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

2006－ＡＲＣ－168（２）
2006／6／8

演算結果再利用による高信頼かつ低消費電力なプロセッサに関する検討

橋口 陽祐† 井上 弘士†† 村上 和彰††

† 九州大学 大学院システム情報科学府 情報理学専攻
†† 九州大学 大学院システム情報科学研究院 〒816-8580 福岡県春日市春日公園 6-1
科学技術振興機構さきがけ 〒332-0012 埼玉県川口市本町 4 丁目 1 番 8 号
E-mail: †hashiguchi@c.csce.kyushu-u.ac.jp, ††{inoue,murakami}@i.kyushu-u.ac.jp

あらまし プロセッサにおけるソフトエラー耐性の低下が問題になっている. ソフトエラーとは, 雑音が原因で回路が一時的に誤動作する現象である. 信頼性を向上させるため, メモリではパリティや ECC 等の誤り検出/訂正コードが用いられる. しかしながら, 組合せ回路にこのような誤り検出/訂正コードを加えることは難しく, 多くの場合はプログラム実行を多重化 (複数回実行) することでエラー検出を可能にしている. 本研究では, 演算結果の再利用に基づく高信頼かつ低消費エネルギーなプロセッサアーキテクチャを検討する. 本手法ではプログラム中の同一命令の演算結果を演算結果再利用テーブルに保持しておき, それを再利用する. 演算結果再利用テーブルは ECC で保護するため, 各命令の実行を多重化することなく高い信頼性を実現できる. これにより, 信頼性の向上に伴う消費エネルギー・オーバーヘッドを削減する. 定量的評価を行った結果, 従来の多重化に基づく方式では, 多重度 2 のとき 100%であった消費エネルギー・オーバーヘッドを 6.3%に削減することができた.
キーワード ソフトエラー 信頼性 消費エネルギー

A Low-Power, Reliable Datapath by Reusing Execution Results

Yosuke HASHIGUCHI†, Koji INOUE††, and Kazuaki MURAKAMI††

† Dept. of Infomatics, Kyushu University
†† Dept. of Infomatics, Kyushu University Kasuga park 6-1, Kasugashi, 816-8580 Japan
PRESTO, Japan Science and Technology Agency, 4-1-8 Honcho Kawaguchi, Saitama 332-0012 Japan
E-mail: †hashiguchi@c.csce.kyushu-u.ac.jp, ††{inoue,murakami}@i.kyushu-u.ac.jp

Abstract The decrease in the soft error tolerance in processors becomes a problem. The soft error is a phenomenon that the circuit does not malfunctions temporarily by the noise. To improve reliability, there is parity and ECC in the memory. However, it is difficult to add the error detection/correction code in combinational circuits. It enables the error detection by multiplexing the execution program. It has the problem that increases the energy consumption. In this research, We investigate the reliable datapath by reusing execution results. It does not execute the same instruction in detail. It maintains the result in a table, and obtains the result without ALU. The table with ECC can have reliability. The energy consumption depends on the table composition. Result of examining table composition, it can adjust the amount of the increased energy consumption to 6.3%.
Key words soft error, reliability, energy consumption

1. はじめに

トランジスタの微細化, クロックの高速化, および低消費電力化に伴い, プロセッサにおけるソフトエラーの問題が深刻になっている. ソフトエラーとは, 一時的に回路が誤動作を起こす現象である. その主な原因は, 宇宙線が大気内に入射したときに生じる中性子である. 現状では, 中性子による影響をパッケージ技術によって完全に回避することは難しい. したがって, 如何に故障が発生しないようにするかだけではなく, 故障が発生しても正常な結果を出力することが信頼性の向上には必要となる. 携帯電話や PDA などの携帯情報端末において電子商取引が行われるようになってきたことを考えると, 信頼性の向上は重要である. 一方でバッテリの長時間駆動を目的とした低消費エネルギー化が依然として求められており, 信頼性とエネルギー効率

