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

2008－ARC－179 (30)
2008／8／7

レジスタ・ファイル書き込み時タイミング・エラーに関する脆弱性評価

入江英嗣† 五島正裕†
坂井修一† 平木敬†

本論文では，タイミング・エラーによる脆弱性 (vulnerability factor) に着目した評価を行う．レジスタ・ファイル書き込みに注目し，シミュレーション上でモンテカルロ的にエラーを注入して，エラー脆弱性を計測する．この計測は，エラー脆弱性に関する従来の議論を踏まえつつ，バイパスの考慮や，タイミング・エラーの注入など，新しい試みとなっている．実験の結果，バイパスによって 8 割のエラーがマスクされること，軽量化 WAB により 1 桁から 3 桁の脆弱性削減ができることが分かった．また，そのようにして得た値を指標とした，典型的ケース実行のための設計最適化フレームワークについて議論する．

Vulnerability Evaluation of Timing-errors on Register Write

HIDETSUGU IRIE ,† MASAHIRO GOSHIMA ,† SHUICHI SAKAI †
and KEI HIRAKI †

In this paper, we evaluate architectural vulnerability factor (AVF) focusing on timing errors. We evaluate AVF of register writes by simulating the errors in the Monte Carlo method. Based on conventional discussion of AVF, several novel features such as bypass effects and timing-error characteristics are introduced to the experiment. The evaluation results showed that 80% of the errors are masked by bypassing, and the light weight WAB reduces AVF by 1 or 3 orders of magnitude. We also discuss the design optimization framework using these AVFs for typical case execution.

1. はじめに

プロセスルールが 100nm を切り出した 2002 年頃から，製造バラツキあるいは遅延バラツキと呼ばれる問題がプロセッサアーキテクトに認識されるようになってきた[1)2)]．これは製造，電圧，温度などの要因 (PVT) によって個々のトランジスタのパラメタにバラツキが生じ，スイッチング速度やリーク電流の分布が広がるという問題である．例えば，32nm のプロセスルールでは，チャネル内に含まれるドープ原子は数十個に過ぎず，一つ一つの原子の確率的な振る舞いがトランジスタの性能に大きな影響を与えてしまう[3)]．

動的及び静的な遅延バラツキの最も深刻な影響は，チップ内に含まれる 1G 個ものトランジスタ間のバラツキである．チップ内のトランジスタ全てを安定動作させるためのマージンは膨大なものとなり，微細化一世代分の恩恵を打ち消してしまうと言われている[4)]．

この問題に対するアグレッシブなアーキテクチャ技術として「典型的ケース実行[5)]」が提案されている．典型的ケース実行では，プロセッサを敢えて楽観的な電圧または周波数で動作させ，エラー発生情報をフィードバックして動的にマージンを最適化する．動的に調整する利点は，温度や電圧バラツキなどの，動的な要因に対する効率化である．更に，本来はタイミング違反となる周波数でも，入力値次第で (キャリが発生しない，書き込み値が変わらないなど) 動作可能となるため，バラツキ対策を超えて，「建設的タイミング違反[6)]」のように，従来は踏み込まれなかったマージンをも効率化することが期待できる．

典型的ケース実行では，タイミング・エラーの検出および回復コストの軽量化が課題である．通常，エラー検出には冗長実行が用いられるが，典型的ケース実行では高効率が目標であるため，三重化やタンデム・アーキテクチャ[7)]のように 2～3 倍のコスト投入は難しい．典型的ケース実行を提案した Austin らは，当初シャドウ・ラッチ[8)]のような軽量な検出機構を，一部のクリティカル・パスのみに実装することを構想していた．しかし，設計が最適化されるほど，全てのパスがクリティカルに近づき，大量のシャドウ・ラッチが必要となってしまう．

このため，典型的ケース実行の利点を活かすためには，どのユニットのエラーをどの程度までカバーする

† 東京大学大学院 情報理工学系研究科
Grad. School of Info. Science and Technology, The University of Tokyo

かという，設計上の取捨が必要になる．我々は，この取捨をエラー脆弱性☆(vulnerability factor) に着目して行うフレームワークを考えている．本論文では，レジスタ・ファイル書き込みに関するエラーに着目し，モンテカルロ的なエラー注入実験によって評価を行う．また，この評価をふまえてフレームワークに関する議論を行う．エラー脆弱性の議論は，宇宙線による SEU については行われていたが，我々の知る限り，タイミング・エラーについてはまだ行われていない．

