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.

ソフトエラーを回避するLUTカスケード・エミュレータについて

中原　啓貴† 　笹尾　勤†

† 九州工業大学 情報工学部 〒 820–8502 福岡県飯塚市大字川津 680–4

あらまし　LUT カスケード・エミュレータは任意の順序回路を実現する．まず, 与えられた順序回路の組合せ論理関数部分を LUT カスケードに変換し, LUT カスケードのセルデータをメモリに格納する．次に, セルデータを逐次読み出しながら, 多出力論理関数を評価する．LUT カスケード・エミュレータのソフトエラーに対する耐性を向上するために, メモリを誤り訂正符号で符号化し, 誤り訂正回路で誤りを訂正する．また, メモリを定期的にスキャンし, 潜在しているソフトエラーを検出して修正する．フリップ・フロップは TMR を用いてソフトエラーを回避する．提案手法は 1 ビットのソフトエラーは全てマスクする．通常の LUT カスケード・エミュレータと比較してミッションタイムを 3 桁延長させた．

キーワード　LUT カスケード, 再構成可能アーキテクチャ, 自己適応性アーキテクチャ, 関数分解

A Soft-Error Tolerant Look-Up Table Cascade Emulator

Hiroki NAKAHARA† and Tsutomu SASAO†

† Department of Comoputer Science and Eloctronics, Kyushu Institute of Technology
kawazu 680–4, Iizuka, Fukuoka, 820–8502 Japan

Abstract　An LUT cascade emulator realizes an arbitrary sequential circuit. We convert the combinational part into multiple LUT cascades, and store LUT(cell) data into a memory in the LUT cascade emulator. It evaluates multi-output logic functions by reading cell data sequentially. To improve torelance to soft errors, we encode cell data stored in the memory by error correcting codes. Also, we add an error-correcting circuit. A scanning circuit periodically scans the memory. When it detects a soft error, it remove the error by write-backing the correct data into the memory. To avoid the soft error for flip-flops, we employ a TMR (Triple Module Redundancy) technique. Our method detects the soft errors in a single bit. Also, the mission time of our method is more than 1000x of an ordinary LUT cascade emulator.

Key words　LUT cascade, Reconfigurable architecture, Self-adaptability architecture, Functional decomposition

1. はじめに

プロセス微細化による高集積化に伴い, LSI 設計はますます複雑になっており, 消費電力増加に加え, 回路のバラつきやランダム故障に伴う信頼性低下が問題となっている [1]．高エネルギーの外部放射線よって半導体記憶素子に一時的に電荷が生成し, ビット情報が反転する再現性の無い現象はソフトエラーと呼ばれ [13], ランダム故障の主な要因とされている．近年, 微細化による電源電圧の低下とトランジスタの寸法縮小により半導体素子が蓄える電荷容量が減少し, 保持データの反転に必要な電荷量 (臨界電荷量) が減少している．それに従い, 低エネルギー α 線, 高エネルギー宇宙線粒子, 及び熱中性子によるソフトエラーは宇宙空間ばかりでなく, 比較的放射線が弱い地上でも発生し, さらに発生頻度は顕著になっている [14].

ソフトエラー対策は様々な方法が提案されている [12]．一つの方法として, 放射線に強いデバイスを採用する方法がある．しかし, 高コストであり, 最先端デバイスよりも数世代前の性能しかない．低コスト・高性能を維持しつつソフトエラーに対処する最も現実的な方法は, 回路に冗長性を持たせることである．メモリは誤り検出・訂正回路を導入することでソフトエラーに対処可能である．ソフトエラーはランダムに発生するが, 通常は半導体記憶素子を破壊することはなく, 正しい値を書き直すことで正常な動作に戻すことができる．SRAM ベースの FPGA では, 回路の動作を逐次監視し, ソフトエラーを検出した場合, FPGA を書き直す手法が提案されている [15].

