クラスタ型データパスによるスーパースカラプロセッサの低消費電力化

佐藤幸紀[†,☆] 鈴木健一[†] 中村維男[†]

スーパースカラ方式プロセッサの広域的なデータパスを複数の局所性を持つ処理要素（PE）に分割してクラスタ化することにより，低消費電力で高並列な処理が可能である．しかしながら，データパスを分割する度合いを増すにつれて，局所化された処理要素間の通信や負荷不均衡により高並列なPEが効果的に活用できないため，結果的にクラスタ化を行わない場合と比べて性能が低下する恐れがある．本論文では，クラスタ型スーパースカラプロセッサの利点である低消費電力性をさらに引き出すために，クラスタ化を進めた場合のIPCの低下を抑える手法を提案する．局所化された処理要素を効率良く利用するために，プログラムに内在する命令の逐次性に着目する．プログラムの逐次性の指標としてプログラム実行時のオペランドの状態とレジスタファンアウトを利用してプログラムの逐次性をクラスタ型プロセッサの局所的処理に対応させる．さらに，局所化された高並列な処理要素を有効利用するために隣接するPEにおいて協調処理を行うことを提案する．実行駆動シミュレーションを実施した結果，隣接PEの協調を行うことにより，高並列な処理要素を効率良く利用しつつ，レジスタファイルの消費電力をクラスタ化を行わない場合と比べて12分の1程度に削減可能であることが分かった．

Designing a Low-power Superscalar Processor Based on Its Clustered Datapath

Yukinori Sato,[☆,†] Ken-ichi Suzuki[†] and Tadao Nakamura[†]

Recently, clustering the complex and centralized datapaths of superscalar processors into localized PEs (clusters) becomes a popular approach to realize low-power and highly-parallelized processors. However, clustering tends to decrease the performance of a processor because of inter-PE communication and workload imbalance among PEs. In this paper, we focus on sequentiality of instructions in programs and apply it to the localized processing on clustered processors. From the analysis of sequentiality using operand status and register fanout, we present that cooperation of neighboring PEs is an effective approach to perform localized processing in a clustered processor. From the results of execution-driven simulations, a clustered processor with the cooperation of neighboring PE achieves better IPC with faster, smaller and lower energy register files than other schemes so far considered. This means that the clustering with the cooperation of neighboring PEs boosts the advantages of clustered processors.

1. はじめに

近年主流であるスーパースカラプロセッサにおいては，利用可能な並列度を増加させるにつれて，レジスタファイルやフォワーディング機構などのデータパスを構成する要素は複雑化し，その回路規模や消費電力は急激に増大することが知られている．したがって，マイクロプロセッサの設計において，命令レベルの並列処理を活用できるようにするだけでなく，消費電力の増大を招くような複雑なハードウェア構造とならないことが求められている．このような背景において，スーパースカラのデータパスに対してクラスタ化を行ったクラスタ型スーパースカラプロセッサが注目を集めている．

従来のスーパースカラ方式の問題とされていた広域的かつ複雑なデータパスは，クラスタ化を行うことにより複数の局所化された小規模なデータパスに分割される．一般的にデータパスを構成する重要な要素であるレジスタファイルや命令発行キューは，マルチポート化されたレジスタセルにより構成されるため，エントリ数やポート数に比例して遅延時間，回路面積，消

† 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
☆ 現在，北陸先端科学技術大学院大学情報科学センター
Presently with Center for Information Science, JAIST

費電力が増大することが知られている．したがって，クラスタ型プロセッサにおいてクラスタ化を行う度合いを増すにつれて，それらの遅延時間，回路面積，消費電力を削減することが可能である．また，クラスタ化によりフォワーディング機構を同一の PE に限定することも，フォワーディング機構の遅延を支配するといわれる配線遅延を減少する点で有効である[1]．

クラスタ型プロセッサにおいてクラスタ化を行う度合いを増すにつれて，低消費電力化に代表されるクラスタ化の利点の恩恵が期待される反面，PE 間の通信や負荷の不均衡が発生するため，クラスタ化を行わなかった場合と比べて IPC が低下すると予想される．そこで，本論文では，データパスをクラスタ化する度合いを増すことにより低消費電力化を図りつつ，プログラムに内在する命令の逐次性をふまえつつ命令列を構築する．さらに，命令列の処理をクラスタ化により生じた局所的な構造における処理へと対応させて処理を行い，クラスタ化された高並列な処理要素を効率良く利用する方式の確立を目指す．

