社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2005－ＳＬＤＭ－121（28）
2005／10／21

プログラムの動作を考慮したコンピュータシステムのソフトエラー数見積もり技術

杉原真† 石原亨†† 橋本浩二††† 室山真徳††

† 財団法人九州システム情報技術研究所，〒814-0001 福岡市早良区百道浜 2-1-22 福岡 SRP センタービル 7F
†† 九州大学，〒814-0001 福岡市早良区百道浜 3-8-33 福岡システム LSI 総合開発センター 3F
††† 福岡大学，〒814-0180 福岡市城南区七隈 8-19-1
E-mail: †sugihara@isit.or.jp

あらまし 集積回路の加工寸法の縮小に伴い，回路の動作電圧とノイズマージンが低下し，宇宙線に起因するソフトエラーが無視できなくなってきた．本稿では，CPU，キャッシュ，及び主記憶からなるコンピュータシステムのソフトエラーの発生回数をシステムレベルで評価する手法を提案する．キャッシュメモリやレジスタなどのハードウェア単体のソフトエラー率を単純に足し合わせることによって求めたソフトエラー率はしばしば悲観的な値となる．これは，コンピュータシステムで発生するソフトエラーが，ハードウェア単体のソフトエラー率 SER だけでなくプログラムの振る舞いや使用されている誤り訂正技術の種類と適用箇所に依存するためである．本稿で提案するソフトエラー数の見積もり技術は，プログラムの動的な振る舞いを考慮し，ソフトエラー数を見積もるものである．はじめに，メモリ階層を有するコンピュータシステムのためのソフトエラー数見積もりモデルを議論する．次に，本モデルを用いて，サイクル精度の命令セットシミュレーションによりソフトエラー発生回数を見積もるアルゴリズムを議論する．提案手法はコンピュータシステムにおける高信頼化の指針を与えるものである．

キーワード ソフトエラー，信頼性，見積もり

A Reliability Evaluation Technique for Soft-Error Susceptible Computer Systems

Makoto SUGIHARA†, Tohru ISHIHARA††, Koji HASHIMOTO†††, and Masanori MUROYAMA††

† ISIT, 2-1-22 Momochihama, Sawara-ku, Fukuoka 814-0001 Japan
†† Kyushu University, 3-8-33 Momochihama, Sawara-ku, Fukuoka 814-0001 Japan
††† Fukuoka University, 8-19-1 Nanakuma, Johnan-ku, Fukuoka 814-0180 Japan
E-mail: †sugihara@isit.or.jp

Abstract As the feature size of integrated circuits shrinks, their voltage and noise margins are lowered and the soft error issue becomes more and more serious. In this paper, we propose a reliability evaluation technique for soft-error susceptible computer systems. Conventional static soft-error estimation for computer systems with SERs of memory modules results in their pessimistic SERs. This is because memory cells under usage depend on dynamic behavior of computer systems and not all soft errors on memory modules make computer systems faulty. In our proposed technique, dynamic behavior of computer systems is taken into account for accurate soft-error estimation. First, we introduce a model to evaluate soft-errors of computer systems. To our best knowledge, this is the first study to model dynamic behavior of computer systems. Next, soft-error estimation algorithm considering dynamic behavior of computer systems is proposed. Our reliability evaluation technique gives system designers a means to estimate the number of soft errors of their products early in design with low cost.

Key words Soft Error, Reliability, Estimation

1. はじめに

回路の特定箇所が永久的に破壊されるハードエラーとは異なり，ソフトエラーはデバイスの動作中にランダムに発生する一過性の誤動作である．1979 年に May によって，パッケージ中に含まれる放射性物質から放出される α 線がソフトエラーの原

因であることが突き止められて以来 [5]，そのメカニズムの解明と防止策に多大な努力が払われてきた [8]．昨今の微細化技術の進歩により集積回路の最小加工寸法はナノスケールにまで達し，高速，高機能，かつ，低電力消費である回路を設計できるようになった．最小加工寸法がナノスケールに達するとき，集積回路中のトランジスタはますます微細化され，ソフトエラーを引き起こす α 粒子などの外乱の影響を受けやすくなる．これまでは，ソフトエラーの影響は宇宙空間向け回路などのミッションクリティカルなシステムの設計においてのみ考慮され，民生機器の設計では考慮されてこなかった．今後は，民生機器向けの集積回路においても高信頼化設計及び信頼性評価技術はますます重要となる．

