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

2006－ARC－170（13）
2006／11／30

命令グルーピングによる効率的な命令実行方式

佐々木　　広[†]　近藤正章[†]　中村　　宏[†]

近年のマイクロプロセッサは，多命令同時発行やアウト・オブ・オーダー実行をサポートしており，非常に複雑さが増してきている．とりわけ，動的命令スケジューリング機構をはじめとするそういった機能の肝となる機構において，多命令を同時に処理するために，必要なポート数や容量は増加する傾向にあり，消費電力の増大は免れない．本稿では，依存情報を用いた命令のグループ化による動的命令スケジューリング機構の電力削減および，動的命令カスケーディング手法による効率的な命令の実行方式を提案する．

Instruction Grouping: Providing an Efficient Execution Using Dependence Information

Hiroshi Sasaki,[†] Masaaki Kondo[†] and Hiroshi Nakamura[†]

Microprocessors are becoming much more complicated than before as they support superscalar and out-of-order execution. The number of ports and the size of such central structures that includes dynamic instruction scheduling logic is becoming quite complex and thus dissipates significant energy. The proposed method groups several instruction as a single issue unit using dependence information and provides an efficient instruction execution. The present paper describes the microarchitecture mechanisms and shows evaluation results for energy savings and performance.

1. はじめに

近年の汎用マイクロプロセッサでは，多命令同時実行，アウト・オブ・オーダー実行可能なものが多く，非常に複雑さが増してきている．汎用マイクロプロセッサは，様々なアプリケーションを効率的に実行することが求められ，また旧来のバイナリコード資産を実行する場合もあることから，高速化のためには実行時に並列性を抽出することが必要となる．また，携帯電話をはじめとする携帯機器や組み込み機器の高機能化，また多様化を受け，今後はそれらに搭載される組込型プロセッサにおいても，汎用のプログラムを高速に実行することが必須となり，多命令同時実行やアウト・オブ・オーダー実行といった動的な命令レベル並列性抽出技術が必要になると考えられる．

しかし，多命令同時実行，アウト・オブ・オーダー実行を行う動的命令スケジューリング機構の問題点として，ハードウェアの複雑化による消費電力の増大が挙げられる．特に，バッテリ駆動の携帯機器では消費電力増大は許容できない問題である．また，現在ではハイエンドシステムにおいても消費電力増大にともなう発熱量の増大が深刻化しており，複雑な動的命令スケジューリング機構を用いることが難しくなっている．このため，多命令同時実行や，アウト・オブ・オーダー実行機構のハードウェアの簡単化，低消費電力化は，非常に重要な課題である[4),7),8),15)]．

我々はこれまでに，命令キュー，および命令スケジューリング機構の消費電力削減手法として，「依存情報を用いた命令グループ化による動的命令スケジューリング機構の電力削減手法」[13)] を提案している．この手法は，命令間の依存情報を用いて命令をグループ化し，そのグループを一つの命令発行単位として扱うことで，命令キューやスケジューリング機構のサイズ／ポート数の増加を抑えつつ，より多くの命令の保持，および発行を行うものである．本機構により，従来の動的命令スケジューリング機構よりも少ないハードウェア量で，ほぼ同等，あるいはそれ以上の性能が得られ，また消費電力を大きく削減できる．

また，依存情報を用いるという着眼点を命令の演算部分に適用する，動的命令カスケーディングというアーキテクチャ手法を用い，同期／非同期混在型のプロセッサ構成である GALS (Globally-Asynchronous Locally-Synchronous) 型のマイクロプロセッサの高性能化を目的とした，「GALS 型プロセッサにおける動的命令カスケーディング」[17)] を提案している．動的命令カスケーディングとは，命令キューから発行された命令の実行において，データ依存によって並列には実行できない 2 命令を同一サイクルに実行することに

† 東京大学 先端科学技術研究センター
Research Center for Advanced Science and Technology, The University of Tokyo

