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

2004－ARC－159 (9)
2004／7／30

メモリ依存予測を利用したフォワーディング局所化手法

入江英嗣[†,††] 高田正法[†] 坂井修一[†]

実行コアを複数の実行クラスタへ分割する“クラスタ型アーキテクチャ”は、広い実行幅と高クロック動作の両立を実現する方法として注目されている。クラスタ型アーキテクチャに関する様々な研究が行われている一方で、分散局所化の難しいメモリ参照処理がボトルネックとなることが指摘されている。各クラスタに小容量のD0キャッシュ(以下、D0)を設けるための手法はいくつか検討されているが、曖昧な依存関係や実行遅延予測のため、効果的な構成とすることが難しい。

我々は既に、メモリ参照を分散局所化する手法として、メモリ依存予測を利用した“分散投機メモリフォワーディング”を提案している。本論文では、複製D0と分散投機メモリフォワーディングの比較評価を行い、それぞれの有効性を検討する。評価から、実行遅延予測失敗によるペナルティのため、プリミティブな複製D0が有効となるためには、大きなD0容量が必要な事が分かった。一方、分散投機メモリフォワーディングは適用率に限界があるものの、実行遅延予測失敗によるペナルティがないため、クラスタ内に小容量バッファしか利用できない場合に有効である事が分かった。

Memory Forwarding Localization Using Dependence Prediction

Hidetsugu Irie,[†,††] Masanori Takada[†] and Shuichi Sakai[†]

Clustered Microarchitecture design which partitions its execution core into multiple execution clusters attract attention as the way to achieve wide and fast processing. While there are various studies on Clustered Microarchitectures are done, it is often pointed out that the cache access overheads limit its performance. To overcome this bottleneck, we have proposed the technique “Distributed Speculative Memory Forwarding” which uses memory dependence prediction and localize memory processing.

In this paper, we estimate the proposed technique and the “Replicated Intra Cluster Cache”, and compare their efficiency. Simulation results show that the scheduler replay caused by cache misses severely degrades performance of intra cluster cache. On the other hand, proposed technique shows good performance despite small buffer sizes.

1. 序　　論

情報処理の中核となるマイクロプロセッサには常に性能向上が期待されている。マイクロプロセッサはデバイス微細化、スーパパイプライニング技術、並列実行技術によって性能を向上させてきた。しかし、現行のスーパスカラ方式は、肥大したデータパスに配線遅延が大きく影響することが予想され、性能向上の限界を指摘されている。

この問題に対し、次世代プロセッサの選択肢として、クラスタ型アーキテクチャが注目されている。クラスタ型アーキテクチャでは実行コアを複数のクラスタに分割し、発行キュー、レジスタ、データパス等を局所化する (図 1)。処理が複数クラスタに分散することによってオーバヘッドが生じるが、タイミング・クリティカルパスとなっていたユニットが小型化し、更なるスーパパイプライニング効果が期待できる。

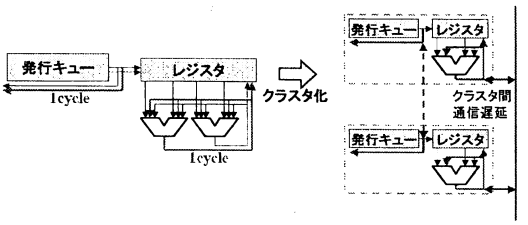

図 1　クラスタ型実行コア

このような利点を背景に、クラスタ型アーキテクチャの研究が盛んに行われているが、一方で、メモリ参照がボトルネックとなる事が指摘されている。[8),9)] 等、多くの関連研究では、各クラスタで共有される集中型キャッシュ構成を仮定しているが、この構成は高

† 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, University of Tokyo
†† 科学技術振興機構
Japan Science and Technology Agency

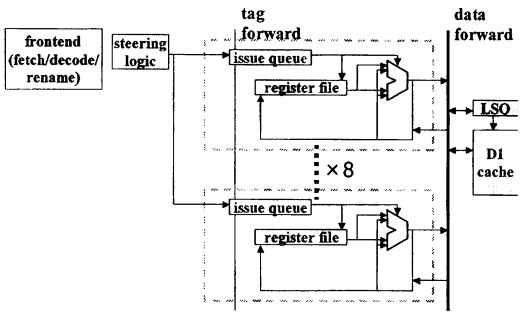

図 2 ベースライン・アーキテクチャ

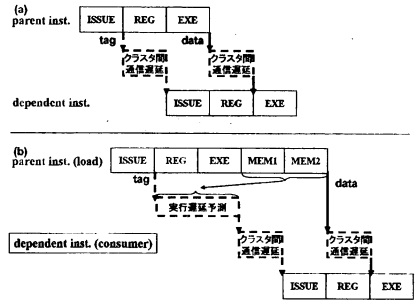

図 3 実行パイプライン

速な参照を期待できない。一方、各クラスタに D0 を載せて高速化を図る手法は、曖昧なメモリ依存関係や、実行遅延予測ミスのペナルティ等のため、性能向上に結びつけることが難しい。