電子機器でのソフトエラーの影響が深刻になる一方，ソフトエラーを考慮した信頼性見積もり技術については発展途上の段階にある．[7,13,14] においては，欠陥注入法 (fault injection) を用いるソフトエラー見積もり手法を議論している．欠陥注入法は欠陥をネットリスト等に埋め込んでシミュレーションし，欠陥がない場合と比較して欠陥の数を見積もる方法である．欠陥注入法は出力結果に影響を与えない欠陥についてもシミュレートできる．シミュレーション時間がより短い確率的な解析手法 [2] では確率計算によってエラーを伝搬させ全体の欠陥の数を見積もる．上記の解析手法はいずれも設計の初期段階には適用できず，シミュレーション時間が長くなる．[6,14] では，アーキテクチャレベルでプログラムの実行結果に影響を与える欠陥を考慮した見積もり手法を提案している．例えばプログラム実行時には使用しなかった命令や NOP 命令は欠陥が入っても最終的な出力には影響を与えない．抽象度が欠陥注入法よりも高いため，設計段階早期に利用できシミュレーション時間が短い．プログラムの振る舞いが欠陥へどのように影響を与えるかについての研究も行われている [3]．プログラムのメモリアクセス方法の違いを考慮しソフトエラーを取り扱っている．ただし変数の生存期間を考慮し，ソフトエラーを取り扱ったものではない．

メモリは回路構造が規則的であるために，メモリの SER(soft error rate：ソフトエラー率) は単位面積当りの SER を実験的に得ることによって算出することができる．この SER はメモリ上の全てのソフトエラーを対象とするものである．動的な振る舞いを持ち合わせるコンピュータシステムにおいては，メモリの空間的あるいは時間的使用量はその動作に依存し，メモリ上の全てのソフトエラーがコンピュータシステムのソフトエラーとして発現するわけではない．この意味で，コンピュータシステム中のメモリの信頼性を正確に評価することは難しい．論理回路の場合は，回路の構造がメモリと比べて不規則で複雑な構造をとるために，信頼性の評価が難しい．上村らは論理レベルでのソフトエラーシミュレータを開発した [1,9–12]．上村らは，130nm ノードの論理回路の SER と比較して 65nm ノードの論理回路の SER は 9 倍以上になると予測している．今後は論理回路のソフトエラー評価技術も重要性を増すと予測される．

今後の集積回路設計における課題の一つは，システムレベルにおける信頼性評価技術の確立であり，喫緊の課題として挙げられる．本稿では，ソフトウェアとハードウェアの両者からなるコンピュータ・システムを設計する早期の段階で，そのシステムの信頼性を正確に評価する技術を提案する．ハードウェアとソフトウェアからなるコンピュータシステムのソフトエラーを見積もるとき，メモリや論理回路のようなハードウェアの単体部品としての静的な SER のみではなく，プログラムに規定されるハードウェアの振る舞いも考慮しなければならない．これはプログラムによって規定されるコンピュータシステムの振る舞いによってメモリや論理回路などのハードウェア資源の占有時間が異なり，コンピュータシステムの信頼性を左右するためである．ソフトエラーを生じるハードウェアの使用時間を得ることは容易なことではない．コンピュータシステムは，レジスタ，L1 キャッシュ，L2 キャッシュ，及び主記憶などとメモリが階層的に配置され，メモリの空間的及び時間的使用部分を同定することが難しい．提案する信頼性評価技術は，サイクル精度の命令セットシミュレーションにより動的な振る舞いを考慮して，メモリ階層構造を持ち合わせるコンピュータシステムの信頼性を見積もるものである．我々の知る限り，提案手法は，サイクル精度の命令セットシミュレーションによりコンピュータシステムの動的な振る舞いを考慮し，信頼性を評価する初めての試みである．

本稿は次のように構成される．2. 節ではデータの生存期間を考慮したメモリ上のソフトエラーをモデル化する．3. 節では，コンピュータシステムの命令メモリに関する動作に依存するソフトエラーについて議論する．4. 節では，コンピュータシステムのデータメモリに関する動作に依存するソフトエラーについて議論する．データメモリは命令メモリと異なり，書き込みがある意味で異なる．5. 節では，サイクル精度の命令セットシミュレーションを用いた信頼性見積もりアルゴリズムについて述べる．6. 節では，提案する手法に基づきいくつかのベンチマークプログラムに対して信頼性見積もりを行う．7. 節で本論文のまとめ，今後の課題について述べる．