以下，本論文は次のように構成される．第 2 節ではレジスタ・ファイル書き込みの傾向について述べ，メモリセルに書き込まれたものの，読み出されない値が多いことを示す．第 3 節では，エラー致命率の定量化を行った先行研究について紹介し，それらの研究と，本論文での実験との差異について述べる．第 4 節は実験方法を，第 5 節でその結果を示す．第 6 節では，このようにして得た値を指標とした，典型的ケース実行アーキテクチャの設計フレームワークについて述べる．第 7 節で関連研究について述べ，第 8 節でまとめを行う．

2. 物理レジスタアクセスの傾向

プロセッサ信頼性において，一般にレジスタ・ファイルは重要ユニットと考えられている[9]．レジスタ・ファイルはアーキテクチャステートに直結し，また投機ミス時やエラー検出時には回復の基点となる．このような重要性から，SPARC64VI ではレジスタ・ファイルにパリティを付加し，IBM G5 では ECC を付加している[10]．

ところが，以下に示すデータは，書き込みエラーによって生じた値が，その後読み出される確率は低いことを示唆している．**図 1** は，レジスタ値が生成されてから参照されるまでのサイクル数を，累積グラフによって示したものである．プロセッサシミュレータ鬼斬[11]の改良版である鬼斬弐を用い，SPECint2006 ベンチマーク 429.mcf について，先頭 100M 命令をスキップした後，続く 100M 命令を計測している．シミュレーションに用いたプロセッサ緒元を**表 1** に示す．なお，サイクル数の数え方は「値を生成する命令の最後の EXE ステージ」から「値を使用する命令の最初の EXE ステージ」となっており，最短で 1 である．

このグラフからは，レジスタ値の過半数は，生成された次のサイクルで既に最後の参照を終えてしまうことが分かる．ここで，図 1 中，2 サイクル目と 4 サイクル目に引いてある縦の点線は，それぞれレジスタアクセスレイテンシを 1 あるいは 2 サイクルとした場合の，バイパス期間である．この期間内に最後の読み出しが終了する場合，実はレジスタ・ファイルのメモリセルへ書き込まれた値はその後参照されないまま上書きされることになる．このような，バイパス期間で完結して

☆ セキュリティ用語との混同を避けるための本論文での造語

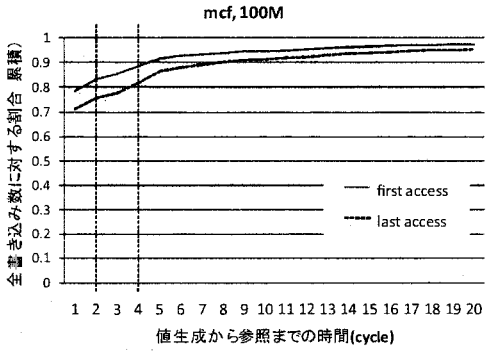

図 1 レジスタ値生成から参照までのサイクル数
Fig. 1 Cycle Times between Generation and Reference of a Register Value

いるレジスタ値は，429.mcf の場合 70%を越えていることが分かる．バイパス期間の読み出しは，バイパス・ネットワークあるいはバイパス・バッファ[12]を介して行われる．これらのユニットはバッファ・サイズが小さいこと，あるいは主な遅延源が配線であることから，遅延バラツキの影響を受けにくい．少なくとも，バイパス部分の保護はレジスタ・ファイルの保護とはまた異なる技術の議論となる．これ以降の評価では，バイパス部分はエラーフリーと仮定する．

割合の値はプログラムによって大きく変化するが，SPEC2000 および SPEC2006 の全アプリケーションを通して，上に述べたような傾向は共通している．この傾向は，レジスタ・ファイルのメモリセルの実際の利用率が高くないことを示唆している．

もし，レジスタ・ファイルのメモリセルで生じるエラーの影響が軽微であるならばレジスタ・ファイル保護は軽量化できる．しかし，重要ユニットであるため，低確率といえどもレジスタ値破壊が生じた場合に，それがどの程度致命的であるかは自明ではない．この影響を定量化する議論が，エラー脆弱性である．