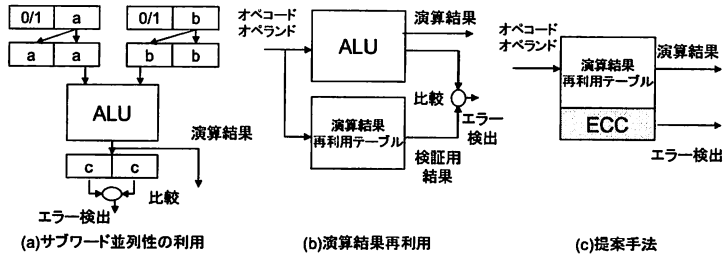

図 1　各手法の概要図

は一対の課題となっている.

従来のプロセッサ高信頼化手法として, 時間/空間冗長性を利用する手法がある. 時間的冗長性を実現する場合, プログラムを複数回実行し, それらの結果を比較 (多数決) する. 一方, 空間的冗長性を実現する場合には複数の演算器で同一命令を実行し, 結果の比較を行う. しかしながら, これらの手法は複数の演算結果を得るために冗長な演算が必要となるため, 性能ならびに消費エネルギーのオーバーヘッドが増大する. この問題を解決するために冗長な演算を行わなくても結果の比較 (多数決) を行う手法が提案されている [5][6]. しかしながら, これまでは性能オーバーヘッドの隠蔽を目的とする場合が多く, 消費エネルギーに関してはほとんど議論されていない.

本研究では故障の種類をソフトエラーに限定し, プロセッサの信頼性の向上を目的としたアーキテクチャレベルでの演算の新たな手法を提案する. 本手法では演算結果を ECC(Error Correcting Code) で保護された演算結果再利用テーブルに保持しておく. そして, 同一命令が実行されたとき, 演算結果再利用テーブルに保持されている演算結果を参照することで高い信頼性を維持する. これにより, 演算自体を行わないことで消費エネルギーを削減する. 本稿では, 信頼性と消費エネルギーに関して, 定性的ならびに定量的評価を行う.

本稿は以下の構成である. 第 2 節では関連研究を紹介する. 次に第 3 節では演算結果再利用に基づく高信頼化手法を提案し, その詳細を説明する. 第 4 節で定性的評価, 第 5 節で定量的評価を行う. 最後に第 6 節で本稿をまとめる.

2. 関連研究

プロセッサ高信頼化手法として時間/空間冗長性を利用する手法がある. しかしながら, この手法は通常の演算結果に加え検証用の演算結果が必要であり, 性能および消費エネルギーのオーバーヘッドが問題となる. 本節では, この検証用結果の生成における性能オーバーヘッドを削減する手法について紹介する.

2.1 サブワード並列性の利用

演算を実行する場合,2 つのソース・オペランドが必要となる. その 2 つのソース・オペランドがハーフワード以下に収まる場合, 図 1 の (a) に示すように下位ビットの値を上位ビットにコピーして演算を実行する. これにより, 空間冗長性を実現することで 1 回の実行で 2 つの演算結果を得ることができる. そのため, エラーを検出するための冗長な演算は不要となる [5].

2.2 演算結果再利用

演算において, 図 1 の (b) に示すように, 演算器とは別に演算結果を保持する演算結果再利用テーブルを追加する. 命令実行時に, 演算結果再利用テーブルに同一命令が保持されていない場合は, 演算を複数回実行するなどの従来手法により演算結果を生成し, 演算結果再利用テーブルに保持する. 同一命令が保持されている場合は, 演算結果再利用テーブルを参照することで検証用結果の生成が不要となる [6].

3. 演算結果再利用に基づくプロセッサの高信頼化

第 2.2 節にて紹介した演算結果を再利用する手法においても演算結果再利用テーブルを用いる. しかしながら, その目的は検証用演算の削除であり, 演算結果を比較することで信頼性を向上させている. 一方, 我々は演算結果再利用テーブルを ECC で保護し, 高い信頼性を維持することで演算自体を省略する手法を提案する. この手法を MAC(Memorizational Arithemic Cut) と呼ぶ. 本手法により, 多重実行を行わないことで演算における消費エネルギーを削減する.

