情報処理学会論文誌 Vol. 49 No. 6 2016–2028 (June 2008)

典型的ケース設計のための レジスタ書き込み保証アーキテクチャ

入江英嗣[†1,*1] 杉本健[†2]
五島正裕[†2] 坂井修一[†2]

近年のプロセッサでは，製造ばらつきや動作時の温度ばらつきの激化により，十分なマージンを見込んで設計することが困難になりつつある．このため，偶発するタイミング・エラーを動的に検出・回復するためのマイクロアーキテクチャ技術が研究されるようになってきた．本論文では，アーキテクチャステート保護の要となる，レジスタ・ファイル書き込み時のタイミング・エラーに注目する．提案手法では，命令の実行結果を書き戻すときに小容量のバッファ（WAB）にも同じ値を保持し，その後双方を読み出して一致比較を行い，書き込みを検証する．検証読み出しは，後続命令の実行を妨げないように行われる．シミュレーションを用いた評価を行い，提案手法では，バッファ容量として 16 エントリを見込めば，性能低下をほとんど起こさずにタイミング・エラー検出を行えることを確認した．

Microarchitectural Register Writing Assurance for Typical-case-designing

Hidetsugu Irie,[†1,*1] Ken Sugimoto,[†2]
Masahiro Goshima[†2] and Shuichi Sakai[†2]

Recently, it has been getting unrealistic to design microprocessors with sufficient margins because of increasing process and temperature variability. Thus, microarchitectural techniques that dynamically detect and recover from timing-errors have come to be researched. In this paper, we focus on timing-errors in register-file writing. Our technique writes execution-results to the small buffer (WAB) in addition to the register writing. Then it verifies the writing by comparing the relevant value from the register-file and WAB. Verifications are performed in the way not to disturb the execution of succeeding instructions. The simulation result showed that the proposal technique detects timing-errors without significant performance degradation when WAB of 16-entries is provided.

1. 序　　論

今日，コンピュータ・システムは社会の隅々にまで浸透しており，その中核となるマイクロプロセッサには非常に高い信頼性が求められている．

マイクロプロセッサの信頼性を脅かす要因のうち，今後深刻となると予想される新しい課題の 1 つに，**動的なタイミング・エラー**がある．動的なタイミング・エラーとは，回路遅延の動的な変化によって信号のタイミングに齟齬が生じ，設計者の想定外の動作が生じる過渡故障である．

従来，タイミング・エラーは，設計/製造上の，いわば静的な問題であり，マージンを見込んだ設計と出荷検査によって対策されてきた．しかし，次章で詳しく述べるように，今後の LSI 製造プロセスでは，微細化とともに，回路遅延のばらつきが激化することが予想されている．さらに，動作時の温度ばらつきも激化しており，遅延ばらつきの増加を招いている．一方で，消費電力の制約は厳しく，大きなばらつき幅に対して一律に安全なマージンを確保することが難しくなりつつある．このような背景から，超微細化プロセスで製造されるマイクロプロセッサの，高効率性と高信頼性を両立させることは，デバイス，回路，マイクロアーキテクチャの各分野に共通する重要課題となっている．

この課題に対するアプローチの 1 つとして，タイミング・エラーを動的に検出/回復させるマイクロアーキテクチャを，楽観的なマージンで動作させる手法が考えられる．このようなアーキテクチャでは，例外的に生じる動的タイミング・エラーに対してはセーフティネットを用いて安定動作を保証し，普段の実行では不要なマージンを削って効率的に動作する．このようなアプローチを可能とするプロセッサ技術として，**RAZOR**[1),2)] や**カナリア・フリップフロップ**[3)] が提案されている．

しかしこれらの技術は，プロセッサの構成要素の中でもロジック部分に対する対策であり，レジスタ・ファイルのようなメモリ回路で構成されるユニット，特にその書き込み処理についてそのまま適用することができない．レジスタ・ファイルはアーキテクチャステートを保

†1 独立行政法人科学技術振興機構
Japan Science and Technology Agency
†2 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
*1 現在，東京大学大学院情報理工学系研究科
Presently with Graduate School of Information Science and Technology, The University of Tokyo

持し，エラー発生時の回復の基点となる．もし，レジスタ・ファイルの書き込み処理時にタイミング・エラーが発生すれば，任意の値が書き込まれ，実行プロセスは暴走してしまう．

そこで本論文では，レジスタ・ファイル書き込み時タイミング・エラーの検出/回復を行うアーキテクチャ手法を提案する．提案手法では特にスーパスカラ・プロセッサを対象として想定している．これは，最新デバイス寸法で製造され，また用途的にも製造ばらつきや熱の影響が深刻と考えられるためである．

提案手法の基本は，単純なライト・ベリファイである．すなわち，レジスタ・ファイルに書き込んだ値を再び読み出して一致比較することにより，書き込みの正しさを確認する．検証読み出しは，命令実行を妨げないように，リードポートに空きがあったときなどに行われ，このため提案手法による検証が性能（IPC）に与える影響はごくわずか（1%以下）である．

以下，本論文は次のように構成される．まず，2 章では，タイミング・エラーと関連研究についてまとめ，続く 3 章で提案手法の概要について述べる．4 章では，レジスタ・ファイル書き込みについて詳細に議論し，タイミング・エラーによってレジスタ・ファイルに何が起こりうるのかを明らかにする．5 章では提案手法の実装について詳しく説明し，続く 6 章でシミュレータを用いた評価の結果を示す．

2. タイミング・エラー

2.1 設計，製造とタイミング・エラー

同期回路における典型的なタイミング・エラーとしては，クロック期間内に回路の出力が間に合わず，古い出力がサンプリングされることがあげられる（**図 1**）．タイミング・エラーに関して，従来の設計，製造は，以下のように進められる．回路遅延が設計に対して変動する要因は，主に，電源電圧，温度，製造ばらつきの 3 つである．具体的には以下のようなものがあげられる：

(1)　製造時のばらつき
　(a)　露光精度の低下によるゲート長，ゲート幅，配線幅のばらつき
　(b)　不純物密度分布のばらつき
　(c)　エッチングや平坦化などの工作精度の低下による配線高さのばらつき
(2)　動作時の消費電流に起因するばらつき
　(a)　IR ドロップによる電源電圧のばらつき
　(b)　温度分布のばらつき

設計時には，まず，それぞれについて許容範囲を設定し，すべての動作条件 —— 特に最悪条件において正しく動作するように回路を設計する．それでも，製造ばらつきが許容範囲外であるようなチップが製造されることは原理的に避けられない．そのようなチップは出荷検査によって取り除かれる．

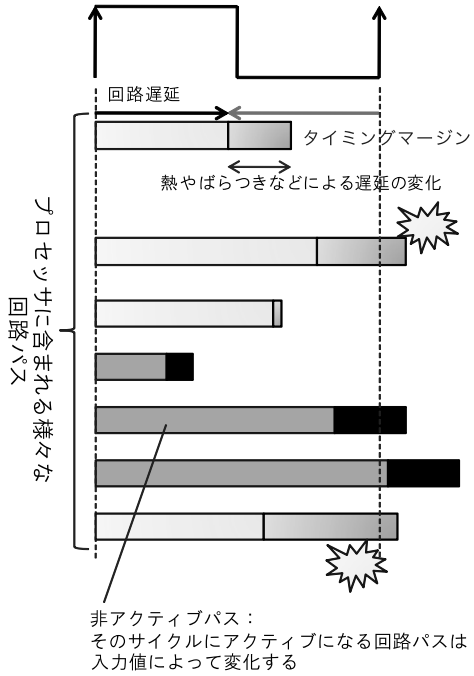


図 1　回路遅延，タイミング・マージン，ばらつきとタイミング・エラーの関係の概念図
Fig. 1　Conceptual diagram of the relationship between circuits delay, timing-margins, variations and timing-errors.

