，ISCAS'89 ベンチマークセットおよび MCNC ベンチマークセット [11] の順序回路の一部を用いた．なお，ソフトエラーは FF でのみ発生し，また，failure 状態，masked 状態に吸収されるまでの間に再び発生することはないと仮定して実験を行った．なお，遷移確率の計算は全入力パターンを用いた論理シミュレーションにより行い，各入力パターンは一様に現れると仮定した．

実験においては，以下の 4 通りの実装で，ソフトエラーが発生し外部出力に伝搬する確率を計算し，実行時間および連立方程式の元数の比較を行う．

（1） ナイーブな実装 (base)
（2） 状態集合の分割を用いた実装 (partition)
（3） pre-failure 状態の削除を用いた実装 (pre-failure)
（4） 状態集合の分割および pre-failure 状態の削除を用いた実装 (partition+pre-failure)

なお，状態集合の分割においては，最も細かく分割する場合，各初期状態毎の到達可能状態の集合に分割できる．本実験においては，ソフトエラーの発生を仮定する FF 毎に初期状態の集合を分け，それぞれの到達可能状態を分割後の集合とした．

実験結果として，各実装の実行時間を表 1 に，連立方程式の元数を表 2 にそれぞれ示す．表 1，2 において，benchmark の列は，ベンチマーク回路を表わす．また，#input，#FF はそれぞれ回路の入力数，FF 数を表わす．time の列は各実装の実行時間を表わし，単位は秒である．また，実行時間は，total の列に処理全体の実行時間を示し，abs prob の列に吸収確率計算の実行時間を示す．なお，< 0.01 は，実行時間が 0.01 秒未満であったことを表わす．また，表 2 において，#unknown は，連立方程式の元数を表わす．partition，partition+pre-failure の列において，max の列は，分割された状態集合のうち最大の元数であり，sum の列は，各状態集合の状態数の総和を表わしている．なお，表 1，2 は，FF 数の昇順でソートしている．

まず，比較的規模の小さいベンチマークにおいては，高速化手法の効果はほとんど見られない．これらの回路は，FF 数が少なく，到達可能状態も少ないために，連立方程式の処理が実行時間に占める割合が小さいためであると考えられる．

次に，s344 に着目する．ナイーブな実装では 12600 秒程度の実行時間を要しているのに対し，状態集合の分割を用いた実装

では 5900 秒程度，また，pre-failure 状態の削除を行った場合，状態集合の分割および pre-failure 状態の削除を行った場合ともに，1100 秒程度に実行時間を削減している．特に pre-failure 状態の削除は高速化の効果が大きく，10 倍ほどの高速化となっている．ナイーブな実装において 11000 秒ほどを要している吸収確率の計算時間を，数秒程度にまで短縮している．連立方程式の元数を見ると，ナイーブな実装では元数は 690000 程度であるが，pre-failure 状態の削除により，30000 程度まで削減できており，非常に効果が大きいことがわかる．s344 は，一時状態のうち pre-failure 状態の占める割合が非常に高いといえる．

表中で最も FF 数が多い s382 においては，ナイーブな実装で 89000 秒程度，状態集合の分割を用いた実装で 65000 秒程度，pre-failure 状態の削除を用いた実装で 63000 秒程度，二つの高速化手法をともに用いた実装では 43000 秒程度となっている．状態集合の分割，pre-failure 状態の削除ともに単独で同程度の高速化となっており，また，同時に用いた場合は，さらなる高速化の効果が得られている．連立方程式の元数を見ると，ナイーブな実装が 1500000 程度であるのに対し，状態集合の分割では最大で 780000 程度，また，pre-failure 状態の削除では，1150000 程度に削減している．s344 ほどではないが，元数を削減しており，実行時間の削減につながったといえる．

また，状態集合の分割を用いた際の，元数の総和に着目する．いずれの回路においても，総和は，もともとの状態数と比較してそれほど大きな値になっていないことがわかる．これはすなわち，状態集合を分割しても，重複した状態に対する処理を行う機会はそれほど多くないということを示している．

以上より，実行時間の大部分を連立方程式の計算が占めている回路においては，状態集合の分割，pre-failure 状態の削除ともに効果的であるといえる．また，本実験では，状態集合の分割を，初期状態の集合をソフトエラーの発生を仮定する FF 毎に分割することにより行ったが，分割の最小単位は個々の初期状態からの到達可能状態の集合であるので，さらに細かく分割することが可能である．s382 では状態集合の分割を用いた場合の元数の最大値は 780000 程であり，分割の単位を細かくすることで，さらなる元数の削減および実行時間の削減が期待できる．また，s1196 のように到達可能状態の列挙が実行時間の大部分を占める回路では，元数は削減されているが全体の実行時間はさほど削減されていない．

5. おわりに

本稿では，順序回路のソフトエラー耐性評価の高速化手法として，状態集合の分割と pre-failure 状態の削除を提案した．状態集合の分割は，連立方程式の計算を複数回に分けて行うことで一度に扱う状態数を削減する手法であり，また，pre-failure 状態の削除は，計算に必要のない状態を削除する手法である．高速化手法の効果により，ナイーブな実装と比較して最大 10 倍ほどの高速化を確認した．また，状態集合の分割において，より細かい単位で分割することにより，さらなる高速化の効果が期待できる．しかしながら，吸収確率の計算以外の処理がボトルネックとなっている回路に対しては，全体の実行時間はさほど削減できない．よって，実行時間のボトルネックの一つである遷移確率の計算の高速化が今後の課題として挙げられる．

謝　辞

本研究の一部は，科学技術振興機構 (JST) の戦略的創造研究推進事業 (CREST)「統合的高信頼化設計のためのモデル化と検出・訂正・回復技術」の支援によるものである．

文　献

[1] 情報理論とその応用学会編．“符号理論とその応用”．培風館，2003.

[2] 上村大樹, 戸坂清春, 芹沢芳夫, 岡秀樹, 佐藤成生．“中性子ソフトエラーシミュレーションの新展開”．信学技報, ICD, Vol. 105, No. 2, pp. 37–42, 2005.

[3] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, L. Alvisi. “Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic”. In *Proc. of DSN*, 2002.

[4] Smita Krishnaswamy, Stephen M. Plaza, Igor L. Markov, John P. Hayes. “Enhancing design robustness with reliability-aware resynthesis and logic simulation”. In *Proc. of ICCAD*, pp. 149–154, 2007.

[5] Yan Lin and Lei He. “Device and Architecture Concurrent Optimization for FPGA Transient Soft Error Rate”. In *Proc. of ICCAD*, pp. 194–198, 2007.

[6] W. S. Wang B. Zhang and M. Orshansky. “FASER: Fast Analysis of Soft Error Susceptibility for Cell-Based Design”. In *Proc. of ISQED*, pp. 755–760, 2006.

[7] M. Zhang and N. R. Shanbhag. A Soft Error Rate Analysis(SERA) Methodology. In *Proc. of ICCAD*, pp. 111–118, 2004.

[8] N. Miskov-Zivanov and D. Marculescu. “MARS-C: Modeling and Reduction of Soft errors in Combinational Circuits”. In *Proc. of DAC*, pp. 767–772, 2006.

[9] N. Miskov-Zivanov and D. Marculescu. “Modeling and Optimization for Soft-error Reliability of Sequential Circuits”. *IEEE Trans. Computer-Aided Design.*, Vol. 27, No. 5, pp. 803–816, May 2008.

[10] 赤峰悠介, 吉村正義, 松永裕介．“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 論文集, pp. 121–126, 2009.

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