MAC では, 演算において演算器とは別に演算結果を保持する演算結果再利用テーブルを必要とする. 図 2 に示すように, 本テーブルは各エントリにオペコード,2 つのソース・オペランド, 演算結果が保持される.

演算命令実行時, 演算結果再利用テーブルを参照する. 実行命令のオペコードと 2 つのソース・オペランドを用いて連想検索を行う. 一致するエントリが存在する場合は同一命令と認識され, 該当エントリに格納された演算結果を読み出す. この場合を「ヒット」とする. 一致しない場合は, 従来手法と同様に演算実行の多重化を行い, その結果をオペコード, ソース・オペランドと共に演算結果再利用テーブルに保持する. この場合を「ミス」とする.

4. 定性的評価

4.1 信頼性

MAC のエラー検出は, 複数の演算結果の比較でなく ECC によるものである. ここでは, 従来の単純な多重実行 (ここでは, 演算器を 2 個用いた多重度 2 の空間冗長を想定), 従来の演算結果

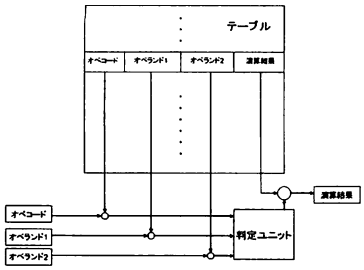

図 2　構成概略図

再利用 (図 1 の (b)), および MAC の 3 つの手法の信頼性について評価する. なお,ECC は 1 ビットエラーの訂正と,2 ビットエラーの検出を前提とする. 各々の手法において, エラーを検出できない状況は以下のようになる.

・空間多重実行の場合:　エラーが検出されないのは, 現在の演算結果と検証結果において, 同じビット位置で故障が発生したときである.

・演算結果再利用の場合:　演算結果再利用テーブルに保持されている値と現在計算した値において, 同じビット位置で故障が発生したときである. しかし, 演算結果再利用テーブルを保護するか否かでその信頼性は大きく異なる. 演算結果再利用テーブルが ECC によるエラー保護を受けているとき, エラーが検出されないのは共に 3 ビット以上の故障が発生し, そのビット位置が同じときである. しかしながら, この場合は ECC と演算結果比較の 2 回のエラー検出を行っている. そこでエラー検出が 1 回の場合を考える. このとき演算結果再利用では比較を行うので演算結果再利用テーブルの ECC による保護はなくなる.

・MAC の場合:　演算結果再利用テーブルが ECC によるエラー保護を受けているので,1 エントリで 3 ビットの故障が発生したとき, その検出が不可能となる.

各手法の信頼性を評価する前提条件として, 演算結果再利用テーブルに保持されている演算結果は常に正しい値とする. また, 演算器における 2 ビット以上の故障についてエラー検出されない割合は非常に小さいため, 演算器については 1 ビットの故障 (単一故障) のみを考える. さらに, 故障は演算器ならびに演算結果再利用テーブルでのみ発生すると仮定する. このとき,$F_{ALU}, F_{mem}, F_{空間}, F_{再利用}, F_{MAC}$ を以下のように定義する.