2. データの生存期間を考慮したソフトエラー

本節では，CPU，キャッシュ，および，主記憶を有するコンピュータシステムにおけるソフトエラー数を見積もりを行うために，システムに影響を及ぼすソフトエラーをモデル化する．具体的には，メモリの時間的，局所的使用量を同定するためにワード単位で求めたデータの生存期間を元にソフトエラー数を見積もる．ここで，「データの生存期間」とはデータの誕生時から消滅時 (あるいは，無効になる時) までの時間を指す．

一般に，CPU を中心とするコンピュータシステムはレジスタ，キャッシュ，及び，主記憶メモリといった複数のメモリからなる階層構造を持ち合わせる．対象とするコンピュータシステムのメモリ階層は N_{mem} 個のメモリモジュールから構成されるとし，CPU に近い方から順に $M_1, M_2, \cdots, M_{N_{\mathrm{mem}}}$ と表記する．CPU を中心とするコンピュータシステムにおける命令は，

（1） コンパイラなどでプログラム (命令メモリ上に蓄えられるデータ) を生成したものを命令メモリ上にロードする，

（2） 命令メモリから実行する命令を CPU にフェッチし，CPU で用いる，

といった手続きで用いられ，基本的にはリードオンリーである．また，コンピュータシステムにおけるデータは，

（1） プログラムの初期値として与えられ，データメモリ上にロードされる，あるいは，以前の CPU の実行で生成されたデータとして与えられ，データメモリ上にロードされる，

（2） データメモリから CPU で用いるデータにフェッチし，CPU で用いる，

といった手続きで用いられる．命令メモリとデータメモリの相違点は CPU からメモリへの書き込みの有無である．いずれせよ，両者のデータには生存期間がある．CPU がワード単位で処理を行うといった性質上，データの生存期間はワードデータ毎に異なる．

メモリモジュール M_i 上の 1 ワードのデータにおいて単位時間当りに発生するソフトエラーの回数，すなわち SER を

$error\_rate_{M_i}$ とする．モジュール M_i 上に 1 ワード長のデータ d を $time$ の時間だけ保持するとき，当該データのソフトエラー数 $error_{M_i}(d)$ は

$$error_{M_i}(d) = error\_rate_{M_i} \cdot time \tag{1}$$

と示される．

CPU から最も遠いメモリモジュール $M_{N_{\mathrm{mem}}}$ から CPU まで当該データを持ってくるとき，メモリモジュール M_i に当該データ d を $retention_{M_i}(d)$ の時間だけ保持するとき，当該データを CPU まで持ってくる過程において，

$$error_{\mathrm{system}}(d) = \sum_i error\_rate_{M_i} \cdot retention_{M_i}(d) \tag{2}$$

だけのソフトエラーを生じる．なお，$retention_{M_i}(d)$ はデータ d を CPU に伝搬するためにメモリモジュール M_i 上に保持しなければならない最小の時間を示し，コンピュータ・アーキテクチャに依存するものである．

3. 命令メモリのソフトエラー

プログラムの実行の開始から終了まで考えるとき，各々の命令には生存期間が存在し，これはプログラムの実行時間とは異なる．概して命令の誕生はプログラムを命令メモリにロードしたときである．命令メモリは基本的にリードオンリーであり，プログラムの実行時に変更されることはない．命令メモリのソフトエラーを考えるためには，各メモリモジュールにおける命令の読み出し動作を考えれば良い．具体的には，各メモリモジュールが当該命令を保持する時間のうち，どの部分がソフトエラーの影響を受けるかを考慮すれば良い．

まず，問題を細分化し，命令メモリ空間のアドレス a に保持される 1 ワードデータについて考える．アドレス a にある命令に対して i 番目に行われる命令フェッチを $if(a,i)$ と表記する．なお，$if(a,0)$ はプログラムの動作開始時を示す．例として，アドレス a のデータのコピーがレジスタ，キャッシュ，及び主記憶メモリ上に保持される過程を図 1 に示す．図において矩形は該当するメモリモジュールに命令のコピーがあることを示し，矩形のラベルはコピーの誕生時期を示す．この例では，アドレス a にある命令は 3 回だけアクセスされている．