したがって製品出荷後には，電源電圧と温度が設定された許容範囲内に収まっている限り，動的なタイミング・エラーはほとんど発生しないことになる．動的なタイミング・エラーとしては，冷却ファンの故障などに起因する熱暴走がほとんど唯一の例であった．そのため従来，「タイミング・エラー」といえば，設計段階において，与えられたタイミング制約を満たせないことを指した．

2.2 設計/製造時のばらつき対策技術

しかし近年では，製造ばらつき，特に不純物密度分布のばらつきによって，各トランジス

タの閾値がばらつくことの影響が深刻であり，チップ内ばらつきの激化[4]として顕著に現れている．これは，LSI の微細化によって，トランジスタや配線が原子のサイズに近づいたことによるものであって，原理的に防ぐことが難しい．

ばらつきが増大していくと，従来の最悪値に基づく設計手法は悲観的になりすぎることとなる．パスを構成するトランジスタ，配線がすべて最悪であるような確率はきわめて低い．そこで，ランダムなばらつきを統計的に取り扱う設計手法[5]が確立されつつある．また，諏佐ら[6]は，製造後のばらつきに応じて，各チップのクロック供給回路の遅延設定を調整し，歩留まりを向上させる技術を提案している．

本論文で注目しているレジスタ・ファイルに関して，Liang ら[7]は，製造ばらつきに応じてレジスタ参照にかけるサイクルを可変させる，タイミング・エラー予防手法を提案している．この手法では，BIST（Built-In Self-Test）時にレジスタ・ファイルをチェックして遅延の大きいエントリを検出し，そのエントリへの参照時にはバブルを挿入して通常よりも余分にサイクル数をかける．また Liang ら[8]は，レジスタ・ファイルとしてタイミング・エラー耐性の高い DRAM セルを用いるアーキテクチャを提案している．

2.3　動的なタイミング・エラー検出による最適実行

しかしながら，統計的なばらつき対策をふまえてもなお，回路遅延のばらつきは，温度やその時点でアクティブなパスなどの，動作時の状況に大きく影響されるため，マージン設定が難しい．

そこで，実行中にあえてタイミング・エラーを起こさせ，動作周波数にフィードバックすることにより，その時点の最適な動作周波数を得るアプローチが提案されている[2]．このアプローチでは，タイミング・エラーの動的な検出/回復機能が前提となる．

フィードバックは，**DVFS**（Dynamic Voltage and Frequency Scaling）技術を用いて，たとえば以下のように行えばよい．まず，電圧を一定に保ったまま動作周波数を徐々に高めていくと，いずれタイミング・エラーが発生することになる．そこで，タイミング・エラーを検出したら，動作周波数を（少し）元に戻し，処理をやり直す．同様に，周波数を固定したまま電圧を必要最低値に収束させることも可能である．

このようにすると，以下のような，2 段階の効果が得られる．第 1 に，見積りではない，その個体が実際に持つ最高動作周波数での動作が可能になる．

第 2 に，その時点でアクティブとなるパスによって定まる最大動作周波数に合わせた最適化が可能となる．実際には，システムのクリティカル・パスのすべてが毎サイクル機能しているわけではない．たとえば加算器のキャリー生成回路では，実際にキャリーが長い距離を伝播しなければ，非常に短い時間で演算を終えることができる．

したがって，このような手法を用いれば，チップ全体のクリティカル・パスではなく，実際にほぼ毎サイクル機能する中で最もクリティカルなパス —— いわば，動的なクリティカル・パスによって決まる最高動作周波数での動作が可能となる．

この議論は，非同期設計と比較するといっそう興味深い．標準的な 2 線 2 相式回路では，クロックではなく，回路の実際の遅延に基づいてラッチへのサンプリングが行われる．したがって，上述の第 1 段階の最適化は自然に実現できることになる．しかし非同期設計では，第 2 段階の効果を得ることはできない．先のキャリー生成回路の例では，2 線 2 相式回路では，キャリーが伝播されなかった場合でも，されなかったことがラッチまで伝わる必要がある．

このように，タイミング・エラーを動的に検出・回復する技術は，単純に回路の信頼性を高めるだけでなく，従来踏み込まれなかった部分への最適化を可能とする技術である．

2.4　既存の動的タイミング・エラー検出技術

エラー検出は基本的に冗長実行によって可能となる．しかし，タイミング・エラー検出を目的とする場合，単純な空間的あるいは時間的冗長実行では不十分である．これは同程度のタイミング・マージンを持つ回路で冗長実行を行っても，双方が同じタイミング・エラーを起こして同じ値を出力する可能性が無視できないためである．

このため，タイミング・エラー検出においては，タイミング・マージンの異なる回路による冗長化が必要である．タイミング・マージンに十分に差があれば，マージンの少ない回路の方が先にエラーを起こすと考えることができる．

DIVA

Austin らが提案した **DIVA** アーキテクチャ[9]は，プロセッサの実行コア全体に対する検証用ロジックとしてチェッカ・コアを用意し，それぞれ同じ命令を実行することで，設計ミスから実行中の過渡故障に至るまでの，プロセッサに生じる様々なエラーを幅広く検出する．

DIVA アーキテクチャを用いれば，チェッカ・コア側のタイミング・マージンを大きく設定することによって，タイミング・エラーの検出が可能になると考えられる．タイトなスケジューリングや複雑なデータパスをともなう実行コアと異なり，チェッカ・コアは自然にマージンが備わっている可能性が高い．また，チェッカ・コアのパイプライン段数を増やしてマージンを増やすことはそれほど困難ではない．

ただし，例外はレジスタ・ファイルである．DIVA アーキテクチャでは，レジスタ・ファイルはチェッカ・コアと実行コアとで共有されており，その正しさは ECC で保証されると

している．しかし ECC は書き込み時のエラーを想定した技術ではなく，複数ビットに任意の値が書き込まれるような，書き込み時タイミング・エラーには無力である．また，DIVA では，冗長実行時のパフォーマンスを維持するために，レジスタ・ファイルのポート数を増加させており，この面でもタイミング・エラー耐性を弱めている．

RAZOR

Austin らは，また，回路レベルに近い技術として，**RAZOR** を提案，試作し，評価している[2]．RAZOR ではタイミング・クリティカルなパスに対して，**シャドウ・フリップフロップ**を付加する．ロジックの出力は，通常のクロックで動作するフリップフロップと，やや位相の遅れたクロックで動作するシャドウ・フリップフロップの双方にそれぞれ取り込まれる．ロジックの遅延の増加のために通常のラッチで古い値が取り込まれても，遅れた位相で動作するシャドウ・ラッチでは正しい値がサンプリングされる可能性が高い．そのため，双方の出力を比較することによりタイミング・エラーを検出できる．