命令の逐次性[2),3)] は，スーパースカラ方式のような命令レベル並列性を利用する方式においては利用されなかったが，クラスタ型プロセッサのようなハードウェア構造における局所性を利用する方式においてはソフトウェアの局所性を見出す重要な概念となる．このような命令の逐次性の概念をクラスタ型スーパースカラ方式の命令ステアリング方式とデータパスの構造に応用することで，クラスタ化による IPC の低下を抑えつつ，データパスの分割で得られた低消費電力化の利点を最大限に活用する手法を提案する．

本論文の構成を以下に示す．2 章では，本論文において想定するクラスタ型スーパースカラプロセッサの構成と実験環境を述べる．3 章では，プログラムに内在する命令の逐次性の傾向について解析を行い，解析の結果得られた知見をクラスタ化されたデータパス上で利用する手法を論じる．4 章では，逐次性をさらに効率良く利用するために，隣接 PE 間の協調動作を提案し，評価する．5 章では，提案する手法の IPC とレジスタファイルの消費エネルギーの比較から，隣接 PE の協調による低消費電力化の効果を見積もる．6 章では関連研究について論じる．7 章は結論である．

2. クラスタ型スーパースカラプロセッサ

2.1 マイクロアーキテクチャの概要

図 1 に本論文で想定するクラスタ型スーパースカラプロセッサを示す．本論文では，クラスタ型プロセッサの構成を X*Y（X：PE 数，Y：PE 内の命令発行幅）と表す．また，クラスタ化を行うデータパスの要素はレジスタファイルとフォワーディング回路とし，それ以外のデータパスの構成はスーパースカラ方式と同じであるとした．クラスタ化により，レジスタファイルは各 PE に 1 つずつ配置され，演算結果のフォワーディングも同一 PE 内のみに限定される．クラスタ化により局所化された各 PE におけるレジスタファイルの構成方式として，各レジスタファイルに個別のレジスタインスタンスが保持される非重複分散レジスタファイル構成を採用した[4),5)]．

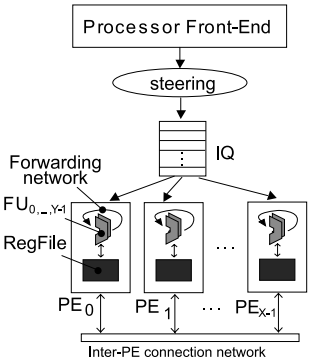

図 1 X*Y 構成のクラスタ型スーパースカラプロセッサ
Fig. 1 A clustered superscalar processor with X*Y configuration.

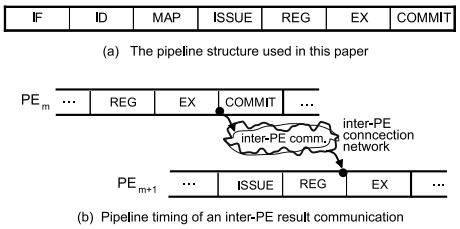

図 2 パイプラインの構造
Fig. 2 The timing of the pipeline.

図 2 (a) に Alpha21264 のパイプラインの構成をベースとした本論文で想定するパイプライン構成を示す．クラスタ化に対応するために，MAP ステージにおいて命令ステアリング機構がそれぞれの命令に対して実行する PE の割付けを行うと同時に，演算結果の出力先のレジスタを割り付けると変更した．命令は MAP ステージにおいてステアリングされた後，命令キュー（IQ）に格納され，ISSUE ステージにてオペランドが利用可能であれば命令は wakeup される．wakeup された命令は対応する PE の演算資源が利用可能であれば select され，REG ステージにおいてレジスタが読み込まれた後，EX ステージにおいて実行される．

演算結果のフォワーディング回路のクラスタ化によ

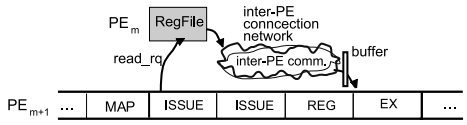

図 3 他 PE からのレジスタの読み込み
Fig. 3 The pipeline timing at inter-PE register read.

表 1 主要なアーキテクチャパラメタ
Table 1 Dominant architectural parameters.