・F_{ALU}:　演算器での時間当りの 1 ビットエラー発生回数

・F_{mem}:　演算結果再利用テーブル 1 エントリでの時間当りの 1 ビットエラー発生回数

・$F_{空間}$:　空間多重実行においてエラーが検出されない回数

・$F_{再利用}$:　従来の演算結果再利用でエラーが検出されない回数

・F_{MAC}:　MAC でエラーが検出されない回数

オペコードを 8 ビット, 各オペランド, 演算結果を 32 ビットとすると 1 エントリは 104 ビットとなり, 演算結果再利用テーブルにおける演算結果エントリでのエラー発生回数は $\frac{32}{104}F_{mem}$ となる. これより, 各々の方式においてエラーが検出されない回数を求めると式 (1)(2)(3) のようになる.

$$F_{空間} = \frac{1}{32}F_{ALU}^2 \tag{1}$$

$$F_{再利用} = \frac{1}{104}F_{ALU} * F_{mem} \tag{2}$$

$$F_{MAC} = F_{mem}^3 \tag{3}$$

ここでメモリと演算器のエラー発生回数を面積と中性子数から見積もる.

まず, メモリにおけるエラー発生回数を求める. 中性子の数は 1 時間当たり $1cm^2$ に 20 個程度とされている [4]. 全ての中性子で 1 個当たり 1 ビットの値が反転すると仮定する.SRAM1 ビットあたりの面積は $90nm$ プロセス技術で $1\mu m^2$ である [8]. オペコード 8 ビット, オペランド、演算結果を 32 ビットとすると 1 エントリは 104 ビットになる. このとき中性子がエントリにあたる回数は約 2.0×10^{-5} 個/h になる. これより

$$F_{mem} = 2.0 \times 10^{-5}回/h \tag{4}$$

と推測できる.

次に, 演算器のエラー発生回数を求める.Intel の Pentium4 の場合, 演算器の割合はおよそ 5 %程である [7]. このときの演算器の面積は約 $5.7mm^2$, マスク率 (中性子が当たっても結果に影響されない割合) を 99.999 % (10 万分の 1) とする. なお,ARM のプロセッサにおけるマスク率が 95〜98%という報告がある [1]. このとき, マスクされないエラー発生回数は約 1.1×10^{-5} 回/h になる. これより

$$F_{ALU} = 1.1 \times 10^{-5}回/h \tag{5}$$

と推測できる. よって式 (4)(5) を式 (1)(2)(3) に代入すると式 (6)(7)(8) となる.

$$F_{空間} = 3.78 \times 10^{-12}回/h(3.78 \times 10^{-2}FIT) \tag{6}$$

$$F_{再利用} = 2.12 \times 10^{-12}回/h(2.12 \times 10^{-2}FIT) \tag{7}$$

$$F_{MAC} = 8.0 \times 10^{-15}回/h(8.0 \times 10^{-5}FIT) \tag{8}$$

これより,MAC の信頼性は高いことがわかる.

4.2　消費エネルギー

消費エネルギーにおいて, 空間多重実行, 従来の演算結果再利用, および MAC の 3 つの手法にて比較を行い,MAC の有効性を明らかにする. なお, 複数の演算結果の比較による消費エネルギーは考慮せず, 演算器と演算結果再利用テーブルの消費エネルギーで議論する. また, 演算結果再利用テーブルにおける書込みと読み出しの消費エネルギーは等しいもの仮定する. ここで,$E_{mem}, E_{ALU}, x, n, \alpha, E_{空間}, E_{再利用}, E_{MAC}$ を以下のように定義する.

・E_{mem}:　メモリへの読み出し, 書込みによる 1 回当りの消費エネルギー

・E_{ALU}:　演算器による 1 回の演算消費エネルギー

・x:　演算結果再利用テーブルのヒット率 ($\frac{総ヒット回数}{総実行演算命令数}$)

・n: 実行される演算命令数
・α: ECC での保護による消費エネルギー増加率
・$E_{空間}$: 空間多重実行時の消費エネルギー
・$E_{再利用}$: 演算結果再利用の消費エネルギー
・E_{MAC}: MAC を用いたときの消費エネルギー

このときの各手法の消費エネルギーは式 (9)(10)(11) となる.