しかし，レジスタ・ファイルをはじめとする RAM に対しては，この手法は適用できない．SRAM の個々のセルの中にシャドウ・フリップフロップを用意することは現実的ではない．シャドウ・フリップフロップには，6T セルを超えるコストがかかるからである．ただし，RAM の読み出し処理に関しては次に述べるような手法が存在する．

レジスタ・ファイル読み出し時のタイミング・エラー対策

レジスタ・ファイルの読み出し時のタイミング・エラー検出に関しては，以下の 2 つの研究がある．Karl らは，SRAM のセンスアンプを二重化し，やや遅れて動作する**シャドウ・センスアンプ**を追加することでレジスタ読み出し時のタイミング・エラーを検出する手法を提案している[10]．また我々は，相補的ビットラインのそれぞれにシングル・ビットライン用のセンスアンプを取り付ける方法を提案している[11]．値を読み出した後にビットラインの値が相補的になっているかを確認することでタイミング・エラーを検出できる．

タイミング・エラーの動的予防技術

佐藤らは，ロジックにおけるタイミング・エラーを予防する**カナリア・フリップフロップ**を提案している[3]．この手法では，RAZOR 同様，タンデムとなったフリップフロップの値を比較するが，2 つのフリップフロップは同一のクロックタイミングで動作する．カナリア・フリップフロップ側のパスにはあえて遅延が加えてあり，本体の実行パスよりも先にタイミング・エラーを生じ，ちょうど鉱山で毒ガスを検知するカナリアのような役目を果たしている．このように，RAZOR では検証用パスの方がマージンが大きいが，カナリアでは逆に実行パスのマージンの方が大きくなっている．

また佐藤らは，動的タイミング・エラー予防技術として建設的タイミング違反[12] を提案している．建設的タイミング違反では，加算器のキャリーのように，クリティカルパスがアクティブとなる頻度は少ないことに着目し，過去クリティカルパスをアクティブにした命令を記憶することによってタイミング・エラーを予防する．

3. 提案手法の概要

3.1 レジスタ・ファイル書き込み時のタイミング・エラー検出

前章であげた動的タイミング・エラー検出技術は，レジスタ・ファイル書き込み時のタイミング・エラーにはそのまま適用することはできない．しかし，レジスタ・ファイルは，i) プロセッサ内でも熱の上昇しやすいホットスポットであること[13]，ii) SRAM セル・アレイ・アクセスがタイミング・クリティカルであることなど，タイミング・エラーに脆弱なユニットの 1 つであり，書き込み時タイミング・エラー対策は必須である．

3.2 書き込み保証バッファ

そこで本論文では，レジスタ・ファイルへの書き込みが正しく行われたかをチェックし，エラーを起こしていた場合はアーキテクチャステートを回復する手法を提案する．

提案手法の概要を**図 2** に示す．前述したとおり，提案手法の基本は，単純なライト・ベリファイである．書き込み値は，まずレジスタ・ファイルと，検出用に追加される小容量の**書き込み保証バッファ**（**WAB**: Write Assuarance Buffer）の双方に書き込まれる．その後，レジスタ・ファイルの該当エントリから値を読み出し，WAB の保持値と比較して検証を行う．もし値が異なっていたときは，電圧を保ったまま動作周波数を下げてタイミング・エラーが起こりにくい状態にし，WAB に保持されている実行結果を正しい値として実行を再開すればよい．この手法では，レジスタ・ファイルのセル・アレイ・アクセス回路には変更が加えられないため，クリティカルパスがのびることはない．

ここで重要な点は，WAB は検証待ちの値だけを格納すればよく，レジスタ・ファイルに

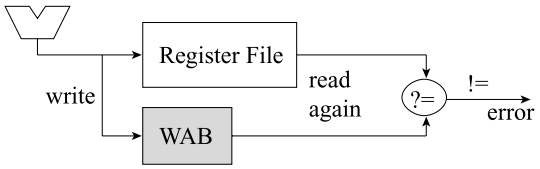


図 2　提案手法の概要
Fig. 2　The Outline of the proposal technique.

比べて容量を少なくできることである．一般に，同じ動作周波数であれば，容量が少ないバッファの方がタイミング・エラー耐性が高いといわれている[14]．このため，2.3 節で述べたような，タイミング・エラーからのフィードバック制御下では，WAB がレジスタ・ファイルに先んじてタイミング・エラーを起こす可能性は低く，WAB の値はタイミング・エラー・フリーであると考えることができる．

提案手法では，レジスタ・ファイルに検証読み出しのためのポートを新たに追加することは想定しない．ポート数を追加してしまうと，レジスタ・ファイルのタイミング・クリティカルパスが増加し，本末転倒となるためである．検証読み出しとオペランド読み出しは同じポートを使用するため，競合を起こして性能低下を引き起こす可能性があるが，5 章で述べる手法により，この影響は軽減される．

4. レジスタ・ファイル書き込み時のタイミング・エラー

本章ではレジスタ・ファイル書き込み時タイミング・エラーについて議論する．レジスタ・ファイルの構成や SRAM 書き込み時の操作の流れを追うことにより，レジスタ・ファイル書き込み時のタイミング・エラーがどのように発生するかを述べる．また，提案手法で前提としている，小容量 WAB によるタイミング・エラー耐性の確保や，電圧や周波数の変更によるタイミング・エラー耐性の回復について，回路の面からその傾向を確認する．

以後の議論では，バックエンドでレジスタ読み出しを行う，MIPS R10000，DEC Alpha 21264 などのプロセッサと同様のパイプラインステージ構成を仮定する．最近の多くのスーパスカラ・プロセッサはこの構成を採用している．ただし，リザベーションステーションを利用する方式など，他の方式においても議論に本質的な違いはない．

4.1 レジスタ・ファイル書き込み処理のクリティカルパス

レジスタ・ファイル書き込みでは，i) 書き込みアドレス（レジスタ番号）をデコードして，該当するセル・アレイの行を選択する，ii) 実行結果をセル・アレイへ転送する，iii) セル・アレイへの書き込み操作を行う，という処理が行われる．i) と ii) の処理は並行して同時に行ってよい．

一般に，処理を複数のパイプラインステージに分けれることによって，タイミング・エラー耐性を高めることができる．レジスタ・ファイル書き込みにおいても，アドレス・デコードや，書き込み値のセル・アレイへの転送などは，パイプライン化が可能である．このような部分は，必要であれば，パイプライン化によってタイミング・エラー耐性を高めることができる．また，パイプラインラッチに，RAZOR などの既存手法を適用することができる．

しかし，ビットラインやワードラインを操作する，セル・アレイ・アクセスは，その間ポートが占有されるので，パイプライン化することができない．レジスタ・ファイルのセル・アレイは，実行幅の 2 倍のリードポートと実行幅と同じ数だけのライトポートを備え，アクセスにかかる遅延は大きい．セル・アレイ・アクセスの遅延は，サブ・アレイへの分割などの技術によって短くすることができるが，削減には限界がある．このため，この部分がレジスタ・ファイル書き込み処理におけるタイミング・クリティカル・パス，すなわちタイミング・エラーの発生箇所となる．