3. エラー脆弱性

3.1 エラー致命率に関する先行研究

プロセッサの回路レベルで生じたエラーは，必ずしもシステムに致命的な事態を引き起こすとは限らない．前節で挙げた，読み出されないメモリセル値のように，プロセッサには，回路レベルエラーをマスクする要素が備わっている．マスク要素としては，しばしば挙げられる NOP 命令や，busy でない演算器への入力の他，様々なものがある．このようなマスク要素に注目して，エラーがどの程度の確率で致命的なものへプロパゲートするかを調べる調査がいくつか行われている．

Wang ら[9]は，Alpha21264 相当のプロセッサを RTL レベルで記述し，モンテカルロ的に SEU(宇宙線による

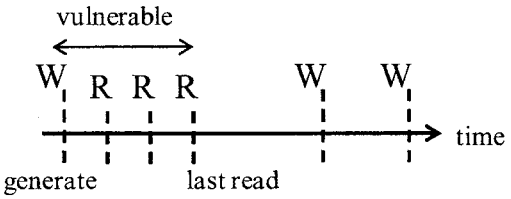

図 2 RVF

1bit の反転) を注入するシミュレーションを行った. ランダムに選んだ実行ポイントで, ランダムに該当ラッチ (またはフリップ・フロップ) を選んで値を反転させ, その後 10000 サイクルにわたる実行を観測するというトライアルを繰り返している. 彼らは, ランダムに選ばれた反転箇所がレジスタ・ファイル中だった場合, 6 割はマスクされ, 4 割の確率で Silent Data Colluption(実行停止には至らないが, 正常な実行とはステートが異なっている状態) に至ると報告している. また, 実行停止に至るケースはほとんどないとしている. 彼らがバイパスをどうモデル化していたかについては記述が明確でないため不明である.

一方, プロセッサに生じるエラーの影響を解析的手法で定量化した研究として, Mukherjee ら[13] の "Architectural Vulnerability Factor"(AVF) が挙げられる. AVF は, プロセッサ内部で生じたエラーがプログラム出力に影響する確率であり, ユニットごとに異なる値を持つ. 例えば, 分岐予測器に生じるエラーは分岐予測を失敗させることはあっても, 実行結果には直接影響しないため, AVF は 0 となる. 一方で命令ウィンドウのようなユニットは 50%に近い高い AVF 値を持つ. 彼らの論文でも, エラー源として SEU が想定されている. 正確な AVF を求めることは困難であるため, Mukherjee らは近似手法によって AVF を求めている.

また, Yan ら[14] は, 物理レジスタ・ファイルの AVF について, Mukherjee らの手法では値の寿命が考慮されていないとして, RVF(Register Vulnerable Factor) を提案している. レジスタ・ファイルに生じるエラーが致命的となるのは定義から最後の参照までの期間であり (**図 2**), この期間の割合を計測して RVF を求め, レジスタ・ファイルに生じる SEU がプログラム出力に影響する確率は 13.8%であるとしている.

3.2 本論文における計測

本論文において, 次節以降で述べるタイミング・エラー AVF 計測は, 以下のような点で先行研究と異なっている. i) まず, 書き込みタイミング・エラーに注目しているため, エラーの傾向が異なる. どの時点でも起こりうる SEU とは異なり, 書き込みタイミング・エラーはレジスタアップデート時に生じ, RVF 的には常にクリティカルな期間で生じる. また, 書き込みタイミングエラーは反転エラーであり, 前の値から反転するビットについて生じる. このため, 宇宙線エラーに比べると

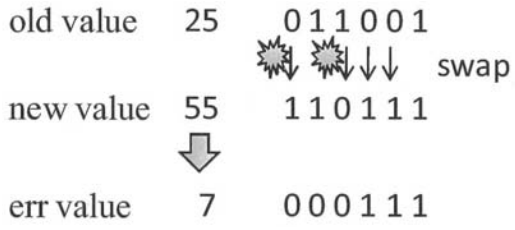

図 3 エラー値のモデル
Fig. 3 Generating Errored Value

容易に複数 bit エラーとなる.

ii) 次に, 解析的な手法ではなく, 実際に注入を行って影響を調べている. これは, モデルが意図していない要素の影響が隠ぺいされてしまうことを防ぐためである. 実際, 今回注目しているレジスタ・バイパスは, RVF の議論に含まれているべきだった要素である (図 2b). 本論文では, エクセキューション・ドリブン・シミュレータである鬼斬 2 に対して, サイクル・アキュレートにエラー注入を行っており, より正確なエラー発生後の挙動を再現している.