$$E_{空間} = 2E_{ALU} * n \quad (9)$$

$$E_{再利用} = (2-x)E_{ALU} * n + E_{mem} * n \quad (10)$$

$$E_{MAC} = 2(1-x)E_{ALU} * n + (1+\alpha)E_{mem} * n \quad (11)$$

ここで, 演算器と演算結果再利用テーブルの消費エネルギー比を式 (12) で定義し, 高信頼化手法を用いない場合 (演算器による 1 回演算) と比較すると, 消費エネルギー増加量 $E'_{空間}, E'_{再利用}, E'_{MAC}$ は式 (13)(14)(15) のようになる.

$$a = \frac{E_{mem}}{E_{ALU}} \quad (12)$$

$$E'_{時/空} = E_{ALU} * n \quad (13)$$

$$E'_{再利用} = (1-x+a)E_{ALU} * n \quad (14)$$

$$E'_{MAC} = (1-2x+(1+\alpha)a)E_{ALU} * n \quad (15)$$

式 (13)(14)(15) より, 各手法における消費エネルギー増加量について以下のことがわかる. 時間/空間冗長性を利用した場合, 検証演算分の消費エネルギー増加が必ず発生する. 演算結果再利用,MAC に関しては, ヒット率, 演算器と演算結果再利用テーブルの消費エネルギー比によって消費エネルギー増加量が変化する. ヒット率, 消費エネルギー比は共にテーブルサイズに依存している. 少ないエントリ数でもヒット率が高い場合, メモリの消費エネルギーが低くなり, 演算器による演算が減少することで消費エネルギー増加量を減少させる効果が期待できる. よって, 少ないエントリ数でヒット率の高い演算結果再利用テーブルの構成法を考える必要がある.

5. 定量的評価

第 4.2 節において, 空間多重実行, 従来の演算結果再利用方式, および MAC の消費エネルギー増加量を定性的に評価した. 本節では以下の値を実験で求め, 消費エネルギーを定量的に評価する.

- 演算結果再利用テーブルのヒット率
- 演算器と演算結果再利用テーブルの消費エネルギー比
- ECC での保護による消費エネルギー増加率

5.1 ヒット率計測における準備

演算結果再利用テーブルの構成法の違いでヒット率がどのように変化するか調べる. ヒット率を決定するパラメータとしては以下のものがある.

- テーブル参照アドレス生成法
- テーブルの実装方法 (共有・専用)
- エントリ数
- エントリ配置法
- エントリ置換法
- テーブルを参照する命令

各要素を変化, あるいは固定してそのときのヒット率を求める. 本実験で固定する要素は配置法, 置換法, 参照する命令とし, 変化させる要素としてテーブル参照アドレス生成法, エントリ数, 実装方法とする.

5.1.1 テーブル参照アドレス生成法

演算結果再利用テーブルを有効に活用するためには, テーブルの使用領域が偏ることがないようにする必要がある. そこで, アドレスとして用いる値としてソース・オペランドと PC を考える. ソース・オペランドを利用する場合, いずれか 1 つの値をアドレスとして用いると, 偏りが生じる恐れがあるので 2 つの値に xor,add を施したものをアドレスとする. 一方,PC を利用する場合は命令の内容とは無関係なため, 偏りのない均一なテーブルの使用が期待できる.

5.1.2 テーブルの実装方法 (共有・専用)

演算器が使われる命令は, 大別すると整数演算命令, ロードストア命令のテーブル参照アドレス生成法, 浮動小数演算命令の 3 種類である. この 3 種類の命令を 1 つの演算結果再利用テーブルに割り当てる (共有) か, それぞれに専用の演算結果再利用テーブルを用意する (専用) かでヒット率が異なるか調べる. なお, 本実験では, 共有と専用のいずれの場合においても総メモリ容量は同一とする.

5.1.3 エントリ数

エントリ数がヒット率に大きく影響を与えるのは明らかである. 実験を行う際のエントリ数は全体で $4\times3, 16\times3, 64\times3, 256\times3, 1024\times3, 2^{15}\times3, 2^{20}\times3$ の 7 通りとし, それぞれのヒット率を求める.