我々の研究グループでは再構成可能アーキテクチャである LUT カスケード・エミュレータを提案している．LUT カスケード [2] は, LUT(Look-Up Table) を直列多段に接続した構造を持ち, 多出力論理関数を実現する．LUT カスケード・エミュレータ [2] は LUT カスケードの LUT データを大規模なメモリに格納し, LUT データを逐次読み出すことで, LUT カスケードの模擬を行うアーキテクチャである．LUT カスケード・エミュレータでは LUT データを格納するメモリと信号線の接続情報メモリを書き換えることで, 種々の論理関数を実現可能である．LUT カスケード・エミュレータはマイクロプロセッサより一桁高速であり, 消費電力は低い．FPGA と比較して, 低速であり,

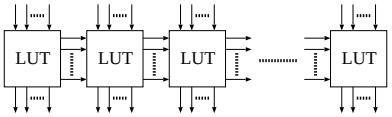

図 1　LUT カスケード

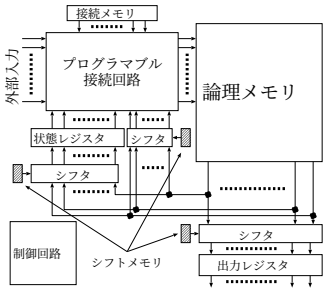

図 2　LUT カスケード・エミュレータ
(順序回路を実現)

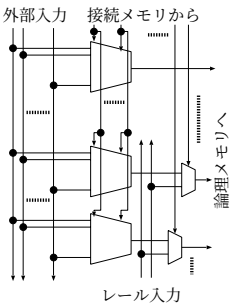

図 3　プログラマブル接続回路

消費電力が高いが, 配置配線が不要であり設計が容易であるという特徴を持つ. ただし, 大容量のメモリを長時間書き換えずに使用するため, ソフトエラー対策が必要である. 後述するが, 提案手法を用いることでソフトエラーに対しては回路全体を書き直す必要がないので, FPGA よりもアベイラビリティは高い. 本論文では, LUT カスケード・エミュレータのソフトエラーを回避する手法について述べる.

2. LUT カスケード・エミュレータ

2.1 LUT カスケード

多出力論理関数を実現する LUT カスケード [2] を図 1 に示す. LUT カスケードは LUT(**セル**) を直列多段に接続した回路である. LUT 間の接続線を**レール**と呼ぶ. LUT カスケードはセル間の接続線が前後のセルに限定されていることから, FPGA や ASIC と比較して配線が簡単であり, 遅延時間の見積りが容易である. LUT カスケードは多出力論理関数を表現する BDD(BDD_for_CF) [3] に対して, 関数分解を繰り返し適用することにより得られる [4]. 従って, 論理合成が比較的容易である.

2.2 LUT カスケード・エミュレータ

LUT カスケードではレールの本数, LUT の段数を一旦決めてしまうと, 実現できる関数は限られてしまう. より柔軟性の高いアーキテクチャとして LUT カスケード・エミュレータが提案されている [2].

図 2 に示す LUT カスケード・エミュレータは順序回路を実現する. **論理メモリ**は LUT カスケードのセルデータを格納する. **プログラマブル接続回路**はセルを読み出すための信号を選択する. **接続メモリ**は接続情報を格納する. 読み出したデータは, **シフタ**を用いてレジスタの所定の位置に格納する. **シフトメモリ**にシフト情報を保持する. **出力レジスタ**は外部出力を保持する. **状態レジスタ**は状態変数を保持する. 状態レジスタは, 状態入力レジスタ, 状態出力レジスタの 2 つで構成される. **制御回路**は LUT カスケード・エミュレータの動作の制御を行う. 図 3 にプログラマブル接続回路の実現法の一例を示す. LUT カスケード・エミュレータの動作を以下に示す.

1. 外部入力の取り込み: 外部入力をレジスタに取り込む. また, 評価するセルの番号 ← 0 とする.
2. セルのアドレス生成: 接続メモリの接続情報に応じて, プログラマブル接続回路は外部入力とレール出力からセルのアドレスを生成する.
3. セルの読み出し: 論理メモリをアクセスし, セルデータを読み出す. 読み出したデータをシフタで出力レジスタの所定の位置に格納する. 同様にレール出力も取り出す. 最終段のセルでなければ, セルの番号 ← セルの番号 +1 とし, 2. へ戻る.
4. 出力レジスタの値を出力する. また状態出力レジスタの値を状態入力レジスタに取り込み, 1. へ戻る.