4. 評価方法

4.1 エラー注入方式

プロセッサ・シミュレータ鬼斬 2 を用いて, モンテカルロ的にレジスタ書き込みエラーの注入を行い, エラー挙動を計測した. レジスタ書き込み時に 1/5000 の確率でエラーが生じ, 本来とは異なる値がレジスタに書き込まれる. この誤り値は, 書き込まれる前の値と書き込まれる値を比較し, ビットが反転している桁について, 50%の確率で反転が失敗することによって生成される (**図 3**)☆. 1/5000 という値には意味はなく, エラー注入回数に対する伝搬率や停止率が重要である.

SPECint2006 各ベンチマークについて先頭 1G 命令をスキップした後にエラー注入トライアルを開始する. エラーが一定間隔で注入され続けることにより, いずれかターミネートエラーを引き起こして, トライアル 1 回分は終了する. ターミネートしなかった場合は最大 100M 命令まで実行する. 今回は, このトライアルを各モデル各ベンチマークについて 10 回繰り返して平均値を得た.

実験は, 以下に示すモデルについて行った.

conventional バイパス期間中の読み出しにもエラーが伝わる従来モデル

bypass バイパス期間を考慮したエラー伝搬モデル

naive discard WAB 軽量版 WAB(後述) によってエラー回復をするモデル

実験に用いたベースラインプロセッサのパラメタは第

☆ この際, 反転桁のないサイレント・ライトのケースを除き, 必ず 1 桁以上の反転失敗が生じる設定とした.

表 1　ベースラインプロセッサ緒元
Table 1　Microarchitectural Parameters for Baseline Processor

フロントエンド	4way, 7cycle
命令セット	Alpha ISA
命令ウィンドウ	32 entries
LSQ	16entries
機能ユニット	4 iALU, 1 iMUL/DIV, 2 LD/ST, 1 fpADD, 1 fpMUL/DIV/SQRT
レジスタ・ファイル	i162 entries, f64 entries, 2cycle
L1 I/DCache	32KB, LRU, 4way, 64B line, 3cycle latency
L2 Cache	4MB, LRU, 8way, 64B line, 15cycle latency
memory access	200cycle

2 節と同様, 表 1 を用い, レジスタ参照レイテンシは 2 サイクルとした.

4.2　評価に用いた軽量版 WAB

Write Assurance Buffer(WAB)[15] は, 我々が提案したレジスタ・ファイル書き込み保証手法である. 書き込み値を WAB に複製しておき, その後レジスタから読み出された値との一致比較を行う. 検証読み出しには, 後続命令によるレジスタ・オペランド読み出しを利用する (passive WAB). この技術のポイントは, i) 検証待ち値のみを保持する WAB は本体レジスタ・ファイルよりも小さくて良い点, ii) エントリ数の差をマージン差としてエラーフリーバッファを得る点, である.

現在, 改良版の WAB は, 1 割前後の性能低下と引き替えに, 全てのレジスタ書き込みをチェックすることができるが, レジスタ読み出し回数を 50%程度増加させてしまう☆. そこで, 本論文では, 100%のカバレージを諦め, かわりにコストを軽量化した naive discard WAB を提案し, エラー脆弱性への寄与を調べた.

naive discard WAB のモデルは単純である. 従来の WAB ではストールして検証を進めていた, WAB エントリ・フルのケースについて, ストールをせず, そのまま新しい値で上書きしてしまう. 検証の機会を得ないまま上書きされた値は当然未チェックのため, エラー脆弱性となる. このエラー脆弱性の量が, 実用上の焦点となる. naive discard WAB では, ストールによる性能低下および追加のレジスタ読み出しは 0 である. また, WAB エントリの nearly full でストールを予測する必要がないので, 制御は単純で, WAB エントリの利用効率も高くなる.