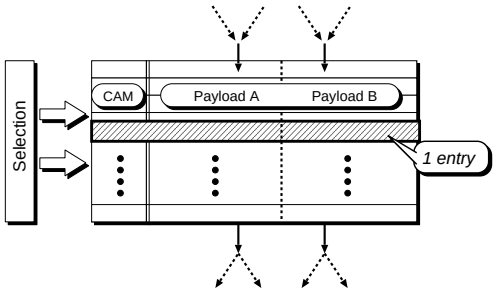

図 1 命令グルーピングのイメージ図

よって，並列性が乏しいプログラムにおいても，余剰となっている演算器を有効に活用し高性能化を図る手法である．

本稿では，これまでに提案している両手法を命令グループ化という観点から組み合わせることによって，動的命令スケジューリング機構においては，依存のある 2 命令を単一の発行単位として扱い，そのあとに各々独立にパイプラインのバックエンドに流すのではなく，動的命令カスケーディングを用いて同一サイクルで処理するという手法を提案する．本手法によって動的命令スケジューリング機構の消費電力を大幅に削減し，なおかつ性能の向上を得ることができる．

本論文の構成は以下のとおりである．次節において，提案手法のアイデア，およびマイクロアーキテクチャを示す．3 章では性能評価環境および評価条件について説明し，4 章で評価結果を示す．5 章で関連研究についてまとめ，6 章で本論文のまとめと，今後の課題について述べる．

2. 提 案 手 法

本章では，それぞれ文献[13),17)] で我々が提案している，命令グループ化による動的命令スケジューリング機構の消費電力削減手法，および動的命令カスケーディングの概要を示し，それらをどのように組み合わせるかについて述べる．

2.1 動的命令スケジューリング機構の電力削減手法

この手法は，複数の命令をグループ化し，1 つの発行単位として扱うことによって動的命令スケジューリング機構のサイズ／ポート数を低減するものである．ディスパッチステージにおいて複数の命令をグループ化し，発行時までのステージにおいてグループ化された命令を 1 命令として扱う．4 命令発行のプロセッサにおいて 2 命令をグループ化した場合のイメージ図を図 1 に示す．ペイロードエリアに 2 命令がグループ化して格納され，グループ化した命令を 1 命令としてウェイクアップ，セレクト，そして発行すれば，実質 2 命令に対してそれらの処理を施したことになる．つまり，命令数に対して必要とされるポート数が半減できることになり，命令キューの複雑さは低減され，大幅な消費電力削減につながると考えられる．

しかしながら，従来の動的命令スケジューリング機構と同等のスループットを確保するためには，できる限り多くの命令をグループ化する必要がある．例えば，4 命令を同時に発行可能なプロセッサにおいて全くグループ化することができなかった場合，最大で 2 命令しか同一サイクルに発行することができなくなる．どのような命令をグループ化するか，およびグループ化された命令を 1 つの発行単位として扱うためのマイクロアーキテクチャの拡張について次節において述べる．

2.1.1 命令のグループ化

この手法では，一方の命令発行の次サイクルに確実に他方の命令が発行可能となる命令の組をグループ化する．したがって，2 命令を発行するためには先に発行される方の命令のみをウェイクアップ・セレクトし，この命令が発行された 1 サイクル後に他方の命令を発行すればよい．グループ化する 2 命令を検出するためのハードウェアを簡素にするため，先行する命令には単一サイクルの実行レイテンシを持つ整数演算命令を対象とする．

2.1.2 グループ化可能な命令の条件

この手法においてグループ化可能な命令の組は以下の 2 つに大別される．

[1] 一方の発行が他方を発行する唯一のトリガー
[2] 同一サイクルに実行が可能

まず，[1] について説明する．下記の 2 命令において，命令 2 は右オペランドがレディであり，左オペランド r5 は命令 1 のデスティネーションとなっている．

$$\begin{cases} \text{命令 } 1 : add\ r5 \leftarrow r3, r2 \\ \text{命令 } 2 : add\ r4 \leftarrow r5, R \end{cases} \quad (1)$$

つまり，命令 1 が発行されるというただ一つの条件によって依存は解消され，命令 2 は次サイクルに発行可能となるため，この 2 命令はグループ化可能といえる．

次に [2] について説明する．以下に示す 2 命令はどちらも両オペランドが揃っている，レディな命令である．

$$\begin{cases} \text{命令 } 1 : add\ r1 \leftarrow R, R \\ \text{命令 } 2 : add\ r2 \leftarrow R, R \end{cases} \quad (2)$$