Fetch and decode	8 instructions
Branch predictor	Tournament branch predictor
IQ, FQ, LQ, SQ size	64
ROB size	256
Issue width	8 issue in total
L1Icache	128 KB, 2way
L1Dcache	128 KB, 2way

り，オペランドを生成する PE とそのオペランドを利用する PE が同一が否かによって，未解決オペランドが利用可能となった際の挙動が異なる．同一 PE 内の演算器により入力オペランドを生成する場合，オペランドはフォワーディング回路により転送されるため，連続して処理を進行することができる．しかし，入力オペランドが異なる PE の演算器から転送されてくる場合は，図 2 (b) に示すようにオペランドの転送に 2 サイクル余分にかかると想定した．

非重複分散レジスタファイル方式の場合，すでに入力オペランドは利用可能な状態であるが，割り付けられた PE 内のレジスタファイルに存在しないことがおこる．この場合，他 PE のレジスタファイルより転送を行う必要がある．本論文では，この転送にかかる遅延を 1 サイクルと想定した．**図 3** は他 PE からのレジスタ読み込みのための PE 間通信のタイミングを示す．

2.2 実験条件

プログラムの性質やクラスタ型スーパースカラプロセッサを評価する実験は，SimpleScalarV4 ツールセットの sim-alpha[6] をベースとするサイクル精度の実行駆動型シミュレータを用いて行った．本シミュレータの主要なアーキテクチャパラメータを**表 1** に示す．残りの構成や，キャッシュ，機能ユニットの遅延は alpha21264 と同様とした．

MediaBench より 4 つのベンチマーク（djpeg, cjpeg, rawdaudio, rawcaudio），SPEC2000CPUint より 7 つのベンチマーク（gzip, vpr, gcc, mcf, perlbmk, bzip, twolf）を選び実験を行った．すべてのベンチマークは，Compaq C compiler v6.5 により-O4 -fast -non_shared オプションを用いてコンパイルした．MediaBench の各プログラムは終了するまで命令の実行を行った．SPEC2000int の各プログラムは 1 G 命令フォワードした後の 100 M 命令の実行を行った．

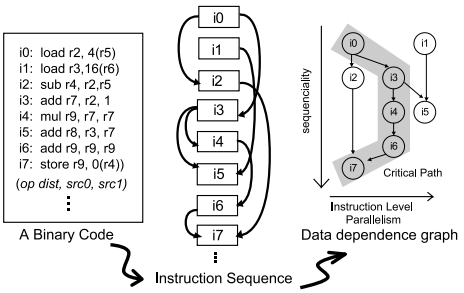

図 4 命令列における並列性と逐次性の関係
Fig. 4 Parallel and serial property in an instruction sequence.

3. 命令の逐次性とクラスタ型スーパースカラ

クラスタ型スーパースカラプロセッサは，クラスタ化の恩恵によりそのデータパスを低消費電力化する反面，PE 間の通信や負荷不均衡により得られる IPC が低下することが知られている[5]．そこで，本質的に不必要な PE 間の通信や負荷不均衡を回避しつつ処理を行うことにより IPC の低下を抑えることが必要である．IPC の低下を抑えるためには，プログラムから局所性を見出し，クラスタ型プロセッサの局所化されたハードウェア構造に対応付けて命令を局所的に処理することが有用であると考えられる．本論文では，プログラムの局所性を抽出する尺度として，命令の逐次性[2),3)] に着目する．

プログラムにおける逐次性を定量的に評価するために，レジスタを媒介としたデータ依存関係のある 2 つの命令が実行される時間差を距離ととらえ，この距離に着目する．プログラムが 1 命令ずつ逐次的に実行されていくと仮定して，データ依存関係のある 2 つの命令間の距離をその間に実行された命令数と定義する[2)]．たとえば，**図 4** において，命令 i2 は命令 i0 に依存しており，距離は 2 である．

スーパースカラプロセッサにおいては，命令間の距離が短い依存関係はフォワーディング機構を用いることで，演算結果を次のサイクルに依存先の命令の入力として利用できる．オペランドの大部分は生成された直後に使用されるという報告[2),7)] や，SPEC2000cpu ベンチマークにおける平均 66%の命令が演算結果のフォワーディングを利用してオペランドを得るという報告[8)] から，命令間の距離が短い依存関係を持つ命令をフォワーディング機構を用いて処理することは頻繁に行われることが分かる．一方で，演算結果のフォ