そこで、13) で我々は、高クロック指向のクラスタ型アーキテクチャにおいて、参照遅延、曖昧なメモリ依存、実行遅延予測等の理想化を行わない評価を行い、集中型キャッシュでは性能が制限される事を示した。また、クラスタ内のバッファを利用してメモリ参照を高速化する手法として、"分散投機メモリフォワーディング" を提案した。

本論文では、我々が提案した手法について、クラスタ内に小さな D0 を置く方式との比較評価を行う。D0 方式としては、もっとも現実的と思われる複製キャッシュ方式に着目した。比較評価を通し、それぞれの手法の有効性について検討する。

以下、本論文の構成は次のようになっている。第 2 節では想定しているベースライン・クラスタ型アーキテクチャについて述べる。第 3 節ではメモリ参照のクラスタ内局所化手法として、複製キャッシュと分散投機メモリフォワーディングを紹介する。第 4 節では、紹介した 2 手法について比較評価を行い、局所化の適用率や IPC への影響について考察する。第 5 節で、関連研究を紹介し、第 6 節でまとめを行う。

2. ベースライン・クラスタ型アーキテクチャ

2.1 アーキテクチャの概要

本論文で想定しているベースラインアーキテクチャを図 2 に示す。高速で動作する命令実行幅 1 のクラスタ 8 個を接続した構成で、高クロック動作と高 IPC 処理の両立を狙っている。ステアリングロジックをフロントエンド処理に追加し、実行クラスタを動的に決定することで、従来のプロセッサとバイナリ互換を保つ。ステアリングロジックには、レジスタ依存関係と負荷バランスを考慮する方式[8),9)] を採用している。

実行パイプラインの様子を図 3 に示す。依存関係のある命令が同じクラスタに割り当てられていた場合には、図の通信遅延は 0 サイクルとなり、連続発行が可能となる。また、スケジューラ駆動方式となっているため、実行遅延が発行時に確定しない命令の場合、実行遅延予測[5)] を行う。例えば、ロード命令の場合、D1 キャッシュヒットを仮定して後続命令が発行する図 3(b)。キャッシュミスが発生した場合、スケジューリングの正しさが保証されなくなるため、オペランド読み出し前の全ての命令について発行前の状態へ巻き戻す、スケジューラ・リプレイ処理が必要となる。

2.2 メモリ参照処理

ベースラインアーキテクチャでは、図 2 のように集中型構成の D1 キャッシュを採用し、参照遅延を長く設定している。また、曖昧なメモリ依存を考慮したスケジューリングは以下のようになっている。まず、ストア命令同士は FIFO を守って発行される。ロード命令は先行する全てのストア命令の発行を待ってから発行される。ただし、ロード命令の不必要な発行遅延を軽減するため、WaitTable[5)] や Store Set[3)] 等のメモリ依存予測手法を採用して軽減を図る。

3. メモリ参照の局所化手法

各クラスタに分散させる D0 構成法として、分散キャッシュによる手法、バンク分割による手法、複製キャッシュによる手法等が考えられる。

分散キャッシュによる手法では、それぞれのクラスタのキャッシュが動的に異なるラインを持ち、それをテーブル等で管理する。バンク分割による手法では、それぞれのクラスタに載るラインを静的に決定しておく。これらの方法は、クラスタ毎に異なるラインを保持できるため、D0 の容量効果は高くなる。しかし、メモリ命令のステアリング時には参照アドレスが決定していないため、参照するラインを持っているクラス

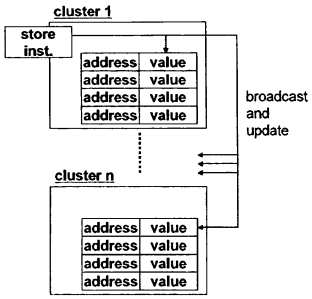

図 4 複製 D0 キャッシュ

タへの割り当てを保証できない。このため、各 D0 は他クラスタからの参照にも対応できなければならず、ポート数の面では局所化の利点が薄れてしまう。また、ロード命令の参照遅延が、クラスタ内 D0、他クラスタ D0、D1、D2 等、多くのバリエーションを持つため、実行遅延予測も難しくなる。