まず，$if(a,1)$ では，L1 および L2 キャッシュのいずれにも当該データは存在せず，主記憶メモリ上にのみ存在する．必然的に主記憶メモリから CPU までデータを持ってくる必要がある．その過程で，当該データはまず L2 キャッシュにコピーされ，続いて，L1 キャッシュ及びレジスタにコピーされる．この例では，CPU から各メモリモジュールに対して命令転送要求があったとき，各メモリモジュールにて遅延が生じるという前提を設けている．あるデータをあるメモリモジュールからその上位のメモリモジュールにコピーする場合，コピーが完了した後に生じる，当該メモリモジュールにおけるソフトエラーは上位のメモリモジュールに影響を与えない．図の斜線の矩形のソフトエラーは $if(a,1)$ の実行に影響を及ぼし，他の部分のソフトエラーは $if(a,1)$ の実行に影響を及ぼさない．

$if(a,2)$ の場合では，$if(a,1)$ と同様に主記憶メモリ上にのみ当該データは存在する．$if(a,1)$ と同様の過程を辿り，当該命令は主記憶からレジスタまでコピーされる．点描画された矩形が $if(a,2)$ の実行に影響を及ぼすソフトエラーである．ここで，$if(a,1)$ で L1 及び L2 キャッシュ上に保持していたコピーは他のデータの上書きによって破棄されており，$if(a,1)$ とラベル付けされている白抜きの矩形部分のソフトエラーは $if(a,2)$ に影響を与えない点に注意されたい．

$if(a,3)$ では，当該命令が存在する最も上位のメモリモジュールは L1 キャッシュである．よって濃く塗られている矩形のソフトエラーが $if(a,3)$ のに影響を及ぼすものである．L2 キャッシュや主記憶メモリを参照する必要はなく，これらのメモリのソフトエラーは $if(a,3)$ に影響を及ぼさない．

さて，あるコンピュータシステム上である入力データを用いてあるプログラムを実行したとする．CPU 上で実行された命令データの列を $i_1, i_2, \cdots, i_{N_{\mathrm{inst}}}$ とする．ある命令データ i_i を実行するために当該データをメモリモジュール M_j 上で保持しなければならない最小の時間を $retention_{M_j}(i_i)$ とするとき，命令 i_i のソフトエラー $error(i_i)$ は，

$$error(i_i) = \sum_j error\_rate_{M_j} \cdot retention_{M_j}(i_i) \tag{3}$$

と与えられる．

よって，全ての命令メモリに関しては以下の回数だけソフトエラーを生じる．

$$error(\boldsymbol{i}) = \sum_i error(i_i) \tag{4}$$

$$= \sum_i \sum_j error\_rate_{M_j} \cdot retention_{M_j}(i_i) \tag{5}$$

ただし，$\boldsymbol{i} = (i_1, i_2, \cdots, i_{N_{\mathrm{inst}}})$ である．

4. データメモリのソフトエラー

データメモリは読み書きを行うものであり，読み出ししか行わなかった命令メモリよりもソフトエラーを考える上で複雑である．命令と同様にデータはプログラムの初期値として与えら

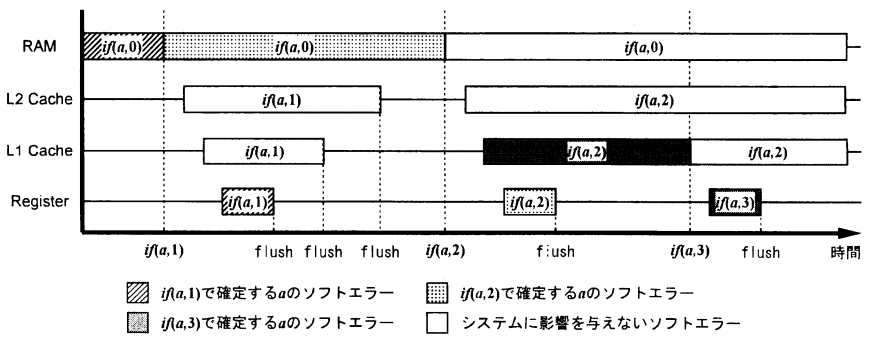

図 1　アドレス a の命令データの生存期間とソフトエラーの影響．

れる一方，命令と異なりデータはプログラム実行中に誕生する．たとえデータがキャッシュや主記憶メモリ上にあっても使われない場合がある．キャッシュや主記憶メモリ上にあるデータには有効な期間と無効な期間がある．各データがどの時点からどの時点まで有効であり，どの時点からどの時点まで無効であるのかを判断することは，システムのソフトエラーを判定する上で必要不可欠である．それでは，データがメモリモジュール上にある時間において，どの部分を有効とし，どの部分を無効とすれば良いのだろうか？本稿では，以下の方針でデータの生存期間を決定する．