この 2 命令には直接の依存関係はないが，どちらもすでにレディな状態にあるため，グループ化することが可能である．この場合，命令オーダーの若い命令が，先に発行される命令として格納される．どちらもレディな 2 命令をグループ化しなかった場合，同じサイクルに両命令を実行できることもあり，グループ化することによって性能的なペナルティが発生する場合もあると考えられる．しかし，この手法においては命令をなるべく多くグループ化し，スループットを確保することによって性能低下を防ぐ必要がある．また，レディな命令はキューにとどまっているサイクル数が短く次から次へと捌けていくためクリティカルな命令であることは少なく，実行が 1 サイクル遅れてもほとんど性能には影響がないと考えられる．上記のような

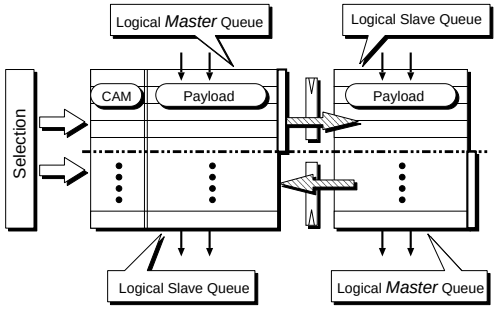

図 2 提案する動的命令スケジューリング機構

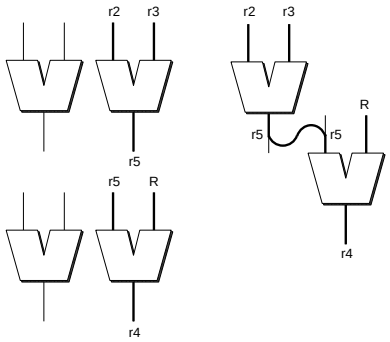

図 3 動的命令カスケーディングのイメージ図

理由から，この手法においてはこういったレディな 2 命令もグループ化の対象とする．

2.1.3 提案する動的命令スケジューリング機構

2 命令を 1 命令としてグループ化する場合における，提案手法の動的命令スケジューリング機構の概要を図 2 に示す．グループ化される 2 命令はそれぞれが図に示すマスターキュー (Master Queue) とスレーブキュー (Slave Queue) の同じエントリ番号のエントリに書き込まれ，以後，1 エントリとして扱われる．マスターキューには対応するスレーブキューのエントリに命令が格納されているかどうかを示す 1 ビットのフラグがあり，スレーブキューに命令を書き込む際，同時にこのフラグを立てる．1 サイクル内に各々最大でマスター・スレーブキューはそれぞれ，ディスパッチ時に 2 命令ずつ書き込みを行うことができ，2 命令を発行可能である．また，セレクトロジックは 2 命令をマスターキューのエントリからセレクトする回路を備えている．マスターキューとスレーブキューの間にはラッチが設けられており，マスターキュー内の命令がセレクトされ，発行のためにアクセスされる際，スレーブキューに命令が格納されていることを示すフラグが立っていたら，ラッチを介して次サイクルにスレーブキュー内の命令がアクセスされ，発行される．したがって，提案する動的命令スケジューリング機構において，ウェイクアップ時に連想マッチを行うための CAM ロジックはマスターキューのみが有している．

図中，キューの中央に引かれている点線を境に，上部は左側がマスターキュー，右側がスレーブキューとなっているが，下部については逆に，左側がスレーブキュー，右側がマスターキューとなっている．このような実装によってディスパッチ時の自由度が増し，例えばマスターキューに 3 命令，スレーブキューに 1 命令を書き込むという要求にも上部左側のマスターキューに 2 命令，下部右側のマスターキューに 1 命令そして上部右側のスレーブキューに 1 命令を書き込むことによって応えることが可能となっている．

上記の実装によってディスパッチ時に命令をグループ化し，そのグループを一つの命令発行単位として扱うという提案手法の論理的動作がポート数削減の影響をほとんど受けることなく実現可能となる．動的命令スケジューリング機構のサイズが比較的小さい場合には十分な ILP を抽出することができず，そのサイズが性能のボトルネックとなる．このような場合，従来の動的命令スケジューリング機構と比べて高い性能を達成することが期待できる．また，サイズが十分に大きい場合にはほぼ同等の性能を達成することが可能であると考えられる．消費電力に関しては，動的命令スケジューリング機構のサイズに関わらず，複雑さが低減したことによる大幅な削減が期待できる．