5.1.4 エントリ配置法

配置法にはダイレクトマップ, セットアソシアティブ, フルアソシアティブがある. フルアソシアティブに関しては高いヒット率が期待できるが, エントリ数の増加に伴い検索時間が大きくなりテーブルアクセス時間に影響を与える. 本実験での配置法はセットアソシアティブ方式とし, 連想度を 1,2,4 とする.

5.1.5 エントリ置換法

置換法は LRU(Least Recently Used) を用いる.LRU は仮想メモリやキャッシュの管理などでもっとも一般的に使われているアルゴリズムで, セット内で使用されずにいた時間のもっとも長いエントリを置換する.

5.1.6 テーブルを参照する命令

ロード命令, ストア命令, 整数演算命令, および浮動小数点演算命令を対象とする. 本実験では,SimpleScalar-3.0 の machine.def にある load/store operations,Integer ALU operations, floating point ALU operations で定義された命令を対象とする.

5.2 実験概要

演算結果再利用テーブルの構成とヒット率の関係を求める実験について, まず実装方法について共有時, 専用時でのヒット率の違いを求める. 次に, テーブル参照アドレス生成法によるヒット率の違いを求める. このとき, 実装方法はヒット率が高かった

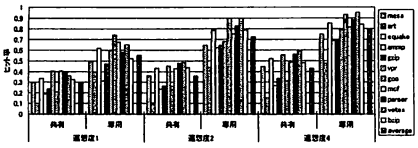

図 3　テーブルの実装方法とヒット率の関係

方に固定する. 最後に, エントリ数によるヒット率の違いを求める. このときも同様に, 実装方法とテーブル参照アドレス生成法はヒット率が高いものを用いる.

演算器と演算結果再利用テーブルの消費エネルギー比を見積もる実験については, 消費電力比を用いる. これは消費エネルギーが消費電力の時間積分で求められるからである. ここで, 演算器と演算結果再利用テーブルの演算における動作時間が等しいと仮定する. これより演算器と演算結果再利用テーブルの動的消費電力を計測することで, 消費エネルギー比を擬似的に求める. 演算結果再利用テーブルの消費電力は dL1 キャッシュの消費電力を用いる. ブロックサイズを 13 バイト (オペコード 1 バイト, オペランド 4 バイト × 2, 演算結果 4 バイト) として実験を行う.

また,ECC での保護による消費エネルギー増加率を見積もる実験について, 消費エネルギー比の見積もりと同様に消費電力を計測する. 演算結果再利用テーブルの消費電力に ECC 使用における消費電力を付加することで求める.

ヒット率の計測においては SimpleScalar-3.0[9] の sim-safe を用いる. ベンチマークは SPEC CPU 2000 ベンチマークセット [9] より,11 個のベンチマークプログラムを用いる. プログラムは先頭から 20 億命令を実行した. 消費エネルギー比,ECC での保護による消費エネルギー増加率の見積もりにおいては wattch[2] を用いる.

5.3 実験結果

5.3.1 演算結果再利用テーブルのヒット率

まず, 実装方法とヒット率の関係を図 3 に示す. テーブル参照アドレス生成法は 2 つのオペランドの xor, 演算結果再利用テーブルサイズは共有時で 3×2^{10} エントリ, 専用時で各 2^{10} エントリとした. 図 3 より, 専用時のヒット率が高いことがわかる. 理由としては, 共有時で命令の種類毎で使用する領域が重なりすぐに置換されていた命令が演算結果再利用テーブルが専用となることで解消されたためであると考える.

次に, テーブル参照アドレス生成法とヒット率の関係を調査した. 実装方法は各命令専用の演算結果再利用テーブルとし, 各エントリ数は 2^{10} とする. まず, 連想度 1 のときのテーブル参照アドレス生成法とヒット率の関係を図 4 に示す.xor,add,PC は順に 2 つのオペランドの排他的論理和,2 つのオペランドの和,PC がアドレスとして用いられていることを意味する. 図 4 より, テーブル参照アドレス生成法にオペランドを用いた方が効果的であることがわかる. 次に, テーブル参照アドレス生成法を xor,add に限定し, 連想度を変化させた場合のヒット率を図 5 に示す.xor,add のヒット率にほとんど差は見られなかった.