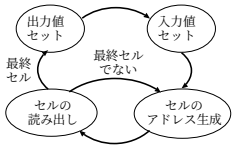

図 4　LUT カスケード・エミュレータの状態図

表 1　各メモリのサイズ

関数名	入力数	出力数	c	r	$M_{connect}$ [bit]	M_{sift} [bit]	M_{logic} [Mbit]
C432	36	7	4	14	1197	28	0.968
C1908	33	25	63	13	80	207	1.000
apex7	49	37	13	14	260	130	1.000
too_large	38	3	9	13	171	54	0.500
rot	135	107	201	10	3618	2211	1.000

c:セル数　r:最大レール数

上記の各動作を 4 つの状態に割り当てた状態図を図 4 示す.

LUT カスケードのセル数を c, 外部入力数を n, 外部出力数を m, 状態変数の個数を s, LUT カスケードの最大レール数を r とする. 図 3 より, 外部入力を選択するための MUX の制御入力数は $\lceil log_2 n \rceil$ であり, レール信号を r 個の 1-MUX で選択する. このときの接続メモリの大きさ $M_{connect}$ は

$$M_{connect} = c(\lceil log_2 n \rceil + r) \quad (1)$$

(ビット) である. また, シフトメモリの大きさ M_{sift} は

$$M_{sift} = c(\lceil log_2 m \rceil + \lceil log_2 s \rceil + \lceil log_2 r \rceil) \quad (2)$$

(ビット) である. 論理メモリのメモリ量を 1Mbit とし, 文献 [7] の手法を用いていくつかの MCNC ベンチマーク関数 [8] を LUT カスケード・エミュレータ上に実現した. そして, $M_{connect}$, M_{sift}, 及び論理メモリの必要メモリ量 M_{logic} (Mega bit) を求めた. 結果を表 1 に示す.

表 1 より明らかなように, 論理メモリが大部分を占める.

3. LUT カスケード・エミュレータの故障モデル

3.1 故障期間の仮定

システムは複数の構成要素からなる. システムの出力の異常を**障害 (failure)** という. 障害は, 構成要素の異常な出力によってもたらされる. 構成要素の異常な出力を**誤り (error)** と呼ぶ. 誤りは構成要素の**故障**の種類によって大きく 2 つに分類される. **ハードエラー**とはトランジスタなどが物理的に破壊される故障である. **ソフトエラー**(注1)(注2) とは放射線の誘起エネルギーによって半導体記憶素子に一時的に電荷が生成し, ビット情報が反転する現象による故障である. 図 5 に半導体の故障率の時間的変化を示す. 故障は発生する時間に応じて 3 つの領域に分類される. **初期故障期間**とは製造上の欠陥や材料不良による故障が発生する期間である. この期間では, ハードエラーが発生すると仮定する. **一定故障期間**とは初期故障期間以降, 摩耗故障が発生するまでに, 一定の確率で故障が発生する期間である. この期間では, ソフトエラーのみが発生すると仮定する. **摩耗故障期間**とは摩耗や疲労により, 故障が発生する期間である. 本

(注1) : 広義では電源電圧や温度の変動による動作不良を指すが, 本論文では放射線の影響のみに限定する.
(注2) : シングル・イベント・アップセット (SEU: Single Event Upset) ともいう.

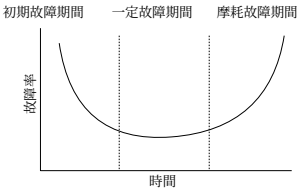

図 5　半導体の故障率の時間的変化

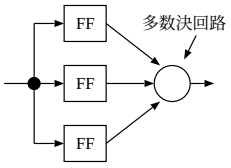

図 6　TMR 方式フリップ・フロップ

論文では, 有限時間である**ミッションタイム**間の信頼性のみ考慮するため, 摩耗故障については考慮しない. ただし, 使用期間中に摩耗故障が発生することのないように, 加速試験 (burning test) を行い, 十分な寿命を持つことを保証する必要がある.