2.2 動的命令カスケーディング

続いて，動的命令カスケーディングの概要について述べる．汎用マイクロプロセッサの多くはパフォーマンスを向上させるためにプログラムの ILP をできる限り抽出し，1 サイクルになるべく多くの命令を実行するために，アウト・オブ・オーダー実行やスーパースカラといった手法を用いている．したがって，同一サイクルに実行できる命令が多ければ多いほど，高い性能が期待できるが，多くの汎用アプリケーションにおいては ILP が命令の発効幅や演算器などの資源に対して十分でない場合がほとんどである．つまり，プログラム実行中の大半のサイクルにおいて，多くのハードウェア資源が使われずにいることが多い．動的命令カスケーディング手法は，このような使用頻度の少ない演算資源を有効に活用することを目的としている．メインとなるアイディアは，「依存関係にある 2 つの命令を 1 サイクルで実行する」，というものである．ここで，依存関係にある 2 命令とは，片方の命令の演算結果がもう一方の命令の入力オペランドとなるような命令のペアである．我々は提案する手法によって依存関係にある 2 命令を同一サイクルに実行するための条件として，プロセッサの電源電圧を高く保ったままクロック周波数を低下させることが必要であると考えている．上記の条件によってサイクルタイムは延長され，信号が伝搬する距離は伸び，サイクルタイムあたりの論理段数は増加し，その結果 2 命令を 1 サイクル内に完了することが可能になる．

例として，命令列が 2.1.2 節の (1) のように続いて

いる場合を想定する．この 2 命令は先ほどグループ化可能であると述べたが，命令 1 の結果が命令 2 の入力オペランドとなっているため，実行順を変えることはもちろん，並列に実行することもできない．したがって，図 3 の左側のように従来のプロセッサ上の ALU でこの 2 命令を実行したとすると，演算の完了に 2 サイクルを要する．ここで，動的命令カスケーディングを適用した場合，命令 2 の右オペランド r1 が ready であればこの 2 命令は同一サイクル内に実行可能となる．この場合図 3 の右側に示すように，1 サイクルでこの 2 命令を実行可能となる．

この手法を用いることにより，本来ならば次のサイクルまで待たなければ実行することができなかった命令が同じサイクル内に実行可能となるため，同時発行命令数は増え，それゆえ Committed Instructions Per Cycle (IPC) は増加する．

2.3 命令グルーピングによる効率的な命令実行

次に，本章で説明してきた，依存のある 2 命令を抽出し，利用するという特徴を持った 2 つの手法を組み合わせることによって，効率的な命令実行を行う手法について述べる．以前の手法では依存のある 2 命令をディスパッチ時において発見し，それらを 1 グループとして動的命令スケジューリング機構のみにおいて処理をしていた．そのあと，これら依存のある 2 命令は，先行する命令が発行され次第，連続したサイクルでパイプラインのバックエンドに流される．本稿では，これら依存のある 2 命令を，連続したサイクルで処理するのではなく，同一サイクル内に動的命令カスケーディング手法を用いて演算することを提案する．本提案手法により，動的命令スケジューリング機構における必要なポート数を減らし，その複雑さ・消費電力を削減するだけでなく，依存のある 2 命令を動的命令カスケーディング手法で処理することによって性能も向上させることが可能となると考えられる．

3. 評　　価

3.1 評 価 環 境

本稿において提案している命令実行方式の性能と消費電力への影響を調べるため，SimpleScalar Tool Set[2] を用いたシミュレーションにより評価を行う．なお，図 2 に示す動的命令スケジューリング機構の評価を行うため，SimpleScalar のマイクロアーキテクチャに変更を加えている．また，消費電力の評価には，Wattch[3] を用いる．

評価プログラムは，SPEC CPU2000[1] の整数ベンチマーク全て (*ref* インプットセット) および，MediaBench ベンチマーク群から mpeg2 エンコードのプログラムを用いる．SPEC CPU2000 についてはプログラムの最初の 2 億命令を fast-forward し，200 万命令をシミュレーションした．

