様式 C-19

科学研究費補助金研究成果報告書

平成 23 年 6 月 10 日現在

機関番号:12601
研究種目:基盤研究(B)
研究期間:2008～2011
課題番号:20300015
研究課題名(和文) プログラマビリティと最大性能を両立するベクトル・アーキテクチャの研究
研究課題名(英文) Study on Vector Architecture for both Programmability and Peak Performance
研究代表者
五島 正裕(GOSHIMA MASAHIRO)
東京大学・大学院情報理工学系研究科・准教授
研究者番号:90283639

研究成果の概要(和文): SIMD は, ベクトル処理の方式として中心的な地位を占めているが, プログラマビリティに問題があり, 複雑化するアプリケーションに対処することができない. 本研究は, プログラマビリティと最大性能を両立することを目標とする. Switch-on-Future-Event マルチスレッディングは, プログラマビリティを犠牲にすることなく, 最大で 33.5%の性能向上を達成することができる. マルチスレッディングのために生じるレジスタ・ファイルの大型化は, 非レイテンシ指向レジスタ・キャッシュ・システムによって緩和することができる. シミュレーションにより, 回路面積は 24.9%にまで削減できることが示された.

研究成果の概要(英文): Although SIMD plays an important role in vector processing, suffers from low programmability and cannot cope with new applications which are becoming more complicated. This research places equal emphasis on programmability and for peak performance. Switch-on-Future-Event multithreading achieves maximum of 33.5% performance improvement without sacrificing programmability. Increase in are of the register file cause by the multithreading is relaxed by the Non-latency-Oriented Register Cache System. Simulation results show the area is reduced to 24.9%.

交付決定額

(金額単位:円)

	直接経費	間接経費	合 計
2008 年度	2,000,000	600,000	2,600,000
2009 年度	5,500,000	1,650,000	7,150,000
2010 年度	6,500,000	1,950,000	8,450,000
総 計	14,000,000	4,200,000	18,200,000

研究分野: 総合領域

科研費の分科・細目: 情報学・計算機システム・ネットワーク

キーワード: 計算機アーキテクチャ, ベクトル処理, SIMD, スーパスカラ・プロセッサ, マルチスレッド・プロセッサ, レジスタ・ファイル, レジスタ・キャッシュ

1　研究開始当初の背景

近年では，SIMD　(Single-Instruction/Multiple-Data stream) が，ベクトル処理の方式として中心的な地位を占めている．現在，Top 500 の上位を占めるスーパコンピュータのほとんどは，ベクトル方式ではなく，汎用プロセッサのクラスタとなっている．その最大性能は，各コアが備える SIMD ユニットによるものである．また近年では，科学技術計算用のスーパコンピュータに限らず，身の回りにあるさまざまな情報機器にもベクトル処理用のプロセッサが搭載されている．そのようなプロセッサとしては，PC などの GPU，SONY Cell BE に代表されるゲーム・コンソール用プロセッサ，デジカメ，ハンディカム，レコーダや携帯音楽プレーヤのビデオ/オーディオ・コーデック，ディスプレイやプリンタの画像処理エンジンなどがある．これらのベクトル・プロセッサの市場は，汎用 CPU に比べても決して小さいものではない．ベクトル・プロセッサはいまや，組み込み機器や携帯機器まで，ありとあらゆる情報機器に組み込まれており，その処理能力/効率は，これらの製品にとって，重要な差別化のポイントの一つとなっている．このようなベクトル処理用 LSI の市場規模は，年間 20 億米ドルに達するとの観測もある．

SIMD の利点と問題点

これらのベクトル・プロセッサのアーキテクチャとしては，4-way 程度の SIMD 命令セットを備えたマルチコア・プロセッサが主流となっている．プロセッサのチップ面積は，演算器と，それら演算器の制御部に分けられる．プロセッサの最大性能は，与えられたチップ面積の何割を演算器に割くかによって決まる．

SIMD は，たとえば 4-way の演算器間で制御部を共有できるため，チップ面積のより多くの部分を演算器に充てることができ，それだけチップ面積あたりの最大性能が向上するのである．しかし，最大性能が向上する一方で，SIMD はプログラマビリティに問題がある．もともと SIMD 命令セットは，1 つの（64-bit 程度の）レジスタに同次座標 (x, y, z, w) や色情報 (r, g, b, α) などの 4 つ組を格納し，要素のそれぞれに同じ処理を施すために開発されたものである．そのため一般のベクトルを処理する場合にも，配列の連続する 4 要素に対して同じ処理を施す場合には，有効に機能する．このような処理は，行列演算など，規則的 (regular) なループとして表現できる．