ワーディングに必要となるサイクル数が増加した場合，すなわち演算結果が次のサイクルで利用できなくなった場合，性能が大きく低下することが報告されている[9),10)]．これより，命令間の距離が短いデータ依存により構成される命令列は，データフローグラフ上のクリティカルパスとなる傾向が強いと推定される．

スーパースカラ方式において並列度を増加させるにつれて，命令間の距離が短い依存関係を実行するうえで鍵となるフォワーディング機構の遅延は著しく増加し，プロセッサの動作周波数を低下させる原因となる．クラスタ型スーパースカラプロセッサは，フォワーディング機構についてクラスタ化を行うことにより，この遅延の増加の問題を解決している[1)]．一方で，クラスタ化した場合，命令ステアリングによっては命令間の距離が短い依存関係の間にも PE 間通信やリソース競合が発生する可能性がある．たとえば，図 4 のデータフローグラフにおいて灰色で示される処理のクリティカルパス上に遅延が挿入された場合，IPC は低下する．

以上をまとめると，クラスタ型プロセッサにおいて命令間の距離が短い依存関係をどのように実行するかにより，得られる IPC は大きく影響を受ける．そこで，クラスタ型プロセッサにおいて命令ステアリングを行う際の依存のある命令間の距離の傾向を調べるために，依存のある命令間の距離とオペランドの状態の関係について調査を行った．オペランドの状態は，アウトオブオーダ実行のためにつねに監視されいる状態であり，オペランドをフェッチする際においてオペランドが利用可能（ready）か利用可能でない（unready）かのいずれかである．命令が IQ にディスパッチされた際に依存のあるオペランドがすでに利用可能である場合は即座に実行可能として wakeup される．図 4 のデータ依存グラフにおいて，命令 i3 まで実行が完了しているとすると，命令 i4，i5 の入力オペランドは ready 状態であり，命令 i6 の入力オペランドは unready 状態である．

表 2 に依存のある命令間の距離の分布を調べた結果を示す．データ依存関係は，レジスタを介した命令間に真依存（Read After Write dependencies）がある場合のみを計測し，メモリを介した依存は検出が困難なため除外した．表中の値は，データ依存関係がある命令に対して命令間の距離と先行するオペランドの状態を調査し，命令間距離ごとに集計した値を全体の依存関係の数で正規化したものである．結果より，unready のオペランドは依存がある命令との距離が短く，ready のオペランドは依存のある命令との距離が長い傾向があることが分かる．したがって，依存のある命令間の距離はオペランドとの状態と強い相関関係があることが推測される．

表 2 オペランドの状態ごとの依存のある命令間距離の分布（SPEC2000cint 平均）
Table 2 The distribution of distance between dependent instructions based on their operand statuses (SPEC2000cint avg.).

distance	1	2	3	4	5
ready	0.0%	0.0%	0.1%	0.1%	0.2%
unready	15.2%	11.6%	7.9%	6.1%	3.6%
distance	6	7	8	9	10
ready	0.3%	0.5%	0.3%	0.2%	0.1%
unready	3.2%	3.1%	2.3%	2.2%	1.7%
distance	11	12	13	14	15+
ready	0.5%	0.3%	0.5%	0.2%	27.9%
unready	1.5%	1.1%	1.2%	0.9%	7.2%

クラスタ型スーパースカラプロセッサにおいて，命令間の距離が短いデータ依存のある命令列を効率良く処理する方式として，!ready（not ready）命令ステアリング方式が提案されている[11)]．!ready 方式は，表 2 に示したように命令間の距離が短いデータ依存関係のある命令はオペランドの状態が未解決である可能性が高いことを利用する．命令間の距離を直接カウントする代わりにオペランドの状態を尺度とするのは，依存のある命令間の距離をつねに観測することは距離を保持する巨大なテーブルを必要とし現実的でないためである．

!ready 方式は未解決オペランドを持つ命令をクリティカルパス上に存在する命令として優先的に PE 間通信やリソース競合を起こさないようにステアリングする．具体的には，未解決なオペランドを少なくとも 1 つ持つ命令をそのオペランドが生成される PE に割り付け，非クリティカルな命令である解決済みオペランドからなる命令は負荷最小の PE に割り付ける．負荷最小の PE を推定するために DCOUNT 指標[12)] を用いた．DCOUNT は，各 PE にディスパッチされた命令の個数をカウンタを用いて監視し，各 PE の負荷を推測するための指標である．