- あるアドレスのデータに対してロード命令を発行した場合，当該データが誕生してからその時点までは有効．
- ユーザに明示的に指定されたデータは，そのデータが誕生してからプログラム終了時まで有効．このようなデータはプログラムの出力となる．

データメモリのソフトエラーを考えるとき，読み出し動作に伴う各階層のメモリモジュールへのデータのコピーを考慮するとともに，書き込み動作に伴う各階層のメモリモジュールへのデータの更新を考慮しなければならない．各階層のメモリモジュールのデータ更新の方針として，ライトバック方式，及びライトスルー方式があり，これらはシステムの信頼性を考慮する上で影響を及ぼすものである．ライトバック方式においては，データの更新の要求があったときに，最も近いキャッシュ上に対してのみ行われる．下位へのメモリモジュールへのデータの更新は当該データが他のデータに置き換わるときに行われる．ライトスルー方式においては，データの更新の要求があったときにキャッシュとその低位のメモリモジュールに対して行われる [4].

ライトバック

ライトバック方式のシステムにおけるソフトエラーについて考察する．アドレス a への i 番目のストアを $s(a,i)$ とし，アドレス a からの i 番目のロードを $l(a,i)$ とする．図 2 の例におけるアドレス a のデータのやり取りでは，まずストア命令が行われている．ライトバック方式を採用しているので，当該データは L1 キャッシュにのみ蓄えられ，L2 キャッシュ及び主記憶メモリへのデータ更新は行われない．次に，ロード命令が続く．このとき，当該データは L1 キャッシュ上に存在するために，L1 キャッシュから CPU へデータが送られる．$l(a,1)$ が取り扱うデータに関してのソフトエラー考えるためには，図の $s(a,1)$ と $l(a,1)$ とラベルづけされた斜線で示される矩形を考慮すれば良い．次にアドレス a へのストア命令が実行される．L1 キャッシュには値が上書きされるので，それ以前に保持していたデータは無効となる．それまでアドレス a に保持していた図の $s(a,1)$ とラベル付けされた白抜きの矩形のソフトエラーはシステムに影響を与えない．続いて，L1 キャッシュ上にあるアドレス a のデータは，他のデータの上書きにより L2 キャッシュに追い出される．アドレス a にあるデータを保持する最上位のメモリモジュールは L2 キャッシュとなる．この後，ロード命令が発行される．当該データが存在する最も上位のメモリモジュールである L2 キャッシュから L1 キャッシュへコピーされ，さらにレジスタにコピーされる．本ロード命令に着目したとき，このデータが誕生してからロード命令が完了するまでが生存期間となる．すなわち，図中で $s(a,2)$ とラベル付けされている点描画された矩形で生じるソフトエラーが $s(a,2)$ に影響を及ぼす．$s(a,2)$ とラベル付けされた白抜きの矩形のソフトエラーは $s(a,2)$ に影響を及ぼさない．

ライトスルー

ライトスルー方式のシステムにおけるソフトエラーについて考察する．図 3 に，アドレス a のデータに着目したときの各メモリモジュールの振る舞いの例を示す．この例では，まずアドレス a にストア命令 $s(a,1)$ が発行される．ライトスルー方式を採用しているので，ストア命令発行時には全てのメモリモジュールに当該データがコピーされる．上位のメモリモジュールから順にコピーが完了する．次に，ロード命令 $l(a,1)$ が発行