3.2　初期故障期間に発生する故障について

LUT カスケード・エミュレータではチップ面積の大部分をメモリが占めるため, 故障の大部分はメモリの故障と考えられる. 初期故障期間ではメモリ部にハードエラーが発生すると仮定する. この期間の故障はメモリの故障診断法 [5], [6] を用いて故障を特定し, スペアセルに切替える手法や, メモリセルの故障箇所を回避するメモリパッキング [16] などを用いて回避可能である. また, 故障回避できない場合は不良品として除くので, これらの故障は一定故障期間では考慮しない. LUT カスケード・エミュレータではチップの大部分をメモリが占める. メモリ以外の部分は十分にマージンをとって設計するものとし, 故障の確率は十分小さいものと仮定する. 本論文では, メモリ及びフリップフロップのソフトエラーについてのみ考慮する.

4. LUT カスケード・エミュレータのソフトエラーの回避

4.1　故障の仮定

一定故障期間ではソフトエラーのみが発生すると仮定する. ソフトエラーが発生する箇所は電荷を蓄える部分, すなわち, メモリセルとレジスタ内部のフリップ・フロップとする. また, ソフトエラーは単一ビットのみ生じるものとする.

4.2　レジスタ内のフリップ・フロップのソフトエラー回避

VLSI の微細化が進み, デザインルールが $0.1\mu m$ 以下になると, フリップ・フロップにもソフトエラーが発生する [9]. フリップ・フロップのソフトエラー対策は様々研究されており, 入力と回路内部ソフトエラーのマスクラッチ回路を構成する方法 [9], 回路内部ソフトエラーに強いラッチ回路構成する方法 [10], ラッチにローパスフィルタを挿入する方法 [11] などが提案されている. 本論文では, 一般的な高信頼化手法である **3 重化 (TMR: Triple Modular Redundancy)** フリップ・フロップを用いてレジスタを実装する. TMR 方式フリップ・フロップを図 6 に示す.

4.3　誤り訂正符号を用いたメモリセルのソフトエラー回避

誤り訂正符号 (ECC:Error Correcting Code) の一種として単一誤り訂正 (SEC: Single Error Correcting) 符号が挙げられる. SEC 符号はコンピュータのメモリの誤り検出・訂正に用いられている. SEC(k,q) 符号は, k ビットの情報部と q ビットの検査部から成る $k+q$ ビットの組織符号である.

SEC 符号 $\vec{w}$ は, 検査行列 $\vec{H}$ に対し $\vec{H}\vec{w}^T = 0$ を満足する符号である. ただし, $\vec{w}$ は $k+q$ ビットである. $\vec{k}$ を情報部とする. SEC 符号は $\vec{H}\vec{G}^T = 0$ を満足する生成行列 $\vec{G}$ を用いて $\vec{w} = \vec{k}\vec{G}$ で得られる. SEC 符号 $\vec{y}$ は $\vec{s} = \vec{y}\vec{H}$ で求められる**シンドローム** $\vec{s}$ を調べることにより誤り検出と誤り箇所を特定することが可能である. SEC 符号では, 生成行列 $\vec{G}$ の行ベクトルの重みの最小値が 3 なので, 単一誤りの検出・訂正を行うことが可能である. 表 2 に必要な検査部のビット数を示す.