比較対象とする命令ステアリング方式として RMBS（Register Mapping Based Steering），Advanced_RMBS の 2 つを取り上げる[13)]．RMBS は Palacharla らが提案したデータ依存に基づく方式[1)] と同様の方式であり，ある命令のオペランドに依存がある場合，必ず依存のある先行命令が割り付けられている PE のいずれかに命令を割り付ける．Advanced_RMBS[12)] は，負荷分散を追加した依存関係に基づく方式であり，RMBS 方式の問題点であった負荷集中による IPC 低下を回避するため，RMBS 方

表 3 オペランドの状態に基づく命令ステアリング方式と正規化 IPC
Table 3 Instruction steering based on operand statuses and the normalized IPC.

	MediaBench avg.	SPEC2000 avg.
RMBS	1.00	1.00
Advanced_RMBS	1.01	1.02
!ready	1.13	1.08

式に各 PE の DCOUNT 指標が閾値を超えた場合に負荷の集中が起こったと見なして負荷が最小の PE に命令を割り付けるという動作を追加した方式である．

表 3 は 8*1 構成における命令ステアリング方式ごとの IPC を RMBS の IPC で正規化した結果である．ここで，各 PE は 40 本のレジスタを持つとした．実験結果より，!ready 方式は RMBS 方式や advanced RMBS 方式より IPC が 1 割ほど高いことが分かる．これより，データ依存関係や負荷分散といった項目のみで命令をステアリングすることは，プログラムの処理時間を決定するクリティカルパスの遅延を増大させる可能性が高いといえる．また，命令の逐次性を考慮してオペランドの状態を指標とすることで高い IPC を得られるといえる．以上から，命令の逐次性はクラスタ型スーパースカラプロセッサにおいてクラスタ化の度合いを増した場合にも IPC の低下を防ぐための重要な概念であり，オペランドの状態は命令の逐次性を定量的に測る 1 つの指標であるといえる．

4. 隣接 PE の協調

クラスタ型スーパースカラプロセッサにおいて低消費電力化と高並列な処理を両立させるためには命令の逐次性は重要な概念である．命令の逐次性を定量的に測る 1 つの指標としてオペランドの状態に着目したが，オペランドの状態が命令の逐次性をどの程度的確に表しているかを詳細に調べる必要がある．そこで，命令が IQ にディスパッチされる際の命令に対するオペランドの状態の分布を調査した．その結果を**表 4** に示す．1 つの命令は入力として 2 つまでオペランドをとりうるため，オペランドの状態とあわせて 9 通りの場合に分類される．表中の null は入力オペランドが不要である場合を示す．

表 4 より，少なくとも 1 つの未解決オペランドを持つ命令は 75%を占め，逆にすべてのオペランドが利用可能にある命令は 25%を占めることが分かった．オペランドの状態の分布において，依然として残りの 75%を占める未解決オペランドを持つ命令どうしが PE 内においてリソース競合を起こしていることが推測される．たとえば，図 4 において命令 i3 の結果が未解決である場合，!ready 方式では i4 と i5 は同一 PE に割り付けられる．しかしながら，クラスタ化により PE 内で同時実行可能な命令数は制限されるため，リソース競合のため i5 の実行が遅れてしまう場合がある．

表 4 命令に対するオペランドの状態（SPEC2000cint 平均）
Table 4 The distribution of operand statuses (SPEC2000cint avg.).

	null	ready	!ready
null	2.1%	12.2%	24.7%
ready	5.3%	5.8%	6.8%
!ready	17.9%	10.2%	14.9%

表 5 レジスタファンアウトの分布（SPEC2000cint 平均）
Table 5 The distribution of register fanouts (SPEC2000cint avg.).

Register fanout	0	1	2	3	4	5+
Percentage[%]	5.8	66.3	15.8	3.5	2.0	6.6

そこで，命令間の依存関係の広がりを表すレジスタファンアウトに着目する[14]．ここで，レジスタファンアウトを「あるレジスタに値が書き込まれてから，そのレジスタに次の書き込みがあるまでの読み出しの回数」と定義する．図 4 のデータ依存グラフにおいては，命令 i2 は命令 i0 に依存しており，命令 i3 のレジスタファンアウトは 2 である．

表 5 にレジスタファンアウトの分布を調べた結果を示す．レジスタファンアウトが 1 である命令は最も多く，平均 65%となった．レジスタファンアウトが 2 である命令は平均 15%を占め，また，レジスタファンアウトが 0 となる命令を含めるとレジスタファンアウトが 2 以下の命令は全体の 9 割を占めることが分かった．