4.2 アドレス・デコーダのタイミング・エラー耐性

なお，ここで，上ではクリティカル・パスではないとしたアドレス・デコーダのタイミング・エラー耐性について，もう少し詳細に確認する．なぜなら，アドレス・デコーダのタイミング・エラーによって，本来とは異なるワードに書き込みが行われるようなことがあれば，そのレジスタの内容が破壊され，アーキテクチャによる回復はほとんど不可能になるためである．

このような破壊的なエラーを回避するためには，前述の議論のように，パイプライン構成によってアドレスデコーダのタイミング・エラー耐性を高めることが有効である．書き込みアドレスは，読み出しアドレスとともに，命令発行ステージの最後で決まる．それに対して，書き込みデータが決まるのは，実行ステージの最後である．つまり，書き込みアドレスが分かった後，ライトバック・ステージで実際にデータが書き込まれるまでには，1 サイクル以上の余裕がある．このため，レジスタ・ファイル読み出しの場合と異なり，レジスタ・ファイル書き込みでは，アドレス・デコーダのパイプライン化は自然に実現できる．

また，アドレス・デコーダをドミノ論理で構成するなどのロジックの工夫も有効である．ドミノ論理で構成すれば，想定外の遅延が発生しても，必要なワードラインがアサートされないことはあっても，異なるワードラインがアサートされることはない．

結論として，パイプライン構成や従来技術を利用すれば，アドレス・デコーダに生じるタイミング・エラーによる，無関係なレジスタへの破壊は生じないといえる．

4.3 セル・アレイ書き込みにおけるタイミング・エラー

以降，レジスタ・ファイル書き込み処理でクリティカルとなる，セル・アレイ・アクセスに注目する．**図 3** に，セルの回路図とレジスタ・ファイル書き込み時のタイミング・チャートの例を示す．まず，**ビットライン**（*BL*）に書き込み値がセットされる．次に，該当する**ワードライン**（*WL*）がアサートされ，該当する行のセルのアクセス・トランジスタが ON

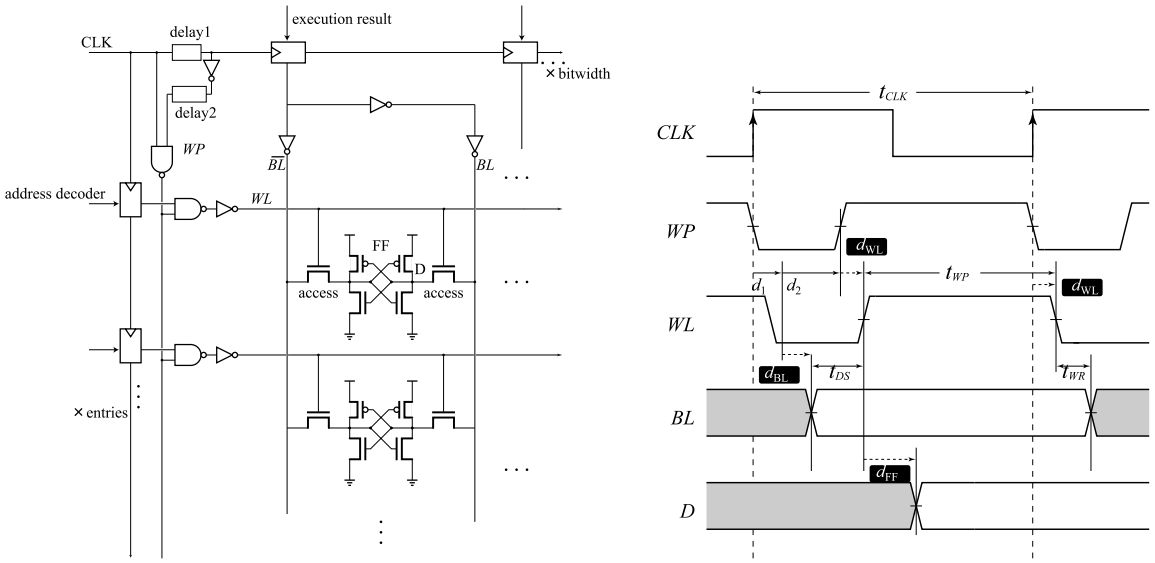


図 3　SRAM セルの回路図（左）とセル・アクセスのタイミング・チャート（右）
Fig. 3　Circuit diagram of SRAM cell (left) and timing chart of cell access (right).

となり，BL の値に従って，セル内のフリップフロップが（必要であれば）反転する．その後，WL はディアサートされ，その時点のフリップフロップの値が保持される．

書き込みが正しく行われるためには，一般には，以下の 4 条件が満たされねばならない：

データ・セットアップ・タイム　$t_{DS} > 0$.

WL がアサートされるまでに BL のセットが完了していなければならない．

ライト・パルス幅　$t_{WP} > d_{FF}$.

WL は，FF の反転に必要な時間 d_{FF} より長くアサートされ続けなければならない．

ライト・リカバリ・タイム　$t_{WR} > 0$

BL は，WL がディアサートされるまで値を保たなければならない．

サイクル・タイム　$t_{DS} + t_{WP} + t_{WR} < t_{CLK}$

1 クロック・サイクル t_{CLK} 以内に，書き込みを終えなければならない．

ここで，図 3 のタイミングチャートに点線で示した WL や BL や FF の回路遅延が想定外に増大し，上の条件のいずれかが破れると，タイミング・エラーとなる．どの条件が破られる場合も，そのサイクルに書き込もうとしていたアドレスに，間違った値が書き込まれるエラーとなる．同じアドレスの各ビットで独立に発生しうるので，マルチビットのエラーとなる可能性が高い．

詳細に見ると，まず BL の遅延が増えてデータ・セットアップ・タイムの条件が破られると，書き込みの開始が送れ，実質的なライト・パルス幅を減少させ，タイミング・エラーを生じさせやすくなる．WL の遅延が増大すると，やはりライト・パルス幅を減少させるほか，ライト・リカバリ・タイムの条件が保てなくなる．ライト・リカバリ・タイムの条件が破られると，次のサイクルに書き込むべき値をフリップフロップが取り込んでしまうタイミング・エラーが生じうる．最後に，フリップフロップの遅延が増大し，ライト・パルス幅の条件が破られると，書き込むべき値が書き込まれないタイミング・エラーが生じる．

この中で特にばらつきの影響を受けやすいのはフリップフロップの遅延であり，ライト・パルス幅の余裕が，タイミング・エラー耐性のうえで重要になると考えられる．

なお，厳密には，データ・セットアップ・タイムとライト・リカバリ・タイムの条件は，

緩和して動作速度を最適化することができる．すなわち，データ・セットアップ・タイムが負であっても，実質的な書き込み時間が d_{FF} を超えていれば，書き込みは成功する可能性がある．また，ライト・セットアップ・タイムが負であっても，WL がディアサートされるまでの時間が d_{FF} を超えていなければ，間違った値はフリップフロップに反映されない可能性がある．