複製キャッシュによる手法は、ストア情報を各クラスタにブロードキャストすることにより、同じ D0 ステートを全てのクラスタで保持する。容量効果は複製数 n に応じて $1/n$ となるが、ステアリングの問題は解決する。

3.1 複製 D0 キャッシュ

複製 D0 の様子を図 4 に示す。ストア命令は割当先の D0 ストアキューへ書き込むと同時に、他クラスタの全ての D0 ストアキューへ同じ情報の書き込みを行う。このブロードキャストの反映にはクラスタ間通信遅延が影響する。また、D0 キャッシュミスが起こった場合のラインフィルも、全てのキャッシュを対象に行われる。複製 D0 では、メモリ命令のステアリング自由度は制限されない。

実行遅延予測は、D0 ヒットを仮定するため、D0 キャッシュミスはスケジューラ・リプレイを引き起こす。このため、D0 を小容量に抑えると却って性能が低下してしまうことが予想される。複製 D0 は容量効果が低いため、充分なヒット率を得るためには多くの D0 容量が必要となると考えられる。

3.2 分散投機メモリフォワーディング

フロアプラン及び参照遅延の問題から、クラスタに載せるバッファ容量は小容量に抑えられた方が有利である。小容量バッファの場合、完全な D0 適用率を得る事は難しいため、スケジューラ・リプレイのペナルティを受けない手法が望ましい。

そこで我々はクラスタ内メモリ参照手法として、投機メモリフォワーディング[11)6)] に着目した。投機メモリフォワーディングは、値予測の一手法であり、スト

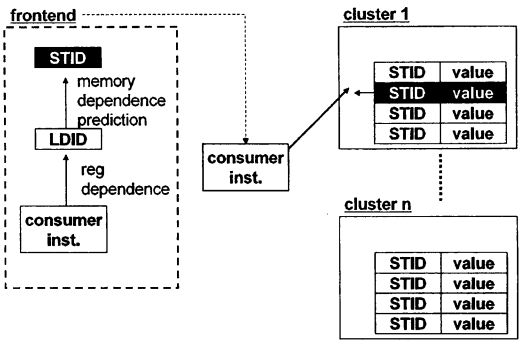

図 5 分散投機メモリフォワーディング

ア命令とロード命令の依存関係を予測して、親ストア命令のストア値を、ロード値として利用する。アドレスで参照するキャッシュ手法の場合、アドレス計算後に D0 を参照して初めて、参照局所化の適用非適用が判明し、実行遅延が確定する。このため、スケジューラは不正確な実行遅延予測に基づいて後続命令を発行しなければならない。一方、投機メモリフォワーディング手法では、親となるストア命令の ID をキーとして参照を行うため、クラスタ内に該当するストア命令の ID があるかを調べる事により、発行時に実行遅延を確定させる事ができる。

更に、投機メモリフォワーディングの前提となっているメモリ依存予測を活用し、メモリ依存のある命令同士を、同じクラスタへと割り当てるステアリングを導入する。このステアリング手法には、以下のような利点がある。まず、バッファの読み書きをクラスタ内に限定し、ポート数を削減できる。また、各クラスタ毎に異なる値を保持する事ができ、容量効果を得られる。更に、メモリ発行条件の伝搬に通信遅延が加わることを防ぐ事ができる。

このように、メモリ依存予測を前提に、フォワーディングとステアリングを行う事により、メモリ参照のオーバヘッド要素を総合的に削減する事が期待できる。我々は、この手法を “分散投機メモリフォワーディング” として提案した。なお、フォワーディング制御は、ロード命令の値を用いるコンシューマ命令に対して行われ、ロード命令は通常の D1 参照を行う。実装や制御の詳細は 13) で議論している。分散投機メモリフォワーディングの様子を図 5 に示す。

4. 評　　価

4.1 評価環境

複製 D0 方式と分散投機メモリフォワーディングの

表 1 ベースライン・パラメタ

number of clusters	8
components of each cluster	64entries IQ, 256physical registers execute 1 instruction/cycle
fetch/retire width	up to16insts/cycle
I-cache	perfect
branch predictor	gshare predictor 16k-entries, 10bit history
D-cache	64kB, 2-waySA, 8cycles hit latency 64byte lines 3 read ports 1read/write port
total pipeline depth	16stages
frontend latency	11cycles
issue to issue latency	1cycle
execute(int)	1cycle
inter cluster forwarding	2cycles