表 2　SEC 符号のビット数

情報部	検査部
16	6
32	7
64	8
128	9

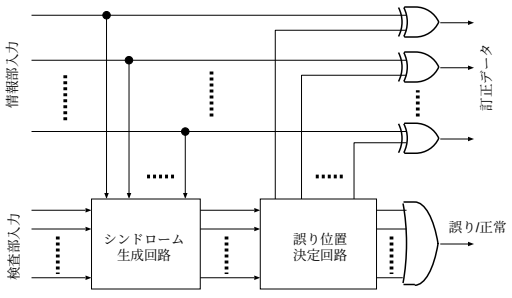

図 7　SEC 符号を用いた誤り検出・訂正回路

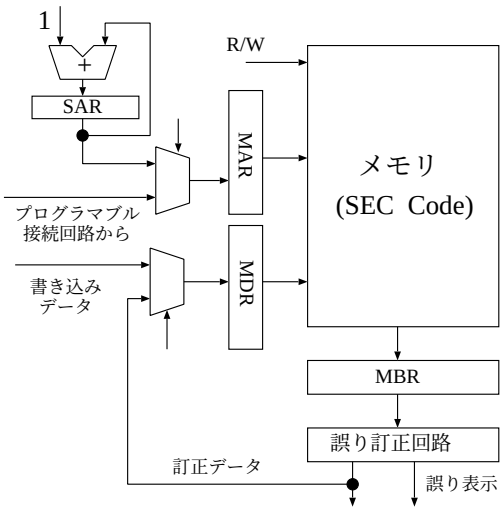

図 8　メモリのスキャンを行う回路

図 7 に SEC 符号を用いたメモリの誤り検出回路を示す. 誤りを検出するには, シンドロームを計算し, シンドロームが 0 ベクトルであるか調べればよい. シンドロームの計算は, EXOR ゲートを用いた回路で実行できる. 誤り位置決定回路は, 非 0 ベクトルのシンドロームから誤りビットを決定し, 誤り訂正データを出力する.

4.4　メモリ内に潜伏するソフトエラーの修復

メモリで発生したソフトエラーは, エラーを保持するワードが読み出されるまで検出されない. また, 誤り訂正回路はメモリ出力の誤りを訂正するが, ソフトエラーは依然としてメモリ内に残る. ソフトエラーが残っているワード内に第二のソフトエラーが発生した場合, SEC 符号を用いた誤り訂正回路は誤りを検出できない. 従って, 単一誤り訂正手法を用いてソフトエラーを回避するには, ソフトエラーを保持するワードに第二のソフトエラーが発生する前に, ソフトエラーを修復する必要がある. 本論文では, メモリを定期的に読み出して (スキャンして) ソフトエラーを検出し, 正しいデータを書き戻す手法について述べる.

図 8 にメモリのスキャンを行う回路を示す. **MAR (Memory Address Register)** はメモリのアドレスを保持するレジスタ, **MBR (Memory Buffer Register)** はメモリの出力を保持するレジスタ, **MDR (Memory Data Register)** はメモリの書き込みデータを保持するレジスタ, **SAR (Scan Address Register)** はスキャンするワードのアドレスを保持するレジスタである. 誤り訂正回路は 4.3 節で説明した SEC

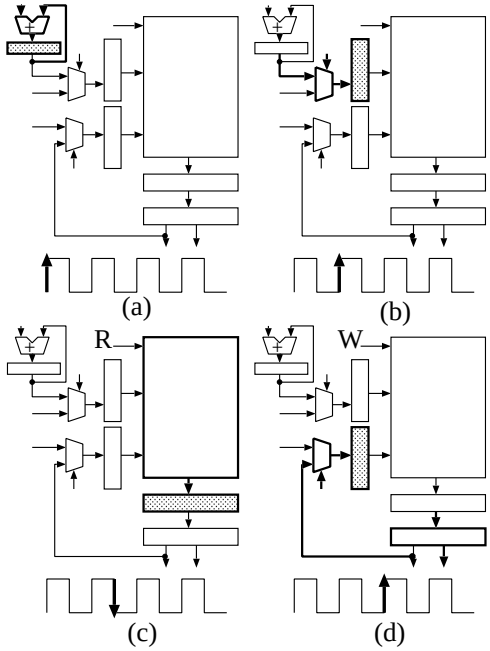

図 9　1 ワードのスキャンの方法

符号用の回路であり, メモリは SEC 符号化されているものとする.

［例 4.1］ 図 9 に 1 ワードのスキャンを行う例を示す. 最後のセルの出力を所定のレジスタに代入している間に, スキャンするワードのアドレスを 1 つ増加する (図 8(a)). 次のクロックの立上りに, スキャンを行うワードのアドレスを MAR に取り込む (図 8(b)). クロックの立ち下がりでメモリを読み出し, MBR に取り込む (図 8(c)). そして, 誤り訂正回路でメモリ出力の誤りを訂正する. 誤りが存在する場合, 訂正データを MDR に取り込む (図 8(d)). 次のクロックで訂正データを書き戻すことで, メモリのソフトエラーを修復する.