4.4 容量とタイミング・エラー耐性

ここでは，以上の議論をふまえ，同じクロック周波数でレジスタ・ファイルと WAB を動作させたときのタイミング・エラー耐性の差について考える．レジスタ・ファイルと WAB の違いは主にエントリ数の違いである．WAB が数エントリから十数エントリなのに対し，レジスタ・ファイルは 100 エントリ前後となる．

エントリ数が増え，ビットラインが長くなると，以下のような影響が出てくると考えられる．

- 同じセル・アレイ内で，セルの位置によって t_{DS} や t_{WR} の変化が大きくなる．
- 含まれるセルが多くなり，フリップフロップの遅延のばらつき幅が増加する．
- ドライバから遠いセルになるほど，フリップフロップの反転開始と反転速度の双方が遅くなる．

最初の項目のためエントリ数が増えるほど，t_{DS} および t_{WR} に必要な時間が長くなる．その一方で残り 2 つの項目のため，t_{WP} に必要とされる時間も増加する．

これらの効果が合わせて現れるため，同じ周波数で書き込み動作を行う場合，エントリ数の少ない方が，タイミング・エラーに対して余裕のある設計が可能であるといえる．

4.5 DVFS とタイミング・エラー耐性

次に，電圧や周波数を動的に調整することによって，レジスタ・ファイルのタイミング・エラー耐性がどのように変化するかを考える．DVFS によるタイミング・エラー調整は，提案手法や多くの関連手法[1)–3),10)] において，タイミング・エラーを検出/回復した後，再び同じタイミング・エラーが発生しないようにするための制御の前提となっている．

BL，WL などの立ち上がり，立ち下りの制御は，クロックパルスからの遅延によって生成され，遅延素子の配置の仕方はいく通りか考えられる．しかし，電圧が上がると遅延素子の遅延は減少するため，どのような配置としても t_{DS}，t_{WP}，t_{WR}，t_{CLK} のうち，DVFS で回復するマージンと回復しないマージンの双方が生じてしまう．回復しないマージンは最悪値を見込んだ設計とせざるをえない．

図 3 の回路は，最もばらつきの影響を受けるマージンは t_{WP} であると考え，DVFS によってこの部分を調整可能としている．このために，WL の立ち下がりを次のサイクルの立ち上がりから生成している．まず，電圧を一定にしたまま周波数を下げる場合について考える．この場合，回路の各遅延は変化しないので，タイミング・エラー耐性はクロックサイクルの増加分だけ大きくなる．一方，周波数を一定にしたまま電圧を上げる場合について考えると，この場合は，WL や BL や FF の遅延が元よりも減少することとなり，タイミング・エラー耐性が高まる．

このように，メモリ回路においても，DVFS によってロジック回路と同様に，タイミング・マージンの調整を行うことが可能である．

5. 提案手法の実装

提案手法では，検証読み出しとオペランド読み出しのポート競合を避けながら書き込み値の検証を進める．検証読み出しの制御について，本論文では，**パッシブ方式**と**アクティブ方式**の 2 手法をを提案する．以下，まず 5.1 節と 5.2 節で各方式について説明したあと，5.3 節と 5.4 節で，それぞれの方式が動作周波数と IPC に与える影響について議論する．

5.1 パッシブ方式

パッシブ方式は，後続命令によるレジスタ・ファイル読み出しに「便乗」して，検証を行う方法である．この方法は，レジスタ・ファイルに書き込まれた実行結果は，近いうちに後続命令のソース・オペランドとして再び読み出される可能性が高いとの見通しに基づいている．

図 4（左）にパッシブの構成を示す．WAB は，物理レジスタ番号をキー，検証される実行結果をその値とする CAM-RAM 構成となる．書き込み検証待ちの物理レジスタ番号は CAM 部に，実行結果は RAM 部に，それぞれ格納する．

命令発行時，レジスタ読み出しリクエストは WAB にも送られ，WAB がそのレジスタ番号で連想アクセスされる．WAB から読み出された実行結果は，レジスタ・ファイルから読み出された値と比較され，検証が行われる．検証終了に合わせて，WAB の該当エントリはアドレス，実行結果とも解放される．

5.2 アクティブ方式

アクティブ方式は，レジスタ・ファイルの読み出しポートが使われない空きサイクルに検証読み出しを行う方式である．多くの命令が 2 つのソース・オペランドを持つため，通常，レジスタ・ファイルの読み出しポートは最大発行幅の 2 倍確保されている．しかし実際には，以下のようなケースも多いため，そのすべてが利用されることは少ない：

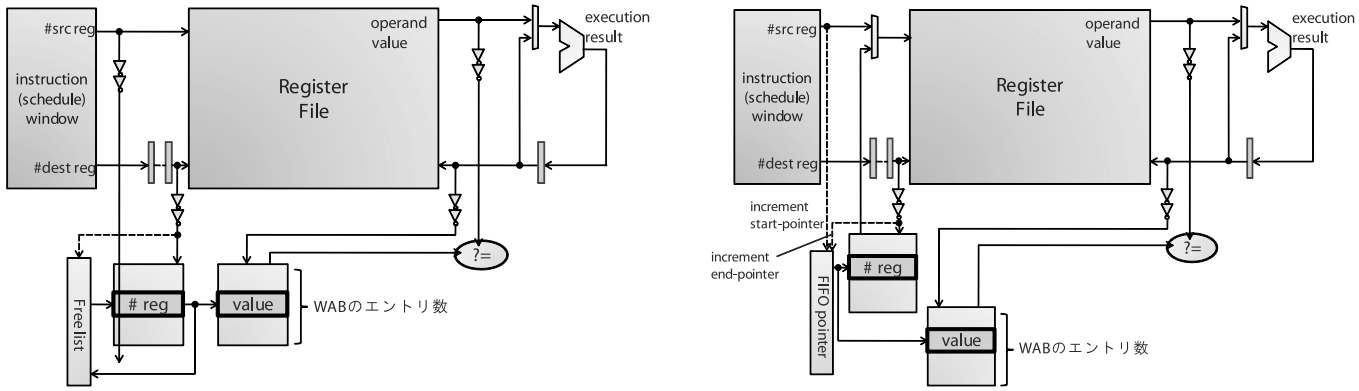


図 4　パッシブ方式（左）とアクティブ方式（右）のブロック図
Fig. 4　Block Diagrams of *passive* WAB (left) and *active* WAB (right).