表 1　評価におけるプロセッサの仮定

Fetch & Decode width	4
Branch prediction	Combined bimodal (4K-entry) gshare (4K-entry), selector(4K-entry)
BTB	1024 sets, 4-way
Mis-prediction penalty	3 cycles
Instruction queue size - floating-point	 32
Issue width - integer - load/store - floating-point	 4 2 2
Reorder buffer size	96
Commit width	4
L1 I-cache	32 KB, 32 B line, 2-way 1-cycle latency
L1 D-cache	32 KB, 32 B line, 2-way 2-cycle latency
L2 unified cache	512 KB, 64 B line, 8-way 10-cycle latency
Memory latency	100 cycles
Bus width	16 B
Bus clock	1/4 of processor core

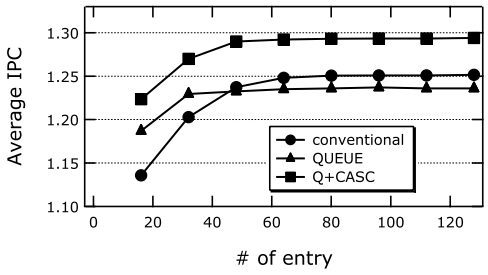

図 4　IPC (全プログラムの平均)

3.2 評価の仮定

表 1 に，本評価におけるプロセッサの仮定を示す．また，命令キューは Alpha 21264[16] に搭載されている，整数命令とロード・ストア命令がディスパッチされるものを仮定している．したがって，先行する整数演算命令とそれに依存のあるロード命令の組み合わせもグループ化することが可能である．評価では，命令キューのサイズを変化させ，従来の動的命令スケジューリング機構と比較する．

また，グループ化対象となる命令を検出するためのハードウェアの消費電力については無視できるものとし，評価には加えていない．動的命令カスケーディングを行うために周波数を下げる必要があると考えられると前章で述べたが，本評価では純粋な IPC の向上率を調べる目的のため，周波数を低下させることなく 2 命令を同時に実行することが可能であると仮定する．

4. 評 価 結 果

4.1 性　　能

まず，従来型の動的命令スケジューリング機構および提案手法において，命令キューのエントリサイズが性能に与える影響を調べるため，図 4 に命令キューのエントリサイズを変化させた場合の，評価に用いた全プログラムの平均 IPC を示す．図中，conventional

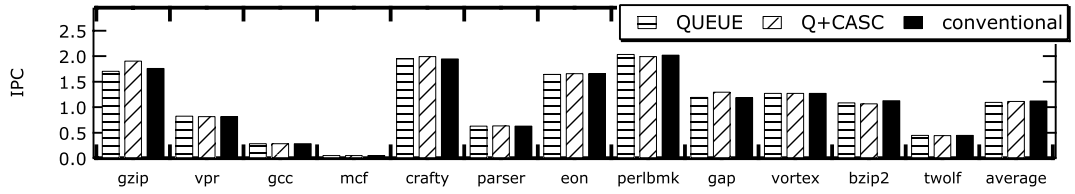

図 5 プログラム毎の IPC (命令キューのエントリサイズ：48)

は従来型における結果を，QUEUE は動的命令スケジューリング機構の消費電力削減手法のみを適用した場合における結果を示している．また，Q+CASC は上記に加えて，動的命令カスケーディング手法を加えた本稿の提案手法における評価結果である．ここで，評価結果における命令キューのエントリサイズが従来型と提案手法で等しい場合，従来型の命令キューは 1 エントリに 1 命令を格納するが，提案手法では図 2 に示すように，1 エントリに 2 命令を格納する違いがあることに注意しなければならない．図 4 より，全ての場合で，命令キューのエントリサイズが増加するにつれて IPC が向上しているのがわかる．これは，命令キューの保持できる命令数が増えるため，より ILP を抽出することができるためである．