LUT カスケード・エミュレータでは各レジスタの更新と初期化を行う間, メモリを使用しない. よって本手法は, メモリを使用しない間にスキャンと修復を行うため, パフォーマンスのオーバーヘッドが生じることはない(注3). ハードウェアのオーバーヘッドは, MAR と MDR へ書き込むデータの選択を行うマルチプレクサ, スキャンアドレス生成回路 (加算器と SAR), 及びスキャンと書き戻しの制御を行う制御回路と制御信号である.

スキャン回路が単一ビットソフトエラーを正しく修復するためには, 第一のソフトエラーが発生してから, 第二のソフトエラーが発生するまでにスキャンと修復作業を完了する必要がある. つまり, 以下の関係を満たす必要がある.

$$Time_{SC} < MTBF \tag{3}$$

ここで, MTBF(Mean Time Between Failure) とは平均故障間隔であり, 誤りが発生する時間の平均値を示す. $Time_{SC}$ はスキャン・修復作業時間であり, メモリに発生したソフトエラーを修復する時間の最悪値を示す. LUT カスケード・エミュレータの最大セル数を $Cell_{max}$, 動作周波数を $Clock$[Hz], メモリ量を M[bit], 1 ワードのビット数を m[bit] とすると, $Time_{SC}$ は以下の式で求められる.

$$Time_{SC} = (2Cell_{max} + 2) \times \frac{1}{Clock} \times \frac{M}{m} \tag{4}$$

(注3)：メモリの書き込み時間が 1 クロックの場合にのみ成立する. 書き込みに数クロックを要する場合は, 終了するまで回路を停止しなければならない.

$Cell_{max} = 128$, $M = 32k \times 32$, $m = 32$ とし, FPGA (Altera Stratix EP1S60F1020C5) 上に LUT カスケード・エミュレータのプロトタイプを設計し $Clock = 40 \times 10^6$ $[Hz]$ を得た. 式 4 より, $Time_{SC} = 0.2113$[sec] であった. SRAM(注4)のソフトエラー発生率を 1000(FIT/Mbit) とすると [12], プロトタイプから得た $Time_{SC}$ は式 (3) を満たすので, 提案手法は SRAM 上に発生するソフトエラーに対処可能である. ただし, FIT は 10^9 時間に発生するエラーの個数を示す単位である. LUT カスケード・エミュレータを ASIC で実現する場合には, $Clock$ をさらに高速にできるため, ソフトエラーに十分対応可能である.

5. 提案手法の評価

5.1 信頼度の評価

本論文で提案したソフトエラー対策による信頼度向上を定量的に評価する. ここで, メモリセルとフリップ・フロップのソフトエラーが発生する確率は同一であるものと仮定する.

時刻 t で動作している確率を時刻 t の**信頼度 (reliability)** とする. メモリやフリップ・フロップを構成する 1 ビット分の回路の信頼度を $R_t(t)$ とすると,

$$R_t(t) = \exp(-\lambda t) \tag{5}$$

である. まず, メモリの信頼度を考える. ワード数を W, 1 ワードのビット数を m とすると, ECC を実装しない場合, メモリの信頼度 $R_{NonSEC}(t)$ は

$$\begin{aligned} R_{NonSEC}(t) &= \prod_{W}^{i=1}(\prod_{m}^{j=1} R_t(t)) \\ &= \exp(-Wm\lambda t) \end{aligned} \tag{6}$$

である. また, ECC を実装した場合 1 ワードの信頼度 $R_W(t)$ は

$$\begin{aligned} R_W(t) &= \exp(-m\lambda t) \\ &\quad + m(1 - \exp(-m\lambda t))\exp(-(m-1)\lambda t) \end{aligned} \tag{7}$$

である. ここで, 第一項は 1 ワードが全て正常である確率を表し, 第二項は 1 ワード内に 1 ビットだけ誤りがある確率を表す. 従って, メモリの信頼度 $R_{SEC}(t)$ は

$$\begin{aligned} R_{SEC}(t) &= [R_W(t)]^W \\ &= [m\exp((1-m)\lambda t) - (m-1)\exp(-m\lambda t)]^W \end{aligned} \tag{8}$$