比較評価を、前述したベースライン・アーキテクチャとメモリ局所化手法を実装したトレースシミュレータにより行った。シミュレータの入力にはSPEC2000ベンチマークから11種を用い、最大256M命令を実行させた。評価に用いたベースライン・アーキテクチャのパラメタを**表1**に示す。

まず、複製D0モデルのIPCを**図6**に示す。このグラフのIPCは、メモリ依存予測にWaitTableを用いた場合の値である。横軸に各クラスタに載せるキャッシュ容量を示しており、容量の増加に伴ってIPCが増加するグラフとなっている。8クラスタ構成なので、プロセッサ全体では横軸の値の8倍のD0容量を持っている。

実線で示した3本の線が複製D0方式のIPCであり、それぞれ、設定したD0参照遅延が、"◇": 1サイクル、"×": 2サイクル、"△": 3サイクルの場合の値である。点線で示した水平の線は、複製D0を用いず、集中型のD1のみの構成におけるIPCである。基準となるベースラインの点線は、D1参照遅延が8サイクルの場合、上の方の点線はそれぞれ、D1参照遅延が1〜3サイクルだった場合のIPCである。

複製D0モデルが意味を持つ条件は、512byteから2Kbyte以上の容量が利用可能な場合であり、それ以下の容量しか利用できない場合は、スケジューラ・リプレイのペナルティを緩和するための手法を導入しなければならない。ベースラインとD0の交点に着目することにより、スケジューラ・リプレイによるペナルティを克服できる容量が分かる。

D0容量を充分に確保できる場合は性能向上を得ているが、D0参照遅延の増加や、フロアプラン上の制約とのトレードオフを考慮しなければならない。

次に、複製D0と分散投機メモリフォワーディングとの比較評価のグラフを**図7**に示す。分散投機メモリフォワーディングは、フロントエンド部分にストアセット予測を用いる。条件を同じにするため、このグラフでは依存予測にストアセット予測を用いた場合のIPCを示している。複製D0の値が図6よりも向上している理由は、ストアセット予測により依存予測が正確になっているためである。

"○" で示した分散投機メモリフォワーディングは、実行予測が正確なため、小容量時でも性能向上を得ている。一方で、容量を増やしても、128byteの時点で性能向上は飽和しており、2kbyte以上の容量が実装可能な場合は、複製D0の方が有利となっている。

また、クラスタ内に複製D0と分散投機メモリフォワーディング機構の双方を設け、両方の手法を組み合わせた場合のIPCを**図8**に示す。このグラフでは横軸はD0の容量を表しており、フォワーディング用のバッファは、512byteで一定となっている。手法を組み合わせた場合でも、複製D0のみの場合とほぼ同様の値となっている。この理由は、以下のように、分散投機メモリフォワーディングの利得が活かされないためと考えられる。まず、D0にヒットしない場合、スケジューラ・リプレイが後続命令を巻き戻してしまうため、投機フォワーディングの効果が打ち消されてしまう。一方でD0にヒットする場合は、投機フォワーディングによる参照遅延削減のゲインが少ない。このため、組み合わせによって双方の利点を活かすためには、フォワーディング適用時には実行遅延予測でD0ヒットを仮定せずD1ヒットを仮定する等、スケジューラ・リプレイ対策が必要であると言える。

図9に複製D0のヒット数、フォワーディングの適用数それぞれを、全ロード命令数に対する割合で示した。複製D0の場合、ヒット率が低いとペナルティが増えてしまう事に対し、分散投機メモリフォワーディングは、実行遅延予測が確実なため、適用率が少なくても確実に性能向上に結び付けている事が分かる。なお、分散投機メモリフォワーディングは、親ストア値を利用することによってコンシューマ命令がロード命令のキャッシュ参照完了よりも早く発行できる時のみ適用され、ロード命令のキャッシュ参照が先に完了する場合、つまりロード命令がクリティカルパスでなかった場合は適用されない。局所化の適用率に注目するとクラスタ内にフォワード可能な値を持っている割合は、実際にフォワードが行われた割合よりも約10%程高く、全ロード命令の3割程度であった。