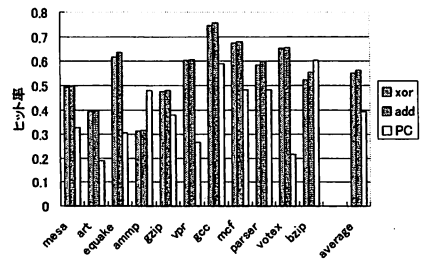

図 4　テーブル参照アドレス生成法とヒット率の関係その 1

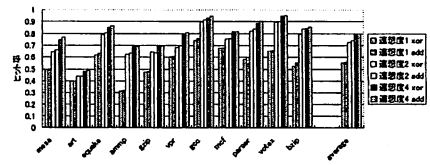

図 5　テーブル参照アドレス生成法とヒット率の関係その 2

図 6　連想度 1 のときのエントリ数とヒット率の関係

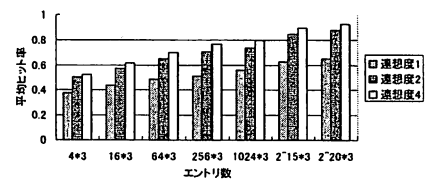

図 7　連想度とヒット率の関係

最後に, エントリ数とヒット率の関係を図 6 に示す. テーブル参照アドレス生成法は 2 つのオペランドの add, 実装方法は各命令専用の演算結果再利用テーブルとしている. 各命令のテーブルのエントリ数を 2^n とする. そのため, 全体のエントリ数は,3 の倍数となる. やはりエントリ数の増加に伴いヒット率も増加している. 連想度 2,4 のときでも同様な結果が得られた. 連想度とヒット率の関係を図 7 に示す. 同じエントリ数でも連想度の大きいほうがヒット率は高い. しかし連想度 2,4 でのヒット率の差は連想度 1,2 でのヒット率の差と比較すると大差はなく, エントリ数によらずその差は 0.05 程であった. このことから, エントリ数を増やすことよりも連想度を増やしたほうが効果があることがわかる.

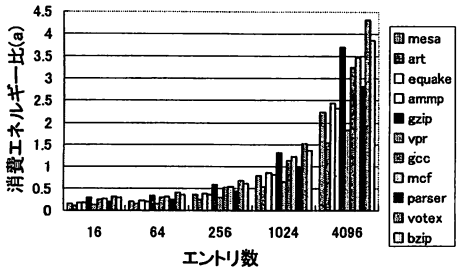

図 8　連想度 4 のときの消費エネルギー比

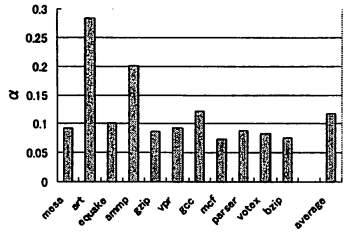

図 9　ECC での保護による消費エネルギー増加率

5.3.2　消費エネルギー比

連想度 4 における演算器と演算結果再利用テーブルの消費エネルギー比を図 8 に示す. なお, 連想度 1,2 にて同様の実験を行った結果, 連想度による消費エネルギー比に違いが見られなかったため, 本稿では割愛する. ベンチマークによっては, エントリ数 1024 で消費エネルギー比が 1 以上となっており, 演算器以上の消費エネルギー値となっていることがわかる.

5.3.3　消費エネルギー・オーバーヘッド

ECC の使用により増加した消費電力は,[3] における結果を参考に 26mW とおく. エントリサイズを 16 バイトとしたときの,ECC での保護による消費エネルギー増加率を図 9 の示す. ベンチマークごとの値は連想度 1,2,4 の値の平均である.ECC での保護による消費エネルギー増加率はベンチマークによって大きなばらつきがあるが, 多くは 0.1 未満であった. この値はエントリサイズが増大すると減少している. そのため ECC での保護による消費エネルギー増加率は 0.1 に固定して消費エネルギーの比較を行う.