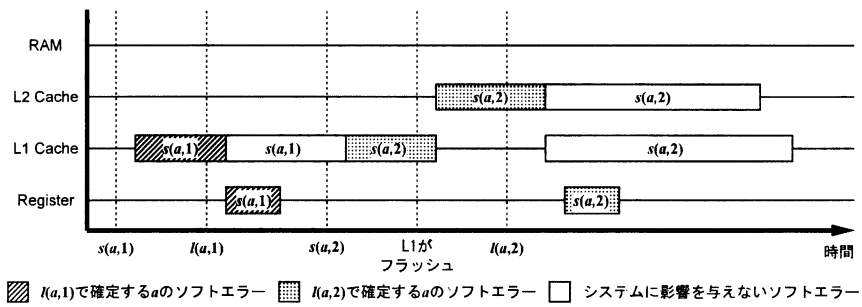

図 2　ライトバック方式におけるデータの生存期間とソフトエラーの影響．

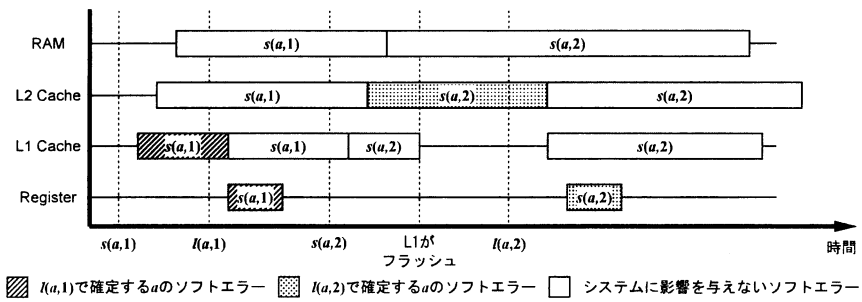

図 3　ライトスルー方式におけるデータの生存期間とソフトエラーの影響．

される．このとき，L1 キャッシュ上に当該データが存在するために，$l(a,1)$ に影響を及ぼすソフトエラーは $s(a,1)$ とラベル付けされた斜線で示される矩形で生じるソフトエラーである．続いて，ストア命令 $s(a,2)$ が発行される．各メモリモジュールにそれぞれ遅延を生じて，当該データがコピーされる．次に，L1 キャッシュ上にある当該データが追い出され，L2 キャッシュが当該データを保持する最上メモリモジュールとなる．この状態でロード命令 $l(a,2)$ が発行されると，$l(a,2)$ に影響を及ぼすソフトエラーは $s(a,2)$ とラベル付けされた点描画で示される矩形で生じるソフトエラーである．ここで注意したいのは，$s(a,2)$ が発行されたときに，L1 キャッシュの $s(a,2)$ とラベル付けされている白抜きの矩形のソフトエラーは $l(a,2)$ に影響を及ぼさない点である．

5. サイクル精度の命令セットシミュレーションによる信頼性見積もり

前説まで議論したように，CPU と複数のメモリ階層からなるコンピュータシステムのソフトエラー数を見積もるためには，命令の抽象度で各メモリモジュールにおける命令あるいはデータの生存期間が必要となる．命令およびデータの各メモリ上での正確な生存期間を得るために，サイクル精度の命令セットシミュレーションが必要となる．簡約化したアルゴリズムを図 4 に示す．アルゴリズムの入力は実行される命令列であり，出力は入力の命令列を実行したときに生じるソフトエラー数 $error_{system}$ である．まず，$error_{system}$ は 0 で初期化される．全てのメモリモジュールにおいては，ワードデータ単位で誕生時刻を記憶する．メモリモジュール上のワードデータの誕生時刻は 0 で初期化される．次に for 文が実行される．この中ではサイクル精度の命令セットシミュレーションを行い，CPU から主記憶までの全てのメモリの振る舞いをシミュレーションする．命令毎に命令のソフトエラーは計算され，$error_{system}$ に加算される．これに伴い，命令の誕生時間は現在の時刻を基準に更新される．当該命令がロードあるいはストア命令の場合，データメモリのソフトエラーに関しても計算が行われる．当該命令がロード命令であれば，当該命令にソフトエラーの影響を及ぼすワードデータの生存期間より算出したソフトエラーを $error_{system}$ に加算する．これに伴い，対象としたワードデータの誕生時刻を現在の時刻を元に更新する．当該命令がストア命令の場合，新たなデータが誕生することを意味するので更新が行われるワードデータの誕生時刻を現在の時刻を元に更新する．命令列の全ての命令に対して以上の手続きを行い，得られた $error_{system}$ が求めるソフトエラー数となる．