クラスタ化を行うことにより，レジスタファンアウトが 1 を超える場合の後続命令において PE 間通信やリソース競合が発生しやすくなる．たとえば，8*1 構成のクラスタ型プロセッサの場合，PE 内での同時実行可能な命令数は 1 であるため，レジスタファンアウトが 2 の後続命令を同一 PE に割り付けるとリソース競合が発生する．表 5 のようにレジスタファンアウトが 2 以上のものは全体の 30%を占めるため，レジスタファンアウトが 2 以上の場合も並列実行を可能にすることが望まれる．

そこで，同一の未解決オペランドを使用する命令どうしのリソース競合を防ぐことを目的として，隣接する PE が協調して処理を行う方式を提案する．この隣接 PE の協調は，隣接 PE 間に 1 方向の局所的な通信経路を追加することと，隣接 PE の協調を支援する

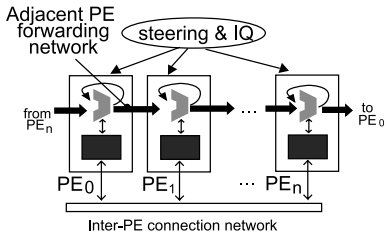

図 5 Adjacent PE Forwarding（AF）構成
Fig. 5 Adjacent PE Forwarding (AF) configuration.

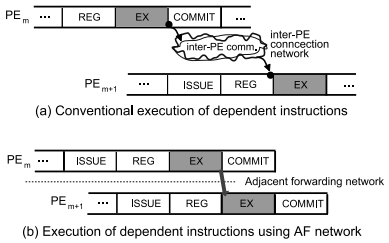

図 6 依存のある命令の実行の様子
Fig. 6 Execution of dependent instructions.

ように命令をステアリングすることからなる．1 方向の隣接 PE 間の通信経路とした理由は，ハードウェア的なコストの面とプログラムにおける命令列の特性の傾向をふまえたことによる．隣接 PE 間に 1 方向の局所的な通信経路を追加するコストは，隣接 PE 間の物理的な距離が配線遅延が問題にならない程度に短く複雑度も低いことから，十分に小さいと考えられる．また，レジスタファンアウトが 2 以下の命令が 90%を占めることから，レジスタファンアウトが 2 である命令の転送を想定した 1 方向の通信経路で十分と考えられる．

図 5 は隣接 PE 間の協調を行うための構成（AF: adjacent PE Forwarding）を示す．各 PE における演算の結果を隣接 PE に直接バイパスする通信経路がすべての隣接 PE 間に追加され，全体としてループを形成する．**図 6** は，依存のある命令が隣接する 2 つの PE で実行される様子を示す．図 6 (a) の従来の構成においては依存のある命令の実行の間に PE 間通信による遅延が発生してしまうが，図 6 (b) の AF 構成においては依存のある 2 つの命令が連続するサイクルで実行できる．

隣接 PE 間のフォワーディング経路を有効に活用するために，レジスタファンアウトの数に基づく命令ステアリング手法を提案する．この命令ステアリング手法はレジスタファンアウトが 1 より大きい場合の後続命令を並列に実行することを目的とする．レジスタファンアウトは Butts ら[15] のように過去の履歴など

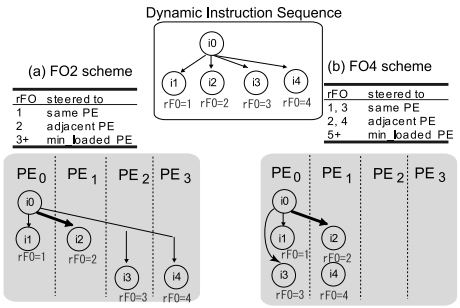

図 7 レジスタファンアウトに基づく命令ステアリング
Fig. 7 Instruction steering based on the register fanout.