である. 次に, p ビットのレジスタの信頼度を考える. ソフトエラー対策を施さない場合, レジスタの信頼度 R_{NonTMR} は

$$R_{NonTMR}(t) = \exp(-p\lambda t) \tag{9}$$

である. TMR 化したフリップ・フロップの信頼度を $R_{FF}(t)$ とすると

$$R_{FF}(t) = R_t(t)^3 + 3R_t(t)^2(1 - R_t(t)) \tag{10}$$

である. ここで, 第一項は 3 個の FF が全て正常である確率であり, 第二項は 3 個の FF の中で 2 個が正常である確率である. よって, TMR フリップ・フロップで実現したレジスタの信頼度 $R_{TMR}(t)$ は

$$R_{TMR}(t) = [3\exp(-2\lambda t) - 2\exp(-3\lambda t)]^p \tag{11}$$

である. ソフトエラー対策を何も施さない場合: $R_1(t) =$

(注4)：LUT カスケード・エミュレータの論理メモリを DRAM で実現することも可能である

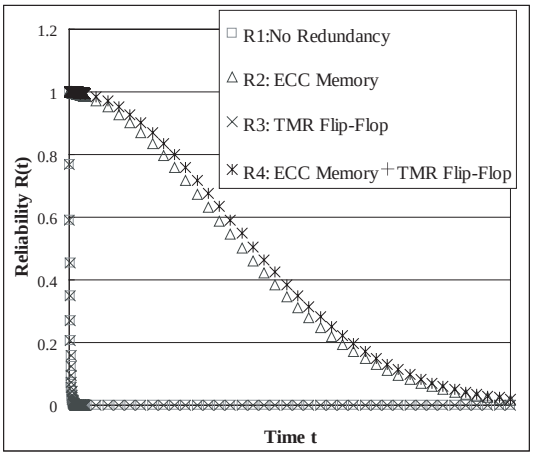

図 10　各手法の信頼度

表 3　各手法の MTID

手法	MTID
ソフトエラー対策を何も施さない	1.0000
各メモリを SEC 符号化	1540.7089
各レジスタを TMR フリップ・フロップで実現	1.0004
各メモリを SEC 符号化し, さらに各レジスタを TMR フリップ・フロップで実現	2608.9992

$R_{NonSEC}(t)R_{NonTMR}(t)$, 各メモリを SEC 符号化した場合: $R_2(t) = R_{SEC}(t)R_{NonTMR}(t)$, 各レジスタを TMR 方式フリップ・フロップで実現した場合: $R_3(t) = R_{NonSEC}(t)R_{TMR}(t)$, 各メモリを SEC 符号化し, さらに各レジスタを TMR 方式フリップ・フロップで実現した場合: $R_4(t) = R_{SEC}(t)R_{TMR}(t)$ に対してそれぞれ信頼度を求めた結果を図 10 に示す.

これらの手法を評価するため, **ミッションタイム改善度 (MTID: Mission Time Improvement Degree)** を定義する. エラー対策を施したシステムの信頼度が 99%まで低下する時間を T_R, エラー対策を施さないシステムの信頼度が 99%まで低下する時間を T_{NonR} とすると,

$$MTID = \frac{T_R}{T_{NonR}} \tag{12}$$

各手法に対して MTID を求めた結果を表 3 に示す.

表 3 より, メモリに誤り訂正符号を実装した場合, ミッションタイムを 3 桁改善することができた. TMR 方式フリップ・フロップを採用しただけでは, MTID はほとんど改善できなかった. これは, メモリ信頼度が支配的であるため, フリップ・フロップのみの信頼度を向上させても全体の信頼度にほとんど影響を及ぼさないからである. 各メモリを SEC 符号化し, 更に各レジスタを TMR フリップ・フロップで実現した場合には, 各メモリを SEC 符号化しただけの場合と比較して, MTID を大きく改善できた. これは, ソフトエラーの影響を低減した場合は, 全体の信頼度はフリップ・フロップの信頼度に大きく依存するため, フリップ・フロップの信頼度の改善が MTID を改善するからである.