5. 評価結果

5.1　エラー伝搬率

バイパスを考慮しない場合 (conventional) と, バイパスを考慮した場合 (bypass) について, エラー伝搬率を**図 4** に示す. 横軸が各ベンチマークを示し, 縦軸は, 発生したエラー値の何割が, 読み出されて伝搬していったかを示している. なお, 平均は相加平均のため, 伝搬率が高い側 (最悪値側) へ寄る傾向となっている. 物理レジスタ・ファイルは, バイパスを考慮しない場合は伝

☆ 初期の WAB のほぼ 100%の増加からはだいぶ改善している

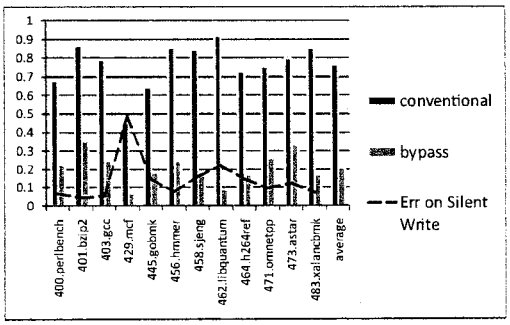

図 4　レジスタ書き込みエラーの伝搬率
Fig. 4　Propagation Ratio of Register Write Error

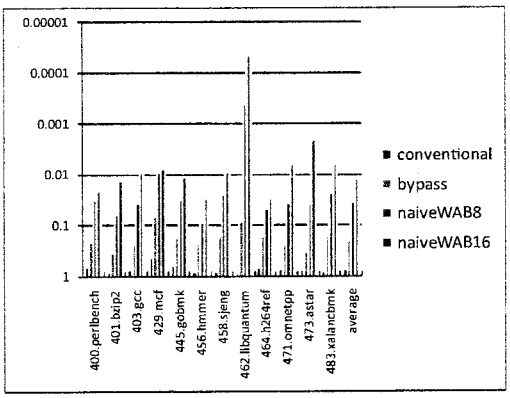

図 5　WAB による伝搬率削減
Fig. 5　Propagation Ratio Reduction by WAB

搬率 80%を越える致命的なユニットとなっているが, バイパスを考慮すると実は 80%を越えるエラーについてマスクする効果を持っていることが分かる. SEU に関する既存研究では, レジスタ・ファイルの脆弱性を過大に見積もっていた可能性がある.

物理レジスタが既に保持している値と, これから書き込もうとしている値が同じだった場合, 反転操作がないため, 仮にタイミング違反が生じていても, 書き込みは成功する. グラフ中の破線は, そのようなサイレント・ライトへのエラーが, 全エラー中でどの程度を占めていたかを示している. サイレント・ライトの頻度は, 429.mcf が 50%と突出しており, これが伝搬率の低さに影響している. その他のベンチマークについては, サイレント・ライトと伝搬率の間に顕著な相関は認められない.

次に, 検出漏れを容認することによってコストを抑える, naive discard WAB の伝搬率評価を**図 5** に示す. 縦軸は対数軸を用いた. 伝搬率はそれぞれ 8 エントリの時に約 1/6, 16 エントリの時に 1/18 となっている. なお, 調和平均では 2 桁および 4 桁の改善となった. 性能及び電力コストとのトレードオフを考えれば, 有力

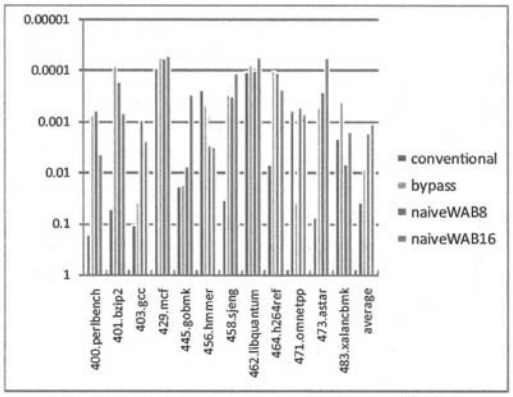

図 6 レジスタ書き込みエラーによるプログラム停止率
Fig. 6 Termination Ratio of Register Write Error

な選択肢と言える. 単純比較はできないが, 先に挙げた Wang らの研究では, レジスタ・ファイルへの ECC 適用による伝搬率の改善は 1/4 程度と報告されている.

5.2 プログラム停止率

図 6 はエラー発生回数に対するプログラム停止率を示している. この停止率が, ユーザが実際に観測するタイミング・エラーの挙動である. 実験前は, この値は伝搬率と相関した値を示し, サイレントエラーの割合を計るための指標となると考えていたが, 実験結果は予想と異なるものとなった. 我々は, この理由は以下の 2 点によると考えている. i) プログラム停止に関しては, 試行回数が 10 と, モンテカルロ手法として非常に少なく, まだ意味のあるデータとなっていない. ii) プログラムが整合性チェックなどを実行した時点で停止している.