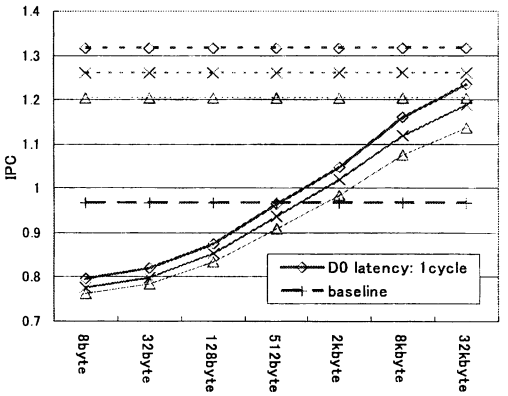

図 6　複製キャッシュ容量と IPC

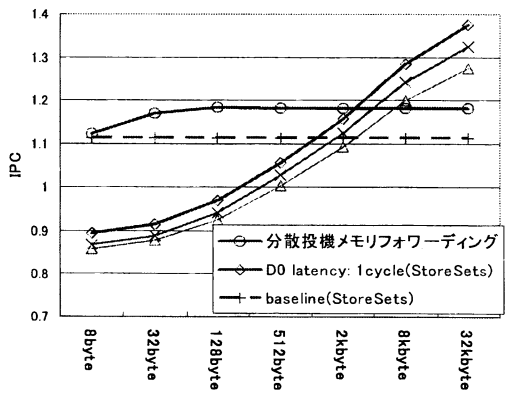

図 7　複製 vs. DSMF

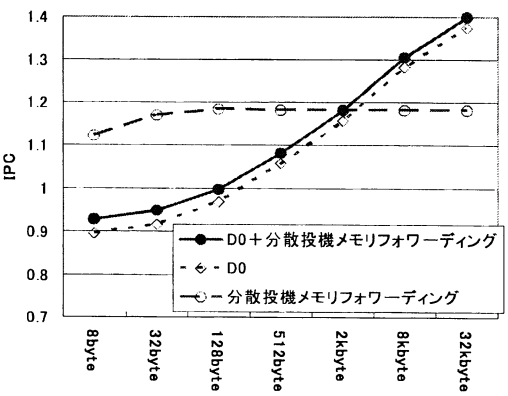

図 8　両方式の組み合わせ

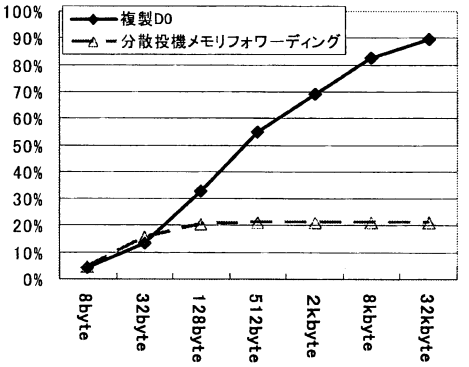

図 9　メモリ参照のクラスタ内局所化率

5. 関連研究

Palacharla ら[7] は、タイミング・クリティカルパスとなる回路について解析を行い、デバイス技術が進むほど、実行幅を増やすことがオーバヘッドとなることを示し、クラスタ構成の有効性を議論した。現行プロセッサでは、DEC Alpha 21264[5] が、2 つのサブクラスタで構成されたデータパス構造を持っている。スケジューラもクラスタ化されるような、より積極的なモデルについて、Canal ら[2] は、リネーム時に動的に命令実行クラスタを決定するステアリングロジックを追加することによって、従来とバイナリ互換を持つクラスタ型アーキテクチャを提案した。

クラスタ型アーキテクチャのポテンシャルを活用するためには、ステアリングロジックの最適化が重要である。Parcerisa ら[8),9] は、命令間のレジスタ依存とクラスタ負荷の、双方のバランスを考慮するステアリング方式を提案した。更に服部ら[14] は、より精度の高い "Local Distance" 方式を提案している。

これらの評価ではメモリ参照処理が理想化されていることが多い。クラスタ型アーキテクチャにおけるメモリ参照に関して、Zyuban ら[12] はアドレスによってバンク分割された D0 を持つ方式を提案している。また、Balasubramonian ら[1] は、バンク分割方式と、集中キャッシュ方式をベースラインとして、クラスタ型アーキテクチャの台数効果を評価している。また、Smith ら[10] の "Instruction Level Distributed Proceccing" アーキテクチャでは、複製 D0 方式が採用されている。最近では Gonzalez ら[4] が、いくつかの D0 手法とロード命令のステアリングについて、比較評価を行っている。