しかし，不規則 (irregular) なループに対しては，SIMD 化は困難である．ここで，SIMD 化とは，効率よく SIMD 命令に変換できることをいう．SIMD 化は，厳密に並列実行しなければならないため，原理的には逐次処理であるベクトル化より難しい．SIMD 化の観点からは，不規則なループは以下の 2 種類に大別できる：1. if-then-else に代表される制御構造を持つもの，2. リスト・ベクトル・アクセスによって可能となるような不規則なメモリ・アクセスがあるもの．1. に関しては，実行許可フラグやマスクによって対処は可能であるが，then パートと else パートを逐次的に実行することになるため，パスの複雑さにあわせて実行効率は何倍も悪化することがある．2. に関しては，スーパコンピュータのようなベクトル・プロセッサとは異なり，リスト・ベクトル機能を備える SIMD 命令セットは現存しない．

アプリケーションの傾向

SIMD 化困難なループは，一定数存在する．ソートはその代表例である．バブル・ソートなどの単純なアルゴリズムならば，SIMD 化は可能である．しかし，クイック・ソートやマージ・ソート等の実用的なアルゴリズムでは，キーの値に依存してメモリ・アクセスが不規則になるため，SIMD 化は困難である．また近年では，ユーザや市場の要求に応えるため，ベクトル処理は複雑化する傾向にあり，このことが SIMD 化を難しくしている．その好例として，最新の動画コーデックである MPEG-4 H.264 AVC があげられる．まず H.264 では，「重い」処理のほとんどは，その「重さ」ゆえ，(r, g, b, α) に対してではなく，輝度 Y に変換した後に行われる．そのため，画像処理の 1 種であるにもかかわらず，(r, g, b, α) の 4 つ組を 1 命令で処理するような典型的な SIMD 化はそもそもできない．

そのような「重い」処理の中には，SIMD 化困難な部分が多く存在する．たとえば，最も重い処理の 1 つである動き検出では，計算量の削減のために多重ループからの脱出を行う必要がある．デブロッキング・フィルタでは，入力データに応じて適応的にアルゴリズムを変更する必要がある．また H.264 では，Context-Adaptive Binary Arithmetic Coding (CABAC) と呼ばれる符号が用いられていて，エンコードに加えてデコードも複雑になっている．デコードは計算能力の低いクライア

ントでも行う必要があるため，より深刻である．もともとデコードは，多数の分岐を含むため SIMD 化が困難であるが，これに加えて CABAC では，コンテクストに応じた適応的なモードの切り替えなどを行う必要がある．

プログラマビリティ

SIMD の way 数を増やせば，最大性能は向上する．しかし，SIMD のみでは，SIMD 化困難なループに対してまったくスピードアップが得られない．最近主流の，SIMD 命令セットを備えるマルチコア・プロセッサは，SIMD ユニットの way 数ではなく，コアの数を増やすことによってスケーラビリティを確保しているが，このこともその理由の 1 つにあげられる．

結局 SIMD は，最大性能と引き換えに，プログラマビリティを失っているのだと言える．ここでいうプログラマビリティとは，単にプログラマによるプログラミングの容易性を指しているのではない．不規則なループでは，実行される命令自体がイタレーションごとに異なるので，“Single-Instruction” では本質的に対処が難しい．SIMD で不規則なループを扱う場合には，コンパイラによる対応が困難であることはもちろん，たとえ熟練したプログラマが機械語を用いてプログラミングを行ったとしても，高い効率は望めない．

2　研究の目的

最大性能とプログラマビリティの観点からは，スーパスカラ・プロセッサは，SIMD の対極に位置する．本研究では，SIMD プロセッサに匹敵する最大性能と，スーパスカラ・プロセッサに匹敵するプログラマビリティを両立するベクトル・アーキテクチャの実現を目指す．具体的な目標としては，peak performance ではなく，sustained performance において，SIMD 命令セットを備えるマルチコア・プロセッサを超えることにあり，スーパコンピュータから専用 LSI まで，あらゆるベクトル処理向けプロセッサの各コアの SIMD ユニットを提案アーキテクチャで置き換えることを狙う．