から予測するのではなく，レジスタに値が書き込まれてからそのレジスタを読んだ回数をカウントすることにより得られるレジスタ使用回数（動的なレジスタファンアウト）を指標とする．提案するレジスタファンアウトの数に基づく命令ステアリング手法は以下のように実現する．未解決オペランドを持つ命令は動的レジスタファンアウトの値により，未解決オペランドを生成する PE あるいはその右側に隣接する PE に命令を割り付ける．未解決オペランドをまったく持たない命令は負荷最小の PE に割り付ける．

図 7 にレジスタファンアウトの数に基づく命令ステアリング手法の詳細を示す．動的レジスタファンアウトの閾値により様々な構成が可能となるが，特にレジスタファンアウトが 2 である命令のみに着目した FO2 方式と，PE 間通信のロスを最小限とするために動的レジスタファンアウトが 4 までの命令を同一 PE あるいは隣接 PE に割り付ける FO4 方式の 2 つを取り上げる．図 7 では，命令 i1，i2，i3，i4 は命令 i0 の結果を利用し，命令 i0 は実行されていないため結果は未解決である場合を想定している．このとき，命令 i1，i2，i3，i4 の動的レジスタファンアウトはそれぞれ 1，2，3，4 である．

図 7 (a) FO2 方式は動的レジスタファンアウトが 2 である命令を隣接 PE にステアリングするため，命令 i2 は隣接する PE に割り付け，命令 i3 や i4 は負荷最小の PE に割り付けられる．図 7 (b) FO4 方式は動的レジスタファンアウトが 2 または 4 である命令の場合，その命令を隣接 PE にステアリングするため，命令 i2，i4 は隣接する PE に，命令 i3 は命令 i0 と同一の PE に割り付けられる．FO2 方式は FO4 方式より隣接する PE におけるリソース競合を抑制することが見込まれるが，FO4 方式より多くの PE 間通信によるロスを発生させる．

Abella らも我々のアプローチのように隣接する PE をリング型に接続し，隣接する PE を利用して負荷分

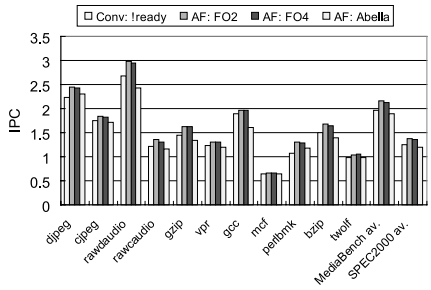

図 8 隣接 PE 間の協調の効果
Fig. 8 The effects of cooperation of adjacent PEs.

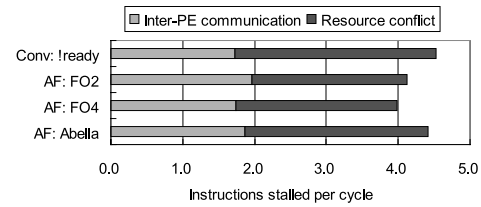

図 9 1 サイクルあたりの平均命令ストール数（MediaBench 平均）
Fig. 9 The number of instructions stalled per cycle (MediaBench avg.).

散を行う命令ステアリングを提案している[16]．しかしながら，Abella らはデータ依存関係がある場合，演算結果をつねに隣接する PE に書き込むとしている．すなわち，オペランドの状態やレジスタファンアウトによらずつねに隣接 PE に割り付けるという命令ステアリングである．したがって，負荷分散が我々の提案する命令ステアリング手法と比べてうまくいかないと予想される．

提案する隣接 PE の協調の効果を確かめるために評価実験を行った．**図 8** は，8*1 構成において隣接 PE 間の協調を評価した結果である．各 PE のレジスタ数を 32 とし，隣接 PE 間の協調の効果を的確に把握するためにバンド幅が 1 の PE 間通信共有バスをモデル化した．図中の conv は隣接 PE のフォワーディング機構を持たない構成を示し，AF は隣接 PE 間にフォワーディング機構を追加した構成を示す．また，各構成の後ろに利用した命令ステアリング方式が示されている．隣接 PE の協調を行う構成について命令ステアリング手法は FO2，FO4，Abella の 3 種類を用いた．

隣接 PE の協調を行うために通信経路を追加し，FO2 あるいは FO4 命令ステアリングを行うことにより，従来の構成と比べて 1 割ほど IPC を向上させることが可能である．FO2 ステアリングと FO4 ステアリングを比べた場合，FO2 ステアリングの方が高い IPC を達成できることが分かる．これは，隣接する PE に多くの命令が割り付けられると，隣接する PE における本来の処理のために必要なリソースを奪ってしまい，IPC が低下してしまうためと思われる．