(1)　命令の発行数が少ない．
(2)　即値を持つなど，レジスタを 2 つ必要としない．
(3)　ソース・オペランドがバイパスから供給される．

アクティブ方式では，WAB は，書き込み検証待ちレジスタ番号を格納する RAM と，実行結果を格納する RAM によって実現される，FIFO となる（図 4（右））．

アクティブ方式は，アイデアおよび WAB 本体はシンプルであるが，実装上は，WAB の先頭複数要素とレジスタ・ファイルの空きポートを適切に接続する回路が必要である（**図 5**）．検証待ちのレジスタ番号は，命令スケジュールと並行して WAB の先頭から読み出され，レジスタ・ファイルの空きリードポートがある場合に，そのポートへのリクエストとなる．FIFO のポインタは，このとき，許可された検証読み出しの数だけインクリメントされる．この調停結果を受けて，検証値が WAB の先頭から読み出され，レジスタ・ファイルから読み出された値と比較される．WAB の該当エントリは，検証終了に合わせて解放される．

5.3　動作周波数への影響

提案マイクロアーキテクチャによる追加操作はほとんどが本来の実行処理とは並列してバックグラウンドで行われ，実行のクリティカル・パス部分への干渉は軽微である．実装についてクリティカル・パスへの影響やその対策をまとめると以下のようになる．

ファンアウト増

パッシブ，アクティブともに，命令ウィンドウやレジスタ・ファイルから WAB への出力

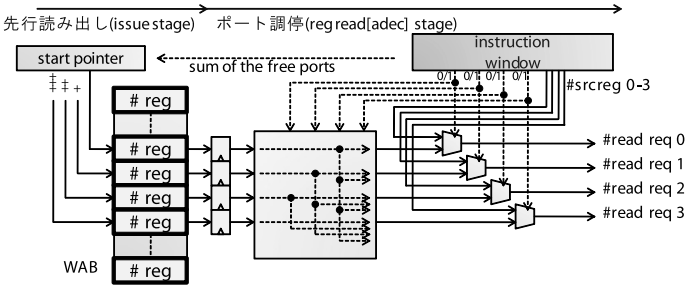


図 5　アクティブ方式におけるポート調停回路
Fig. 5　Port Arbitoration Circuit for *active* WAB.

が追加される．ただしこれらの出力は，提案手法を用いない場合でも必要なものであり，この手法のために新たに命令ウィンドウやレジスタ・ファイルから読み出されるものではなく，影響は，転送先の増加によるファンアウト増である．

図 4 では，上側が本来の実行パス（クリティカル・パス部分），下側が検証パスとなっている．ここで，ファンアウト増による遅延増加を防ぎたい部分は，実行パス部分であり，検証パス部分の遅延は隠蔽できる．そこで命令ウィンドウ，レジスタ・ファイルから小さいインバータを挟んで WAB へ信号を分岐させる実装とする（図 4）．この実装により，実行パス側に与える影響を，小さいインバータ 1 つ分のファンアウト増に抑えることができる．

パッシブ方式の連想メモリ

パッシブの場合，WAB は CAM-RAM 構成の連想メモリで実装され，レジスタ・ファイルを構成する RAM よりも複雑に見える．しかし，図 4 に示されているように，読み出し時には，CAM 部と RAM 部の間でパイプライン化することができ，必ずしも連想アクセスを 1 サイクルで行う必要はない．また，書き込み時には，フリー・リストを用いて空きエントリを特定すればよく，連想検索は必要ない．

アクティブ方式のアービトレーション

アクティブ方式の場合には，空きポートに検証読み出しアドレスを割り当てるための，アービトレーションの遅延が増加する．この遅延は，アーキテクチャ的には，スケジュール終了後の，レジスタ・ファイル読み出し遅延（ただしセル・アレイ・アクセス遅延には影響しない）の増加として現れる．この遅延は，1 サイクルで同時に行える検証呼び出しの数を減らすことにより，削減することができる．

なお，この遅延の影響で動作周波数が悪化するときには，パイプライン化を適用し，レジスタ読み出し段数が増加することになる．この段数増は，スケジューラなどの遅延に比べると影響は少ないが[15]，分岐予測ミス・ペナルティを増加させる．

5.4 IPC への影響

検証は命令実行とは並行して行われるため，検証自体が直接 IPC を悪化させることはない．提案手法が IPC に影響を与えるのは，以下の 2 つのケースである：

(1)　WAB エントリの不足により，バックエンドがストールする．

(2)　アクティブ方式の場合，書き込み値の検証が終わるまでコミットができないため，コミットが遅れて命令ウィンドウのエントリが不足し，フロントエンドがストールする．

WAB エントリの不足

パッシブ/アクティブどちらの実装方法でも，検証が滞れば，WAB エントリが不足する．このような場合には，そのサイクルの命令発行を止め，レジスタ・ファイルの読み出しポートを確保して検証を進め，エントリを解放することになる．

命令ウィンドウ・エントリの不足

もし，検証前の間違った書き込み値が使用され，コミットされると回復できなくなる．アクティブの場合，オペランド読み出しと検証読み出しが独立しているため，検証が終わる前にエラーを起こした書き込み値が使用されてしまう可能性がある．そのため，アクティブ方式の場合，各命令はエラー回復に備えて，検証が済むまでコミットを遅らせる必要がある．

なお，パッシブはソース・オペランド読み出しを利用して検証を進めるので，書き込み値は，最悪でもオペランドとして読み出されたときに検出/修正される．このため，パッシブではエラー回復のためにコミットを遅らせる必要はない．

6. 評　　価

6.1 WAB のエラー耐性の評価

提案手法では，エントリ数の異なる WAB とレジスタ・ファイルのタイミング・エラー耐性の差を利用して，タイミング・エラーを検出/回復する．このような検出法の現実性を調べるため，レジスタ・ファイルおよび WAB を図 3 に示した SRAM セル・アレイ構成で記述し，hspice による回路シミュレーションを行った．ここでの評価は，最新の回路最適化などを反映したものではなく，厳密な評価とはなっていないが，基本的な傾向は示していると考えられる．

デバイス世代は 45 nm を仮定し，トランジスタおよび配線のパラメタには PTM[16] を用いた．*WL* と *BL* の配線パラメタを**表 1** に示す．プロセッサの実行幅は 4way を想定し，レジスタ・ファイルおよび WAB は 8 リード 4 ライトのポートを備えている．また，高速化のための上位 32 ビットと下位 32 ビットへのサブアレイ分割を想定した．この回路について，同じエントリに対してまず書き込み，次のサイクルで読み出す，という動作をシミュレーションした．

正常に動作する状態から動作周波数を上げていくと，書き込み処理開始から次の読み出し処理までの時間が徐々に縮まっていき，やがてタイミング・エラーが発生する．評価した回路では，基本的に，書き込みエラーが読み出しエラーよりも先に発生した．

書き込んだ値を正常に読み出せる最大周波数を調べた結果を**図 6** に示す．異なる電圧および温度の動作条件についてそれぞれ測定した．横軸がセル・アレイのエントリ数，縦軸は最大動作周波数となっている．

表 1　中間層配線パラメータ
Table 1　Parameters for interconnect.

配線幅	100 nm
配線間隔	100 nm
配線高	200 nm
グランドとの距離	200 nm
R	1099.99 kΩ/m
C（対グランド）	17.653 pF/m
C（配線カップリング）	48.802 pF/m

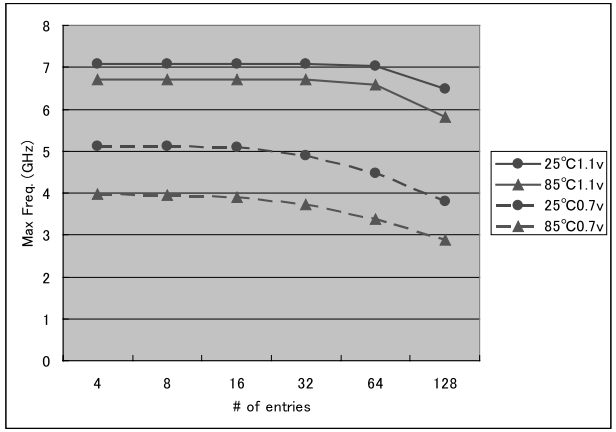


図 6　エントリ数と動作周波数
Fig. 6　Relationship between WAB size and operation frequency.