図 10 は各手法を用いたときの連想度 4 のときの消費エネルギー増加率を示す. エントリ数については, 演算器と演算結果再利用テーブルの消費エネルギー比を求める際に用いたエントリ数に最も近いものを用いた. また, ヒット率は 11 ベンチマークの平均をとったものである. これを見ると消費エネルギー増加量を最小にするエントリ数があることがわかる. 本実験ではエントリ数 48, 連想度 4 で増加率 6.3%が最小であった.

6. 終わりに

本稿では演算結果を再利用することでプロセッサの信頼性を向上させる手法について信頼性, 消費エネルギーの観点から考察を行った. 信頼性については時間/空間冗長性を利用したときと比較しても, 高い信頼性の維持が期待できる. 消費エネルギーについては演算器と演算結果再利用テーブルの消費エネルギー比と演算結果再利用テーブルのヒット率が大きく関係している. 消費エネルギー比については dL1 キャッシュの値を用いて擬似的に求めた. その結果, エントリ数 48, 連想度 4 で消費エネルギー増加率が最小で 6.3%になった.

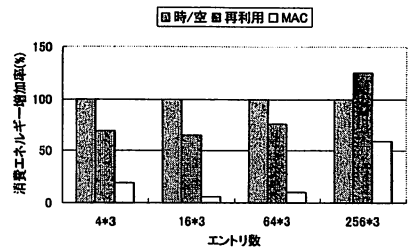

図 10　連想度 4 のときの消費エネルギー増加率

謝　辞

本論文をまとめるにあたり, 多大なるご協力を頂いた九州大学大学院システム情報科学府三輪英樹氏に深く感謝します. また, 本研究を進めるにあたり, 共にご議論頂いた九州大学システム LSI 研究センターならびに安浦・村上・松永・井上研究室の皆様に感謝します. なお, 本研究は一部, 科学研究費補助金 (学術創成研究費:課題番号 14GS0218, 若手研究 A:課題番号 17680005) ならびに,JST さきがけ研究による.

文　献

[1] Jason Blome, Scott Mahlke, Daryl Bradley, and Kristzian Flautner,"A Microarchitectural Analysis of Soft Error Propagation in a Production-Level Embedded Microprocessor," 1st Workshop on Architectural Reliability, Nov. 2005.

[2] David Brooks, Vivek Tiwari, Margaret Martonosi,"Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," Proc. of the International Symposium on Computer Architecture, pp.83-94, June 2000.

[3] Lin Li, Vijay Degalahal, N. Vijaykrishnan, Mahmut Kandermir, Mary Jane Irwin, "Soft Error and Energy Consumption Interactions: A Data Cache Perspective", Proc. of the International Symposium on Low-Power Electronics and Designs, pp.132-137, Aug. 2004.

[4] Reed Electronicd Group, EDN japan, p51-p58, 2005 年 2 月.

[5] 佐藤寿倫, "性能低下ゼロを目指した耐過渡故障マイクロプロセッサ", 信学技報 CPSY2004-60, pp.73-78, 2004 年 12 月.

[6] 佐藤寿倫, 有田五次郎, "命令冗長性を利用したフォールトトレラントプロセッサ", 信学技報 DC2002-36, pp.13-18, 2002 年 12 月.

[7] 村上和彰, 井上弘士, 吉松則文, "アーキテクチャ徹底的なソフト化で性能/電力を向上", NIKKEI MICRODEVICES, p44-p47, 2005 年 3 月.

[8] http://www.intel.co.jp/

[9] "SimpleScalar Simulation Tools for Microprocessor and System Evaluation," URL:http://www.simplescalar.com/.