保護手法によるエラー伝搬率の変化と関係なく, 同じ実行ポイントで停止するケースが多いことからも, 後者が大きな影響を持っていると考えられる. Wang らの先行研究では, レジスタ・ファイルへのエラーは殆どプログラム停止を引き起こさないと報告されているが, これはエラー後のモニタリングが 10000 サイクルと短く, 今回の実験のように整合性チェックルーチンの実行をまたがなかったと予想される. 低確率の中伝搬したエラー値は, その後の実行を歪めてしまうが, その際に, セグメンテーションフォールトなどで停止することは希で, プログラム実行はエラーを抱えたまま続いてしまうことが分かる. 401.bzip2 や 473.astar などは高い伝搬率にも関わらず, 停止率は比較的低くなっている. 冒頭で, 信頼性向上技術を各階層で組み合わせることを述べたが, プログラムによる整合性チェックは書き込みタイミング・エラーに対しても有効であることを示しているグラフとなっている.

表 2 レジスタ書き込み保証の選択肢
Table 2 Alternatives for Register Write Assuarance

	コスト (性能)	コスト (資力)	エラー脆弱性
保護なし	0	0	20.6%
naive discard WAB 8 エントリ	0	CAM8 エントリ	3.7%
naive discard WAB 16 エントリ	0	CAM16 エントリ	1.2%
conservative WAB 16 エントリ	2.6%	CAM16 エントリ+ レジスタ読み出し 65%増	0

6. 典型的ケース実行プロセッサ設計への適用

前節までのようにして, 命令の各パイプライン処理のエラー脆弱性や, 保護技術を適用した場合の改善率を定量化することができる. この指標を用いれば, 次のような設計が可能となる.

i) まず, タイミング・エラー保護にかけるコストの総量を決定する. コスト指標には, トランジスタ数や, ED 積低下量などを用いる. コスト総量は, 動的なマージン削減によって得られるゲインよりも少なく設定する. ii) 次に, プロセッサのタイミング・エラー脆弱性が最も少なくなるように, 各処理に保護コストを分配する. プロセッサのタイミング・エラー脆弱性は, 各処理の脆弱性を重みつき加算したものとなる. iii) このようにして得られたプロセッサについて, 所望のエラー率☆に収まるように, 電圧および周波数のフィードバック制御を行う.

ii) について, 例えばレジスタ書き込みであれば, 第 5 節の結果を用いて, **表 2** のような選択肢が得られる. この場合, かけたコストに応じて脆弱性が改善するため, 他の処理にコストをかけるべきか否かが, naive discard WAB 16 エントリや conservative WAB を採用する判断材料となる. 保護コストの使い方の例として, 前述のシャドウ・ラッチや WAB の他, カナリア[16], シャドウ・センスアンプ[17] や, パイプライン細分化によるマージン確保が挙げられる.

7. 関連研究

遅延バラツキ対策アーキテクチャとして, タンデム・アーキテクチャを用いた手法も提案されている[18]. この技術では, 先行するコアが, オプティミスティックな周波数でプリフェッチを行い, 追随するコアが遅めの周波数で動作してタイミング・エラーをチェックする. タンデム・アーキテクチャは, 冗長実行を前提とすれば効率の良い方式だが, 電力消費は 200%をスタート地点として, そこからどれだけ減らせるかという議論であり, コストはやや高くなる. また, Teodorescu ら[19] は, チップ内バラツキを考慮し, マルチコアを, 同一コアではなく電力消費やスピードの異なるヘテロコア構成とみて, タスク割り当てを最適化する手法を提案し

☆ IBM Power4 の SDC ターゲットであれば 114FIT

ている.

8. まとめ

本論文では、典型的ケース実行アーキテクチャの効率化について、エラー脆弱性に着目した議論を行った. ケース・スタディとして, レジスタ・ファイル書き込みに注目し、シミュレータ上でランダムにタイミング・エラーを注入して挙動を観測した.

実験結果からは, 書き込みエラーの 8 割はバイパスによってマスクされ伝搬しないことや, 検出漏れを容認する代わりに軽量化を行った naive discard WAB ではエラー脆弱性を 1 桁あるいは 2 桁改善できるなどの定量的議論が可能となった. なお 従来のエラー脆弱性の議論では、レジスタ・ファイルのエラーについて、メモリセル部分とバイパス部分のアクセスを分けておらず, メモリセル部分のエラー脆弱性を過大に見積もっていた可能性がある.