提案するアーキテクチャは，以下のとおりである：全体は複数のコアからなるマルチコア・プロセッサであり，各コアは，n 個（n は 4〜8）の演算器クラスタからなる．ループの各イタレーションに対してスレッドを生成し，それぞれを各クラスタで並列に実行する．イタレーション番号 i のスレッドは，$i \bmod n$ 番のクラスタに割りつけられる．ループの並列実行をターゲットとするマルチスレッド・アーキテクチャには数多くの提案がある．その中でも Vector-Thread Architecture は提案するアーキテクチャと非常によく似ている．これらの先行研究と提案アーキテクチャは，以下ように，その目的からして大きく異なる：

1. 各クラスタ内に割りつけられた複数のスレッドを切り替えることにより，（分岐予測ミスを起こす）分岐命令や（キャッシュ・ミスを起こす）ロード命令によって生じる長いレイテンシの隠蔽を図る．先行研究は，従来のベクトル・アーキテクチャ同様の規則的なループを主なターゲットとしており，これらのレイテンシについてはほとんど考慮されていない．

2. プログラムのあらゆる部分からではなく，ループのイタレーションからのみスレッドを生成することに特化して，各演算器クラスタの面積オーバヘッドの削減を目指す．先行研究は，いわゆる台数効果を指向しており，回路面積についてはほとんど評価の対象になっていない．

そのための具体的なアイディアは，次章で詳しく述べる．本研究では，これらのアイディアを 1 つのアーキテクチャとしてまとめあげ，予備実装による評価を通して有効性を検証する．

3　研究の方法

(1)　Switch-on-Future-Event マルチスレッディング

不規則なループを効率よく処理するためには，マルチスレッディングによってレイテンシを隠蔽することが有効である．たとえば分岐を含んだループである場合，各イタレーションをスレッドとし，（分岐予測ヒット率が低い）分岐命令をフェッチしたら，別のスレッドの命令のフェッチを開始すればよい．対象はループ内のイタレーションであるので，隠蔽に必要なスレッド数は十分にあると考えることができる．

各コアでは，スレッドを切り替えながら if 部分（分岐命令とそれが依存する数命令）のみをフェッチし，実行する．分岐命令の実行が完了すると，元のスレッドに制御を戻すことにより，即座に後続の then/else 部分を実行することができる．不規則なループをそのまま実行した場合，分岐予測ミスやメモリ・アクセスのレイテンシにより，コアの実行効率は大きく悪化することが多い．マルチスレッディングは，このようなレ

イテンシを非常に効果的に隠蔽できるため，全体として実行効率を大幅に向上させることが可能である．

ただし，マルチスレッド・プロセッサでは，必要な物理レジスタ数は同時に実行されるスレッド数に比例して増えるため，軽量なマルチスレッディングの実装を行う必要がある．これは，以下で述べる非レイテンシ指向レジスタ・キャッシュにより実現できる．

(2) 非レイテンシ指向レジスタ・キャッシュ・システム

一般的なレジスタ・キャッシュ(RC) は，当然のことながら，RC ヒットを仮定したパイプライン構造を取る．それに対し，提案する非レイテンシ指向 RC システムでは，いわば，RC ミスを仮定したパイプライン構造を取る．すなわち，RC のヒット/ミスに関わらず，メイン・レジスタ・ファイルのレイテンシだけ待った後に実行を行うのである．RC アクセス後に用意されているステージでは，ヒットの場合には何も行わず，ミスの場合にのみメイン・レジスタ・ファイルへのアクセスを行う．RC ミスを起こした命令は，ポートに空きがある場合は時分割でメイン・レジスタ・ファイルから値を得ることができ，その場合は全くパイプライン・ストールを起こさずに実行を継続することが可能である．

上記のパイプライン構造では，メイン・レジスタ・ファイルへのアクセスはミス時にのみ行われる．このため，キャッシュと比較すると，そのアクセス量は大幅に少なく，メイン・レジスタ・ファイルを，アクセス量に見合った 2 ポート程度の RAM で構成することができる見込みである．一般に，RAM の面積はポート数の 2 乗に比例するため，これにより，必要な回路面積を大幅に削減することができる．その結果，パイプラインは数サイクル深くなり，分岐予測ミス・ペナルティが増加することになるが，分岐予測器のヒット率の向上と，マルチスレッディングによるレイテンシの隠蔽により補償することができる．