6. 実　　験

6.1 実験環境

ターゲット命令セットアーキテクチャには ARM V4T アーキテクチャを用い，命令セットシミュレーションには GNU のデバッガ（GDB）を用いた．ターゲットプロセッサは 200MHz で動作すると仮定し，1 クロックサイクル時間内 (5ns) に，メモリ 1 ワード (32 ビット) 中に発生するソフトエラーの個数を 1.0 [10^{-21}/word · cycle] (0.72FIT) と仮定した．このメモリと同じソフトエラー率を持つ 1M ビットのメモリは，およそ 5 年に一回の確率でソフトエラーを発生する．アプリケーションプログラムには compress, JPEG エンコーダ，および MPEG2 エンコーダを用いた．これら 3 種類のプログラムをそれぞれ，1,000,000 命令実行し，その間に発生するソフトエラーの個数を見積もる実験を行った．メモリ階層は，主記憶，レベル 1 キャッシュ（命令キャッシュとデータキャッシュ）およびレジスタファイルを仮定した．それぞれのメモリのソフトエラー率が 1, 2, 4, 8, 16, 32, 64[10^{-21}/word · cycle] の場合の，プログラム実行中に生じるソフトエラー数を見積もった．キャッシュラインサイズおよびキャッシュセット数は共に 32 とし，連想度は 1 から 64 の間の 2 のべき乗の値を変化させて実験を行った．キャッシュリプレイスのポリシーはライトバックとライトスルーの 2 種類を仮定した．

6.2 実験結果

図 5 に，データキャッシュの連想度を 1 から 64 まで変化させた時のソフトエラー数の見積もり結果を示す．実験結果から，

Soft Error Estimation Algorithm

```
Procedure EstimateSoftError
Input Instruction sequence given by trace.
Output error_system: errors for the system.
begin
    error_system を 0 で初期化.
    全てのメモリモジュールのワードデータのソフトエラーを 0 で初期化する
    入力データの全てのワードデータの誕生時刻を 0 で初期化する.
    for 全ての命令 do
        // 命令のソフトエラーの計算
        命令メモリに関して，当該命令に影響を及ぼすワードデータの生存時間を元に算出したソフトエラーを error_system に加算する.
        命令メモリに関して，当該命令に影響を及ぼすとしたメモリ上のワードデータの誕生時間を現在の時間を基準に更新.
        // データのソフトエラーの計算
        if 当該命令はロード命令である then
            当該データに影響を及ぼすワードデータの生存時間を元に算出したソフトエラーを error_system に加算する.
            当該データに影響を及ぼすとしたメモリ上のワードデータの誕生時間を現在の時間を基準に更新.
        else if 当該命令はストア命令である then
            データメモリにおける当該データを格納するメモリ上のワードデータの誕生時刻を更新する.
        endif
    endfor
    return error_system
end
```

図 4　ソフトエラー見積もりアルゴリズム.

キャッシュの連想度が 4 以上になると，システム動作に影響を与えるソフトエラーの数はキャッシュサイズにはほとんど依存しないことが確認できる．これは，プログラムの実行中に使用される命令数や変数の数は，アプリケーションプログラムとその入力データに依存して決まり，キャッシュサイズやキャッシュの構成には依存しないためである．連想度が低い範囲でキャッシュサイズとソフトエラーに強い相関が見られるのは，キャッシュサイズに依存してプログラムの実行時間が変化するためである．キャッシュメモリ中のどこかにソフトエラーが発生する確率はキャッシュサイズに比例して大きくなるが，システムの動作に影響を与えるソフトエラーの数はキャッシュサイズが大きくなると逆に小さくなることが確認できた．

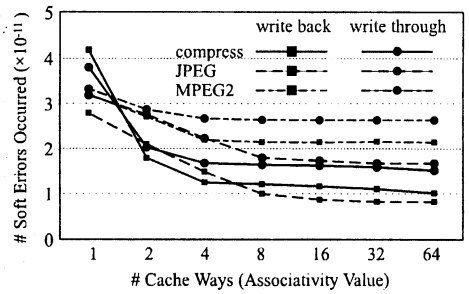

図 5　データキャッシュサイズ vs ソフトエラー数．

図 6 にデータキャッシュおよび命令キャッシュのソフトエラー率を個別に変化させた時のソフトエラー数の見積もり値を示した．命令キャッシュ単体のソフトエラー率低下が全体のソフトエラー数に与える影響が大きいことが確認できる．また，アプリケーションプログラムの種類によってもソフトエラー数が異なる．例えば，同じメモリ構成であっても MPEG2 は compress のおよそ 2 倍ソフトエラーの影響を受けることが分かる．従って，システムのソフトエラー耐性を強化するには，アプリケーションに応じて耐故障技術の適用箇所を適切に決定することが重要である．