エントリサイズが 64 以上の場合，QUEUE, Q+CASC の conventional に対する性能の比率はそれぞれ–1.1%，+4.6%である．以前の提案手法である QUEUE においては conventional に対して若干の性能低下が見られたが，本提案手法である Q+CASC によって QUEUE に対してだけでなく，conventional よりも高い性能を達成していることがわかる．これによって，ポート数の削減による性能の低下分を，動的命令カスケーディングによって IPC という指標の上では補うことが可能であるといえる．また，図 5 に，評価に用いた全プログラムにおける IPC を示す．例えば，*mpeg2*，*gzip*，*gap* などのプログラムにおいて Q+CASC は conventional に対して 5–10%程度高い性能を達成しており，これらのプログラムにおいては，依存のある 2 命令によるクリティカルパスを，1 命令で実行したことによって実行時間が大幅に短縮されたと考えられる．また，本評価では周波数を低下させることなく動的命令カスケーディングが可能であると仮定しているが，実際には多少の周波数低下を許さないと実現は厳しいと考えられる．周波数の低下を許すと本提案手法で得られた IPC の向上分が相殺されてしまうかもしれないが，この場合の対処法として，文献[17] で示したように GALS 型の構成で ALU 部分のみ周波数を低下させるという手法を用いることによって，周波数低下による性能の低下を最小限にし，conventional よりも高性能を達成することが可能であると考えられる．

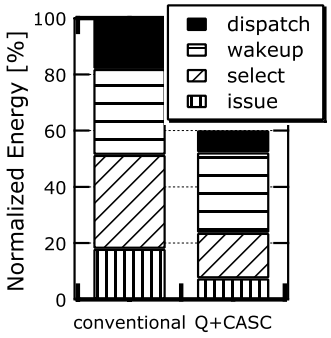

図 6 Q+CASC におけるステージごとの消費電力の内訳 (命令キューのエントリサイズ：48)

4.2 消費電力

提案手法は，2 命令をグループ化し，動的命令スケジューリング機構において 1 つの発行単位として扱うことによって，サイズ／ポート数を削減し，命令キューの消費電力を削減する．図 6 は Q+CASC において従来のプロセッサに対する，動的命令スケジューリング機構の消費電力削減率の内訳を示したものである．棒グラフは上からディスパッチ・ウェイクアップ・セレクト・発行の各ステージにおける結果を示している．図から，従来の動的命令スケジューリング機構に対してディスパッチ・ウェイクアップ・セレクト・発行の各ステージにおいてそれぞれ，約 46%，7%，48%，48%の消費電力を削減したことがわかる．ウェイクアップステージを除く 3 ステージにおいて大幅な電力削減を達成している．ウェイクアップステージの消費電力は，ブロードキャストされてきたタグとの連想マッチを行うための CAM の高さ (エントリ数) に支配されており，どちらもエントリ数が 48 のためほぼ同じ結果となっている．しかし，動的命令スケジューリング機構の消費電力において，ウェイクアップステージの消費電力が支配的な構成の場合は，提案手法において性能が大きく向上しているため，その向上分を電力に変換するという観点から，さらにエントリ数を減らすという手段が考えられる．エントリ数 64 の conventional の性能とエントリ数 32 の Q+CASC を比較すると，Q+CASC が約 2.5%conventional を上回っている．ま

た，そのとき Q+CASC の conventional に対する消費電力削減率はディスパッチ・ウェイクアップ・セレクト・発行の各ステージにおいてそれぞれ約 70%，50%，74%，80%となっており，同じエントリ数同士の比較よりもさらに大幅な消費電力削減を達成している．

5. 関 連 研 究

従来より，動的命令スケジューリング機構の大容量・多ポート化による複雑さや消費電力の増大の問題への対処を目的とした手法，[4)~8),10)~12),15),16)] および命令をカスケーディングすることによって性能向上を達成したり，ハードウェアを効率的に使用する手法は多く提案されている[9),14)]．

単一サイクルレイテンシの処理を取り除くという視点から，命令を Macro-op と呼ばれる単位にグループ化し，ウェイクアップとセレクトのステージをパイプライン化することにより動的命令スケジューリング機構の複雑度を低減する研究が行われている[9)]．また，LSI のランダムなプロセスばらつきを考慮し，プロセッサの論理段数を大きくするために命令をカスケーディングする研究も行われている[18)]．これらの研究は命令をグループ化するという視点は同じであるが，実行レイテンシが単一サイクルである命令をなくすことを目的としている点が本研究とは異なっている．

本稿で提案する命令グループ化による動的命令スケジューリング機構の消費電力削減手法と動的命令カスケーディングを組み合わせる手法は，動的に依存のある命令を検出することで，たとえば汎用プロセッサのように，さまざまな特徴を持つプログラムを実行するような場合にも対応できることも利点として挙げられる．