4 研究成果

(1) Switch-on-Future-Event マルチスレッディング

分岐予測ミスやキャッシュ・ミスなどのミス・ペナルティを頻繁に発生させるのは，少数の静的な命令であることはよく知られている．我々は更に，そのような命令の大部分が比較的小さなループ中で実行されていることを発見した．

このような命令に対する手法としてヘルパースレッディングがあるが，これはプロセッサの資源を消費する．そのため，ミス・ペナルティが減少しているにも関わらず性能向上が最大でも 5.2% にとどまることが評価により分かった．

一方，我々の提案する Switch-on-Future-Event マルチスレッディングではそのような資源の消費が起きないため，より大きな性能向上が期待できる．評価した結果，平均 10.5%，最大で 33.5% の性能向上となることを確認した．

(2) 非レイテンシ指向レジスタ・キャッシュ・システム

現実的な 4-way のスーパスカラ・プロセッサにおいて，非レイテンシ指向レジスタ・キャッシュ・システムは，レイテンシ 1 サイクルのレジスタ・ファイルのそれと同等にまでバイパス・ネットワークを簡略化し，メイン・レジスタ・ファイルのポート数を 12 から 4 まで削減することができる．CACTI を用いたシミュレーションにより，面積と消費電力は，それぞれ，24.9%，および，31.9% にまで削減できることが示された．既存の手法では，そこまでの削減を行うと，IPC が 83.1% にまで低下してしまう．

5 主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 7 件）

[1] Shinobu Miwa, Hironori Ichibayashi, Hidetsugu Irie, Masahiro Goshima, Hironori Nakajo, and Shinji Tomita: Low-Complexity Bypass Network Using Small RAM, Int'l Conf. on Computer Design (CDES'08), pp. 153–159 (2008).

[2] 塩谷 亮太, 入江 英嗣, 五島 正裕, 坂井 修一: 回路面積指向レジスタ・キャッシュ, 先進的計算基盤システムシンポジウム SACSIS2008, pp. 229–236 (2008).

[3] 一林 宏憲, 塩谷 亮太, 入江 英嗣, 五島 正裕, 坂井 修一: 逆 Dualflow アーキテクチャ, 先進的計算基盤システムシンポジウム SACSIS2008, pp. 245–254 (2008).

[4] Ryota Shioya, Kazuo Horio, Masahiro Goshima, and Shuichi Sakai: Register Cache System not for Latency Reduction Purpose, IEEE Int'l Symp. on Microarchitecture (MICRO-43),

pp. 301–312 (2010). DOI:10.1109/MICRO.2010.43.

[5] Ryota Shioya, Daewung Kim, Kazuo Horio, Masahiro Goshima, and Shuichi Sakai: Low-overhead architecture for security tag, IEICE Trans. on Information and Systems, Vol. E94-D, No. 1, pp. 69–78 (2011). DOI:10.1587/transinf.E94.D.69.

[6] 塩谷 亮太, 倉田 成己, 中島 潤, 五島 正裕, 坂井 修一: Switch-On-Future-Event マルチスレッディング, 先進的計算基盤システムシンポジウム SACSIS2010, pp. 157–165 (2010).

[7] 堀尾 一生, 塩谷 亮太, 五島 正裕, 坂井 修一: 面積効率を指向するプロセッサの設計, 先進的計算基盤システムシンポジウム SACSIS2010, pp. 339–346 (2010).

〔学会発表〕（計 17 件）

[1] 喜多 貴信, 塩谷 亮太, 入江 英嗣, 五島 正裕, 坂井 修一: 予測ミスした命令の実行を継続する投機手法, 情報処理学会 研究報告 2008–ARC–178, pp. 7–12 (2008).

[2] 塩谷 亮太, 入江 英嗣, 五島 正裕, 坂井 修一: 回路面積指向レジスタ・キャッシュの評価, 情報処理学会 研究報告 2008–ARC–178, pp. 13–18 (2008).