図 6 では，それぞれの動作条件について，エントリ数が少ないほど最大動作周波数が高くなっている．128 エントリと 8 エントリについて比較すると，0.7 V で 1 GHz，1.1 V でも 500 MHz の差として現れている．低電圧時，高温時のような，タイミング・エラーを起こしやすい条件のときほど，レジスタ・ファイルと WAB の最大動作周波数の差が大きいことが分かる．レジスタ・ファイルと WAB を同じ周波数で動作させた場合，この差の分だけの余裕を WAB が持つことになる．

また，図 6 では，マージンに対する影響は電圧が最も大きく，温度とエントリ数が同程度の影響となっている．このことから，WAB のタイミング・エラー・フリーを実現する手法として，WAB を高い電圧で動作させる，という手法も考えられる．この場合，WAB はエントリ数が少ないので，全体の消費電力を上げることなく，高信頼実行を実現することができる．

6.2　実行性能への影響

WAB は，小さいほどタイミング・マージンが増加し，また回路コストは低下する．一方で，WAB エントリが少ないと，WAB エントリ不足によるストールの影響が大きくなり IPC は低下すると考えられる．ここでは，サイクルレベルのシミュレーションによって，アクティブ/パッシブの 2 つの提案手法の性能評価を行い，最適な WAB 容量を検討する．

SimpleScalar Tool Set Ver. 3.0d の sim-outorder に提案手法を実装し，評価を行った．

表 2　ベースラインプロセッサ緒元
Table 2　Microarchitectural parameters for baseline processor.

way 数	4
命令セット	Alpha ISA
命令ウィンドウ	64 entries
LSQ	32entries
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	i128 entries，f128entries，1 cycle latency
L1 I/DCache	32 KB，LRU，4way，32 B line，2cycle latency
L2 Cache	512 KB，LRU，4way，64 B line，12cycle latency

ベンチマークは SPEC2000 を用い，1 G 命令を実行して計測した．ベースライン・モデルとしたスーパスカラ・プロセッサのパラメータを，**表 2** に示す．

図 7 に，書き込み保証機能を持たないベースモデルの IPC を 1 とした，パッシブ（左）とアクティブ（右）の正規化 IPC を示す．グラフの横軸は SPEC2000 の各プログラムに対応しており，一番右の組はそれらの調和平均である．3 本のバーは，左から，WAB エントリがそれぞれ 8，16，32 のときを表している．WAB の読み出しポート数は，レジスタ・ファイルと同じ 8 としている．

グラフより，アクティブでは，WAB エントリが 8 の場合でも正規化 IPC の平均は 0.933 と性能低下は抑えられている．16 エントリのときには 0.994 となり，性能低下はほとんど見られない．一方で，パッシブは性能低下が大きく，正規化 IPC の平均は，16 エントリで 0.891，8 エントリでは 0.769 となっている．

これは，多くの命令がレジスタ・ファイルからではなく，バイパスによってソース・オペランドを得ているためと考えられる．mgrid を除くベンチマークでは，全実行命令の 30% 以上がバイパスによってオペランドを得ている．命令がバイパスによってソース・オペランドを得ると，パッシブでは検証の機会が失われ，アクティブでは検証の機会が増えることになる．

また，手法のスケーラビリティを確認するために，命令ウィンドウサイズを 32 エントリから 128 エントリまで変化させて同様の評価を行った（連動して，レジスタは 64 エントリから 256 エントリまで変化させた）が，提案手法のオーバヘッドの傾向や相対 IPC はほぼ同様の結果となった．

6.3　WAB の読み出しポート数

前章で述べたように，アクティブ方式では，1 度に検証できる数を多くすると，アービタ

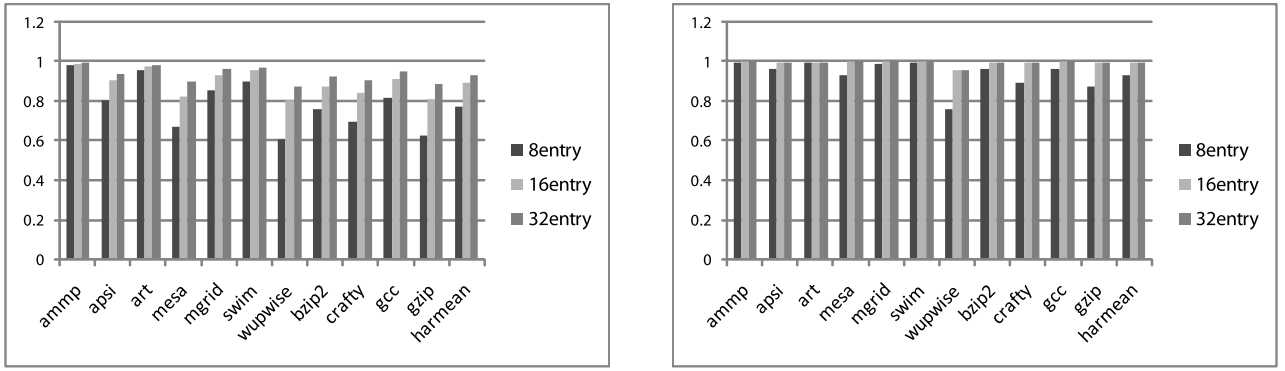


図 7　パッシブ方式（左）とアクティブ方式（右）の正規化 IPC
Fig. 7　Normalized IPC of *active* WAB (left) and *passive* WAB (right).

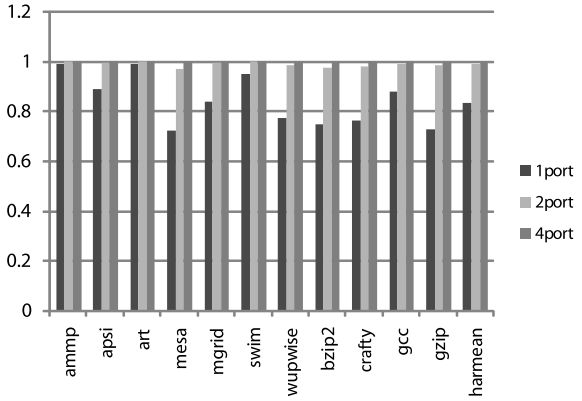


図 8　WAB 読み出しポート数と性能（アクティブ）
Fig. 8　The performance effect of WAB throughput (*Active*).