6. むすび

本論文では, LUT カスケード・エミュレータのソフトエラーを回避し, 信頼性を向上させる手法について述べた. メモリセルとフリップフロップに単一ビットのソフトエラーが発生すると仮定した. ソフトエラーに対する信頼性を向上するために, メモリを SEC 符号で符号化し, 誤り訂正回路で誤りを訂正する. また, メモリを定期的にスキャンし, 潜在しているソフトエラーを検出して修正する. 潜在的なソフトエラーを回避するメモリのスキャン・書き戻し時間は現在のソフトエラー平均発生期間と比較して十分に短いことを検証した. フリップフロップは TMR を用いてソフトエラーを回避する. 提案手法は単一ビットのソフトエラーは全てマスクする. 通常の LUT カスケード・エミュレータと比較してミッションタイムを 3 桁延長させた.

7. 謝　辞

本研究は, 一部, 日本学術振興会・科学研究費補助金, および, 文部科学省・北九州地域知的クラスター創成事業の補助金による. 明治大学井口助教授には有益な助言を頂いた. また, 千葉大学伊東秀男教授には, ソフトエラーに関する貴重な資料を頂いた.

文　献

[1] J. M. Rabaey, "Design at the end of the silicon roadmap," *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, Shanghai, Jan., 2005, Volume 1, pp.18-21.

[2] T. Sasao, Y. Iguchi, and M. Matsuura, "LUT cascades and emulators for realizations of logic functions," *RM2005*, Tokyo, Japan, Sept. 5-6, 2005, pp.63-70.

[3] P. Ashar, and S. Malik, "Fast functional simulation using branching programs," *ICCAD'95*, Nov.1995, pp.408-412.

[4] T. Sasao, and M. Matsuura, "A method to decompose multiple-output logic functions," *Proc. Design Automation Conference (DAC)*, San Diego, CA, USA, June 2-6, 2004, pp.428-433.

[5] J. T. Chen, J. Rajski, J. Khare, O. Kebichi, and W. Maly, "Enabling embedded memory diagnosis via test response compression," *IEEE International Test Conference*, pp.292-298, 2002.

[6] A. Iseno, Y. Iguchi, and T. Sasao, "Fault diagnosis for RAMs using Walsh spectrum," *IEICE Trans. Information and Systems*, Vol. E87-D, No.3, March 2004, pp. 592-600.

[7] H. Nakahara, T. Sasao, and M. Matsuura, "A design algorithm for sequential circuits using LUT rings," *IEICE Trans. Fundamentals of Electronics*, Vol. E88-A, No.12, Dec. 2005, pp. 3342-3350.

[8] S. Yang, "Logic synthesis and optimization benchmark user guide version 3.0," MCNC, Jan. 1991.

[9] M. Nicolaidis, "Time redundancy-based soft-error tolerance to rescue nanometer technologies," *Proc. IEEE VLSI TEST Symp.*, pp. 86-94, 1999.

[10] M. Omana, "Novel transient fault hardended static latch," *IEEE International Test Conf.*, pp. 88-892, 2003.

[11] A. Maheshwari, I. Koren, and W. Burleson, "Techniques for transient fault sensitivity analysis and reduction in VLSI circuits," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 597-604, 2003.

[12] S. Mitra, N. Seifert, M. Zhang, Q. Sbi, and K. S. Kim, "Robust System Design with Built-In Soft-Error Resilience," *IEEE Design and Test of Compt.*, pp. 43-52, Feb., 2005.

[13] H. T. Nguyen, and Y. Yagil, "A systematic approach to SER estimation and solutions," *Proc. of the 41st Intl. Reliability Physical Symp.*, pp. 60-70, Dallas, Texas, 2003.

[14] 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," *Proc. of the Intl. Conf. on Dependable Systems and Networks (DSN'02)*, Washington D. C., June, 2002.

[15] G. H. Asadi, and M. B. Tahoori, "Soft error mitigation for SRAM-based FPGAs," *IEEE VLSI Test Symposium*, pp.207-212, 2005.

[16] T. Sasao, M. Kusano, and M. Matsuura, "Optimization methods in look-up table rings," *IWLS-2004*, June 2-4, 2004, Temecula, California, USA, pp.431-437.