6. まとめと今後の課題

本稿では動的命令スケジューリング機構の消費電力削減，および性能の向上を目的とした，命令グループ化による効率的な命令実行手法を提案した．提案手法はグループ化した命令を単一の発行単位として扱うことによって，動的命令スケジューリング機構の複雑さを低減し，動的命令カスケーディングによって実行サイクル数を短縮するものである．

提案手法を評価した結果，従来の動的命令スケジューリング機構を有するプロセッサと比較して，エントリ数にかかわらず高い性能を達成しつつ動的命令スケジューリング機構の消費電力を大幅に削減可能であることがわかった．今後，動的命令スケジューリングを行うステージだけでなく，他のパイプラインステージにおいても命令のグループ化を適用する方法を検討していく予定である．また，本評価では周波数を低下させることなく動的命令カスケーディングが可能であると仮定したが，周波数の低下が避けられない場合の，GALS 型プロセッサにおける評価を行うことも課題である．

謝辞　本研究の一部は，(株) 半導体理工学研究センターとの共同研究によるものである．

参 考 文 献

1) The Standard Performance Evaluation Corporation (SPEC). http://www.specbench.org.
2) Austin, T. M., Larson, E. and Ernst, D.: SimpleScalar: An Infrastructure for Computer System Modeling., *IEEE Computer*, Vol.35 (2002), 59–67.
3) Brooks,D., Tiwari,V. and Martonosi,M.: Wattch: a framework for architectural-level power analysis and optimizations., in *ISCA*, 2000.
4) Buyuktosunoglu,A., Karkhanis,T., Albonesi,D.H. and Bose,P.: Energy Efficient Co-Adaptive Instruction Fetch and Issue., in *ISCA*, 2003.
5) Canal,R. and González,A.: Reducing the complexity of the issue logic., in *ICS*, 2001.
6) Folegnani,D. and González,A.: Energy-effective issue logic., in *ISCA*, 2001.
7) Goshima,M., Nishino,K., Kitamura,T., Nakashima, Y., Tomita, S. and Mori, ichiro S.: A high-speed dynamic instruction scheduling scheme for superscalar processors., in *MICRO*, 2001.
8) Kim,I. and Lipasti,M.H.: Half-Price Architecture., in *ISCA*, 2003.
9) Kim, I. and Lipasti, M. H.: Macro-op Scheduling: Relaxing Scheduling Loop Constraints., in *MICRO*, 2003.
10) Lebeck, A. R., Li, T., Rotenberg, E., Koppanalil, J. and Patwardhan,J.: A Large, Fast Instruction Window for Tolerating Cache Misses., in *ISCA*, 2002.
11) Michaud,P. and Seznec,A.: Data-Flow Prescheduling for Large Instruction Windows in Out-of-Order Processors., in *HPCA*, 2001.
12) Raasch,S.E., Binkert,N.L. and Reinhardt,S.K.: A Scalable Instruction Queue Design Using Dependence Chains., in *ISCA*, 2002.
13) Sasaki,H., Kondo,M. and Nakamura,H.: Energy-Efficient Dynamic Instruction Scheduling Logic through Instruction Grouping., in *ISLPED*, 2006.
14) Sassone, P. G. and Wills, D. S.: Dynamic Strands: Collapsing Speculative Dependence Chains for Reducing Pipeline Communication., in *MICRO*, 2004.
15) Sharkey,J.J., Ponomarev,D.V., Ghose,K. and Ergin,O.: Instruction packing: reducing power and delay of the dynamic scheduling logic., in *ISLPED*, 2005.
16) Taylor, S. A., Quinn, M., Brown, D., Dohm, N., Hildebrandt, S., Huggins, J. and Ramey, C.: Functional Verification of a Multiple-issue, Out-of-Order, Superscalar Alpha Processor - The DEC Alpha 21264 Microprocessor., in *DAC*, 1998.
17) 佐々木広, 近藤正章, 中村宏：GALS 型プロセッサにおける動的命令カスケーディング, 情処研報 2005-ARC-164, (2005).
18) 佐藤寿倫：命令レベル逐次プロセッサ, 情処研報 2006-ARC-169, (2006).