更に, これらの評価をふまえて, 典型的ケース実行アーキテクチャの設計フレームワークについて提案を行った. このフレームワークでは、定量化されたエラー脆弱性に注目することにより, 限られたコストの中で, 適切に保護手法を選択する.

今後の課題として、i) 他の処理についても同様に脆弱性を評価し、フレームワークを進めていくこと, ii)discard を行う軽量 WAB について、コンパイラ協調などにより脆弱性を改善すること, iii) トライアル数を増やし, 停止率について精度の高い評価を行うこと, が挙げられる.

参考文献

1) Borkar, S., Karnik, T., Narendra, S., tschanz, J., Keshavarzi, A. and De, V.: Parameter variations and impact on circuits and microarchitecture, *Design Automation Conference*.
2) 岡田健一: 集積回路における性能ばらつき解析に関する研究, 京都大学博士論文 (2003).
3) Borkar, S.: Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation, *IEEE Micro*, Vol. 25, No. 6, pp. 10–16 (2005).
4) Bowman, K., Duvall, S. and Meindl, J.: Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration, *IEEE Journal of Solid State Circuits*, Vol. 37-2.
5) Austin, T., Blaauw, D., Mudge, T. and Flautner, K.: Making Typical Silicon Matter with Razor, *IEEE Computer* (2004).
6) 谷野亜沙美, 佐藤寿倫, 有田五次郎: 建設的タイミング違反方式に基づく ALU の HDL 設計とその評価, 信学技報 ICD2002-212 (2003).
7) Rotenberg, E.: AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessor, *Symp. on Fault-Tolerant Computing*, pp. 84–91 (1999).
8) Ernst, D., Kim, N., Das, S., Pant, S., Pham, T., Rao, R., Ziesler, C., Blaauw, D., Austin, T. and Mudge, T.: Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation, *Int. Symp. on Microarchitectre* (2003).
9) N.Wang, Quek, J., Rafacz, T. and Patel, S.: Characterizing the Effects of Transient Faults on a High-Performance Processor Pipeline, *Int. Conf. on Dependable Systems and Networks*.
10) Slegel, T., III, R. A., Check, M., Giamei, B., Krumm, B., Krygowski, C., Li, W., Liptay, J., MacDougall, J., McPherson, T., Navarro, J., Schwarz, E., Shum, K. and Webb, C.: IBM's S/390 G5 Microprocessor Design, *IEEE Micro*, Vol. 19.
11) 渡辺憲一, 一林宏憲, 五島正裕, 坂井修一: プロセッサ・シミュレータ「鬼斬」の設計, 先進的計算基盤システムシンポジウム 2007 ポスター, pp. 194–195 (2007).
12) 三輪忍, 一林宏徳, 入江英嗣, 五島正裕, 富田眞治: 小容量 RAM を用いたオペランド・バイパスの複雑さの低減手法, 情報処理学会論文誌 コンピューティングシステム, Vol. 48, No. SIG13.
13) Mukherjee, S., Weaver, C., Emer, J. and Austin, T.: A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor, *Int. Symp. on Microarchitectre*, pp. 29–42 (2003).
14) Yan, J. and Zhang, W.: Compiler-guided register reliability improvement against soft errors, *Int. conf. on Embedded software*, pp. 203–209 (2005).
15) 入江英嗣, 杉本健, 塩谷亮太, 渡辺憲一, 五島正裕, 坂井修一: パッシブ WAB の改良による低コストなレジスタ書き込みエラー検出手法, 電子情報通信学会研究報告 CPSY2008-3, Vol. 108, No. 1, pp. 13–18 (2008).
16) Sato, T. and Kunitake, Y.: A Simple Flip-Flop Circuit for Typical-Case Designs for DFM, *Int. Symp. on Quality Electronic Design*, pp. 539–545 (2007).
17) Karl, E., Sylvester, D. and Blaauw, D.: Timing Error Correction Techniques for Voltage-scalable On-chip Memories, *IEEE Int. Symp. on Circuits and Systems*, Vol. 4, pp. 3563–3566 (2005).
18) Greskamp, B. and Torrellas, J.: Paceline: Improving Single-Thread Performance in Nanoscale CMPs through Core Overclocking, *Int. conf. on Parallel Architectures and Compilation Techniques*.
19) Teodorescu, R. and Torrellas, J.: Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors, *Int. Symp. on Computer Architecture*.