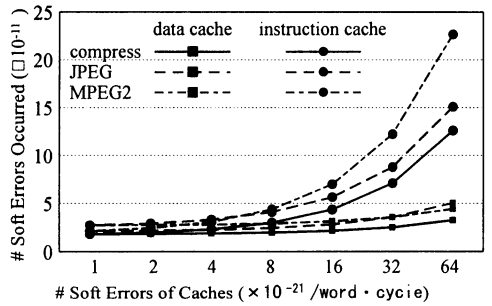

図 6　メモリモジュールの SER vs ソフトエラー数．

7. おわりに

本稿では，CPU，キャッシュメモリ，及び主記憶からなるコンピュータシステムのソフトエラー数を見積もるためのモデル及びシミュレーションベースの見積もりアルゴリズムを提案した．ARM V4T アーキテクチャを対象として，いくつかのアプリケーションプログラムの実行中に発生するソフトエラー数を見積もった結果，システムの動作に影響を与えるソフトエラーの数はキャッシュサイズが大きくなると逆に小さくなることが確認できた．キャッシュサイズが大きくなるとキャッシュモジュールの SER は増加するが，プログラム実行中に発生するソフトエラー数は逆に小さくなることが確認された．これは，キャッシュによる性能向上によりメモリ上に記憶する時間が削減されたことによるものである．また，ソフトエラー数はアプリケーションプログラムの種類によって大きく異なることを確認した．さらには，命令キャッシュとデータキャッシュでは，その単体のソフトエラー率がシステム全体のソフトエラー数に与える影響が大きく異なることが分かった．今後は，提案手法を活用し，システム全体の信頼性を向上させる技術，および，一定の信頼性を保障した上で消費電力や性能を改善する設計技術を開発する予定である．

文　献

[1] 上村大樹，戸坂義春，芹澤芳夫，岡秀樹，佐藤成生，“中性子シミュレーションの新展開，” 電子情報通信学会信学技法，ICD2005-19，vol. 105，no. 2，2005 年 4 月．

[2] G. Asadi, and M. B. Tahoori, “Soft error rate estimation and mitigation for SRAM-based FPGAs,” *Proc. ACM International Symposium on Field Programmable Gate Arrays*, pp.149-160, 2005.

[3] N. S. Bowen, and D. K. Pradhan, “The effect of program behavior on fault observability,” *IEEE Transactions on Computers*, vol. 45, no. 8, pp.868-880, 1996.

[4] J. L. Hennessy and D. A. Patterson “Computer architecture: a quantitative approach,” pp. 401–402, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[5] T. C. May and M. H. Wood, “Alpha-particle-induced soft errors in dynamic memories,” *IEEE Transactions on Electron Devices*, vol. 26, pp. 2–7. 1979.

[6] S. S. Mukherjee, J Emer, and S. Reinhardt, “The soft error problem: an architectural perspective,” Proc. IEEE International Symposium on HPCA, pp.243-247, 2005.

[7] M. Rebaudengo, M. S. Reorda, and M. Violante, “An accurate analysis of the effects of soft errors in the instruction and data caches of a pipelined microprocessor,” *Proc. DATE*, pp.10602-10607, 2003.

[8] E. Takeda, D. Hisamoto, T. Toyabe, “A new soft-error phenomenon in VLSIs: the alpha-particle-induced source/drain penetration (ALPEN) effect,” *IEEE IRPS*, pp. 109–112, 1988.

[9] Y. Tosaka, S. Satoh, and T. Itakura, “Neutron-induced soft error simulator and its accurate predictions,” *Proc. IEEE International Conference on SISPAD*, pp. 253–256, 1997.

[10] Y. Tosaka, H. Kanata, T. Itakura, and S. Satoh, “Simulation technologies for cosmic ray neutron-induced soft errors: models and simulation systems,” *IEEE Transactions on Nuclear Science*, vol. 46, pp. 774-780, 1999.

[11] Y. Tosaka, H. Ehara, M. Igeta, T. Uemura, H. Oka, N. Matsuoka, and K. Hatanaka, “Comprehensive study of soft errors in adbanded CMOS circuits with 90/130nm technology,” *IEEE IEDM*, pp. 941–948, 2004.

[12] Y. Tosaka, S. Satoh, and H. Oka, “Comprehensive soft error simulator NISES II,” *Proc. IEEE International Conference on SISPAD*, pp. 219–226, 2004.

[13] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. Patel, “Characterizing the effects of transient faults on a high-performance processor pipeline,” *Proc. IEEE International Conference on Dependable Systems and Networks*, pp.61-70, 2004.

[14] V. Degalahal, S. Cetiner, F. Alim, N. Vijaykrishnan, K. Unlu, and M. J. Irwin, “SESEE: soft error simulation and estimation engine,” *Proc. MAPLD International Conference*, 2004.