が複雑になる．そこで，アクティブ方式について，16 エントリの WAB の読み出しポート数を変化させて相対性能を調べた．結果を**図 8** に示す．レジスタ・ファイルと同じ 8 ポートであったときの性能を 1 としている．また，WAB エントリ数を 8 エントリとしてポート数を変化させた場合の相対 IPC も，同様の傾向を示した．

この結果からは，WAB の読み出しポート数は 2〜4 で十分であることが分かる．したがって，各命令の 2 番目のソース・オペランドのためのポート，あるいは，3 命令以上発行されなければ使用されないポートのみが WAB と接続されていればよく，図 5 の回路は簡略化することができる．

7. 結　　論

プロセス微細化はマイクロプロセッサの高性能化を後押ししてきたが，近年，微細化にともなうばらつきの問題が，次世代マイクロプロセッサのハードルとして認識されるようになってきた．これに対し，動的なタイミング・エラー検出を前提とした典型的ケース設計は有効な手段の 1 つと考えられる．本論文では，従来の動的タイミング・エラー検出技術の適応の難しかった，レジスタ・ファイル書き込み時タイミング・エラーを検出/回復する手法を提案した．提案手法は，タイミングマージンの大きい WAB を用意し，レジスタ・ファイルの値と比較することで，タイミング・エラーを検出する．WAB のレジスタ・ファイルに対するタイミングマージンの優位は，エントリ数の少なさによって確保される．

まず，レジスタ・ファイルの書き込みとタイミング・エラーについて，回路シミュレーションを交えた議論を行い，エントリ数の差によるタイミングマージンの差，再読み出しによる検出可能性，DVFS によるマージン回復など，提案アーキテクチャの前提となる傾向を確認した．

また，検証読み出しについて，後続命令がレジスタ・ファイル読み出しを行ったときに比

較を行うパッシブ方式と，ポートに空きができたときに検証読み出しを行うアクティブ方式とを提案した．SPEC2000 ベンチマークを用いたシミュレーション評価の結果，性能オーバヘッドの小ささでは，アクティブ方式が適していることが分かった．16 エントリ，2 ポートの書き込み保証バッファがあれば，性能低下の影響はほとんどないことを確認した．提案手法と，ロジック部分のタイミング・エラー検出技術とは直交しており，組み合わせることにより，耐タイミング・エラープロセッサを実現することができる．

本論文の評価では IPC の低かったパッシブ方式だが，レジスタ・ファイル読み出し回数を増やさない，生じたエラーが伝搬しないなど，良好な性質を持っている．この方式の改良が，今後の研究としてあげられる．

謝辞　本論文の研究の一部は，21 世紀 COE「情報技術戦略コア」，および，科学技術振興機構 CREST「ディペンダブル情報処理基盤」による．また，本研究の一部は，東京大学大規模集積システム設計教育センターを通じシノプシス株式会社の協力で行われたものである．

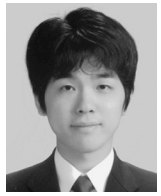
参 考 文 献

1) 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).
2) Austin, T., Blaauw, D., Mudge, T. and Flautner, K.: Making Typical Silicon Matter with Razor, *IEEE Computer* (2004).
3) 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).
4) 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).
5) Mukhopadhyay, S., Mahmoodi, H. and Roy, K.: Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.24, No.12 (2005).
6) 諏佐達也，村川正宏，高橋栄一，飯島洋祐，古谷立美，樋口哲也：タイミング余裕を確保したディジタル LSI の製造後クロック調整，情報処理学会研究報告「数理モデル化と問題解決」，No.29, pp.109–112 (2006).
7) Liang, X. and Brooks, D.: Mitigating the Impact of Process Variations on CPU Register File and Execution Units, *Int. Symp. on Microarchitecture* (2006).
8) Liang, X., Canal, R., Wei, G. and Brooks, D.: Process Variation Tolerant Register Files Based on Dynamic Memories, *Workshop on Architectural Support for Gigascale Integration* (2007).
9) Weaver, C. and Austin, T.: A Fault Tolerant Approach to Microprocessor Design, *Int. Conf. on Dependable Systems and Networks*, pp.411–420 (2001).
10) 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).
11) 五島正裕，入江英嗣，坂井修一：メモリ装置およびメモリ読み出しエラー検出方法，特願 2006-189029 (2006).
12) 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).
13) Skadron, K., Stan, M., Huang, W., Velusamy, S., Sankaranarayanan, K. and Tarjan, D.: Temperature-aware microarchitecture, *30th Int. Symp. on Computer Architecture*, pp.2–13 (2003).
14) Memik, G., Chowdhury, M., Mallik, A. and Ismail, Y.: Engineering over-clocking: Reliability-performance trade-offs for high-performance register files, *Int. Conf. on Dependable Systems and Networks*, pp.770–779 (2005).
15) Stark, J., Brown, M.D. and Patt, Y.N.: On pipelining dynamic instruction scheduling logic, *33rd Int. Symp. on Microarchitecture*, pp.57–66 (2000).
16) Predictive Technology Model (PTM). http://www.eas.asu.edu/˜ptm/

(平成 19 年 10 月 4 日受付)
(平成 20 年 3 月 4 日採録)

入江　英嗣（正会員）

1999 年東京大学工学部電子情報工学科卒業．2004 年年同大学院情報理工学系研究科電子情報学専攻博士課程修了．博士（情報理工学）．2004 年より科学技術振興機構 CREST 研究員．2008 年より東京大学情報理工学系研究科助教．コンピュータ・システムの研究に従事．特に並列処理アーキテクチャおよびディペンダブル・コンピューティング等の研究を進めている．電子情報通信学会コンピュータシステム研究会幹事補佐（2007 年〜）．電子情報通信学会，ACM 各会員．



杉本　　健（学生会員）

2007 年東京大学工学部電子情報工学科卒業．現在，同大学院情報理工学系研究科電子情報学専攻在学中．ディペンダブルプロセッサに関する研究に従事．

五島　正裕（正会員）

1968 年生．1992 年京都大学工学部情報工学科卒業．1994 年同大学院工学研究科情報工学専攻修士課程修了．同年より日本学術振興会特別研究員．1996 年京都大学大学院工学研究科情報工学専攻博士後期課程退学，同年より同大学工学部助手．1998 年同大学大学院情報学研究科助手．博士（情報学）．2005 年東京大学情報理工学系研究科助教授，2007 年同大学同研究科准教授，現在に至る．コンピュータ・システムの研究に従事．2001 年情報処理学会山下記念研究賞，2002 年同学会論文賞受賞．IEEE 会員．

坂井　修一（正会員）

1981 年東京大学理学部情報科学科卒業．1986 年同大学院工学系研究科修了，工学博士．電子技術総合研究所，MIT，RWC，筑波大学を経て，1998 年東京大学助教授．2001 年より東京大学大学院情報理工学系研究科教授．2008 年同副研究科長．計算機システムおよびその応用の研究に従事．特に並列処理アーキテクチャ，相互結合網，省電力アーキテクチャ，ディペンダブルシステム等の研究を進めている．IBM 科学賞（1991 年），市村学術賞（1995 年），IEEE Outstanding Paper Award（1995 年），Sun Distinguished Speaker Award（1997 年）等受賞．本学会では，研究賞（1989 年），論文賞（1991 年），論文誌編集委員（1998〜2002 年），学会誌編集委員（2003〜2007 年），卓越 DB 小委員会主査（2006 年〜），理事（2006 年〜）等．そのほか，日本学術会議連携会員，電子情報通信学会コンピュータシステム専門委員会副委員長，日本学術振興会学術専門委員等．電子情報通信学会，人工知能学会，IEEE，ACM 各会員．