[3] 堀尾 一生, 亘理 靖展, 塩谷 亮太, 五島 正裕, 坂井 修一: ツインテール・アーキテクチャの評価, 情報処理学会 研究報告 2008–ARC–179, pp. 7–12 (2008).

[4] 塩谷 亮太, 五島 正裕, 坂井 修一: 分岐プレディシジョン, 情報処理学会 研究報告 2008–ARC–179, pp. 67–72 (2008).

[5] 安藤 徹, 塩谷 亮太, 五島 正裕, 坂井 修一: プログラムの繰り返し構造に着目した動的なヘルパースレッディング, 情報処理学会 研究報告 2008–ARC–179, pp. 139–144 (2008).

[6] 塩谷 亮太, 五島 正裕, 坂井 修一: プロセッサ・シミュレータ「鬼斬弐」の設計と実装, 先進的計算基盤システムシンポジウム SACSIS2009, pp. 120–121 (2009).（ポスター）.

[7] 伊藤 悠二, 塩谷 亮太, 五島 正裕, 坂井 修一: 最適なロールバック・ポイントを選択するネスティッド・トランザクショナル・メモリ, 情報処理学会 研究報告 2009–ARC–184, No. 5 (2009).

[8] 喜多 貴信, 樽井 翔, 塩谷 亮太, 五島 正裕, 坂井 修一: タイミング制約を緩和するクロッキング方式の予備評価, 電子情報通信学会 技術報告 CPSY2009–26, pp. 61–66 (2009).

[9] 堀尾 一生, 塩谷 亮太, 五島 正裕, 坂井 修一: 面積効率を指向するプロセッサの設計, 情報処理学会 研究報告 2009–ARC–184, No. 27 (2009).

[10] 堀尾 一生, 塩谷 亮太, 五島 正裕, 坂井 修一: 面積効率を指向するプロセッサの設計と実装, 情報処理学会 第 72 回全国大会, pp. 1-181–1-182 (2010).

[11] 伊藤 悠二, 塩谷 亮太, 五島 正裕, 坂井 修一: 最適なロールバック・ポイントを選択するネスティッド・トランザクショナル・メモリの評価, 情報処理学会 第 72 会全国大会, pp. 1-187–1-188 (2010).

[12] 倉田 成己, 塩谷 亮太, 五島 正裕, 坂井 修一: 繰り返し構造に着目した分岐プレディシジョンの改良, 情報処理学会 第 72 会全国大会, pp. 1-213–1-214 (2010). (優秀卒業論文認定).

[13] 堀部 悠平, 三輪 忍, 塩谷 亮太, 五島 正裕, 中條 拓伯: 選択的キャッシュ・アロケーション：マルチスレッド環境におけるキャッシュ利用効率の向上手法, 情報処理学会 研究報告 2010–ARC–190, No. 1, pp. 1–8 (2010).

[14] 伊藤 悠二, 塩谷 亮太, 五島 正裕, 坂井 修一: 最適なロールバック・ポイントを選択するトランザクショナル・メモリ, 情報処理学会 研究報告 2010–ARC–190, No. 9 (2010).

[15] 倉田 成己, 塩谷 亮太, 中島 潤, 五島 正裕, 坂井 修一: Switch-on-Future-Event マルチスレッディングの改良, 情報処理学会 研究報告 2010–ARC–190, No. 27 (2010).

[16] 伊達 三雄, 倉田 成己, 伊藤 悠二, 塩谷 亮太, 五島 正裕, 坂井 修一: ディスパッチト・イメージ・キャッシュ, 情報処理学会 第 73 回全国大会,pp. 1-67–1-68 (2011).

[17] 伊藤 悠二, 塩谷 亮太, 五島 正裕, 坂井 修一: 最適なチェックポイントを選択するトランザクショナル・メモリ, 情報処理学会 第 73 回全国大会, pp. 1-69–1-70 (2011).

〔その他〕

ホームページ　http://www.mtl.t.u-tokyo.ac.jp/

6　研究組織

(1)　研究代表者

五島 正裕（GOSHIMA MASAHIRO）
東京大学・情報理工学系研究科・准教授
研究者番号：90283639

(2)　研究分担者

坂井 修一（SAKAI SYUICHI）
東京大学・情報理工学系研究科・教授
研究者番号：50291290