13) 及び本論文における我々の研究は、クラスタ型アーキテクチャにおけるメモリ参照のクラスタ内局所化に着目している点、メモリ依存や実行遅延予測等の理想化を行っていない点、キャッシュ方式とメモリフォワーディング方式の比較を行っている点等で新規性があると考えている。

6. 結　　論

本論文では、クラスタ型アーキテクチャにおけるメモリ参照処理に着目し、メモリ参照処理を各クラスタに分散局所化する方式の比較評価を行った。曖昧なメモリ依存関係、実行遅延予測を考慮した場合にも有効な手法として、複製D0による方式と、分散投機メモリフォワーディング方式を取り上げ、比較評価を行った。分散投機メモリフォワーディングは非常に少ない容量で一定の性能向上を得る事ができ、一つ一つのクラスタを小規模に抑える構成に適している。一方、2kbyteを越える容量が利用可能な場合には、複製D0方式は安定した性能を得ることができる。

投機メモリフォワーディング手法はメモリ依存予測を前提とするが、この機構がフロントエンドに新たなクリティカルループを生じさせてしまう可能性がある。依存関係に基づいたステアリングロジック等も同様の問題を抱えており、フロントエンドスループットの現実性は今後検討されなければならない。

また、今回想定したベースライン・アーキテクチャでは、クラスタ間ネットワークの帯域が理想化されている。ネットワークトポロジを考慮した場合のステアリング、スケジューリング、メモリ参照局所化等も今後の課題である。

謝辞　本論文の研究は一部、JST CREST「ディペンダブル情報処理基盤」、21世紀COE「情報技術戦略コア」による。

参 考 文 献

1) R. Balasubramonian, S. Dwarkadas, and D. H. Albonesi. Dynamically Managing the Communication-Parallelism Trade-off in Future Clustered Processors. *30th Int.Symp.on Computer Architecture*, pp. 275–286, 2003.
2) R. Canal, J. M. Parcerisa, and A. Gonzalez. A Cost-Effective Clustered Architecture. *Int.Conf.on Parallel Architectures and Compilation Techniques*, pp. 160–168, 1999.
3) G. Z. Chrysos and J. S. Emer. Memory Dependence Prediction using Store Sets. *25th Int.Symp.on Computer Architecture*, pp. 142–153, 1998.
4) J. Gonzalez, F. Latorre, and A. Gonzalez. Cache Organizations for Clustered Microarchitectures. *3rd Workshop on Memory Performance Issues*, 2004.
5) R. Kessler. The Alpha 21264 Microprocessor. *IEEE Micro*, Vol. 19, No. 2, pp. 24–36, 1999.
6) A.Moshovos and G.Sohi. Speculative Memory Cloaking and Bypassing. *International Journal of Parallel Programming*, 1999.
7) S. Palacharla, N. P. Jouppi, and J. E. Smith. Complexity-Effective Superscalar Processors. *24th Int.Symp.on Computer Architecture*, pp. 1–13, 1997.
8) J. M. Parcerisa and A. Gonzalez. Reducing Wire Delay Penalty through Value Prediction. *33rd Int.Symp.on Microarchitecture*, pp. 317–326, 2000.
9) J. M. Parcerisa, J. Sahuquillo, A. Gonzalez, and J. Duato. Efficient Interconnects for Clustered Microarchitectures. *Int.Conf.on Parallel Architectures and Compilation Techniques*, pp. 291–300, 2002.
10) J. E. Smith. Instruction-Level Distributed Processing. *IEEE Computer*, Vol. 34, No. 4, pp. 59–65, 2001.
11) G. Tyson and T. M. Austin. Improving the accuracy and performance of memory communication through renaming. *30th Int.Symp.on Microarchitecture*, pp. 218–227, 1997.
12) V. Zyuban and P. Kogge. Inherently Lower-Power High-Performance Superscalar Architectures. *IEEE Transactions on Computers*, 2001.
13) 入江英嗣, 服部直也, 高田正法, 坂井修一, 田中英彦. クラスタ型プロセッサのための分散投機メモリフォワーディング. 先進的計算基盤システムシンポジウム 2004, pp. 177–186, 2004.
14) 服部直也, 高田正法, 岡部淳, 入江英嗣, 坂井修一, 田中英彦. 発行時間命令差に基づいた命令ステアリング方式. 先進的計算基盤システムシンポジウム 2004, pp. 167–176, 2004.