Abella らの命令ステアリングは，conv 構成と比べても IPC が低いことがある．Abella らの行った実験においても SPEC2000INT ベンチマークを 8*1AF 構成において実行した IPC は conv 構成よりも IPC が低く，SPEC2000FP プログラムを実行した場合や 8*2AF 構成とした場合は conv 構成より高い IPC を達成すると報告している[16]．したがって，Abella らの手法は依存関係の制約が緩い，あるいは，PE 内の同時実行可能な命令発行幅が広いという理由により負荷分散がうまくいく場合に限り高い性能を発揮すると推測される．

隣接 PE の協調の効果を議論するために，クラスタ化により発生したストールの原因を調べた．**図 9** に 1 サイクルあたりの命令ストール数の平均を示す．結果より，AF 構造は従来の構造と比べてリソース競合のためのストールを減らしていることが分かる．また，AF 構造においては FO2 命令ステアリング方式は隣接する PE におけるリソース競合を，FO4 方式は PE 間通信のためのストールをより減らしていることが分かる．しかしながら，FO2 方式の方が高い IPC となったのは，隣接 PE のリソース競合の方が実行のクリティカルパスを延ばす傾向があるからと思われる．また，Abella らの命令ステアリングは隣接 PE においてリソース競合が多発しているため負荷分散がうまくいっていないということが確認できる．

以上より，命令の逐次性を考慮して隣接 PE の協調を行うことは，クラスタ型スーパースカラプロセッサにおいてクラスタ化の度合いを増した場合の IPC の低下を抑制する有効な手法であることが確認された．

5. クラスタ化の効果と IPC

データパスをクラスタ化する度合いを増していった場合や隣接 PE 間の協調を行うことにより最終的に期待できる IPC や消費電力の削減率は設計段階で把握しておくべき重要な項目である．そこで，隣接 PE 間にフォワーディング経路を追加するコストの見積りと，クラスタ型プロセッサにおいて隣接 PE の協調を行った場合のレジスタファイルの評価をレジスタファイルの消費電力や面積の観点から行った．

隣接 PE 間の協調を行うためには，隣接 PE 間に結果のフォワーディング経路を追加する必要がある．この隣接 PE 間のフォワーディング経路を追加することによりプロセッサの動作速度が低下する恐れがあるが，我々は Abella ら[16] が示すようにクラスタ型プロセッ

表 6 クラスタ化の効果と IPC
Table 6 The effects of clustering and the IPC obtained.

	8*1AF	8*1	4*2	1*8
Number of registers per PE	32	32	48	128
Number ofports of reg	3R1W	3R1W	6R2W	16R8W
Number of results forwarded to FUs	2	1	2	8
Operating frequency of RegFiles [GHz]	4.51	4.51	3.71	2.08
Normalized freq. of RegFiles	2.17	2.17	1.78	1
Area of RegFiles [λ]	6.26E+07	6.26E+07	1.47E+08	7.81E+08
Normalized Area of RegFiles	0.08	0.08	0.19	1
Energy for a register access [J/access]	2.87E-10	2.87E-10	7.09E-10	3.60E-09
Normalized energy for reg.	0.08	0.08	0.2	1
IPC (MediaBench avg.)	2.16	1.97	2.29	2.67
Normalized IPC	0.81	0.74	0.86	1
Peak IPC (rawdaudio)	2.98	2.67	2.97	3.59
Normalized peak IPC	0.83	0.74	0.83	1

サにおける PE 間のフォワーディング経路の遅延はプロセッサの動作周波数を決定するうえでクリティカルパスとならないと考える．Abella ら[16] は，隣接 PE 間のフォワーディング経路の遅延が性能低下に結び付かないということを裏付けるため，機能ユニットレベルで実際にレイアウトを行い，隣接 PE 間のフォワーディング経路に必要なレイアウト上の距離は同一 PE 内でのフォワーディングに必要な距離と同程度であると見積もった．フォワーディング機構における遅延は結果を転送するバスの配線遅延が支配的となるといわれていることをふまえると[1]，フォワーディング機構の遅延はフォワーディング経路のレイアウト上の距離に比例するといえる．したがって，隣接 PE 間の結果のフォワーディング経路は十分に局所的であるため，性能低下に結び付かないといえる．

隣接 PE 間のフォワーディング経路の遅延がプロセッサの動作速度低下を引き起こさないのであれば，隣接 PE 間の協調による IPC の向上はプロセッサ全体の処理能力を向上させるといえる．一方で，隣接 PE 間にフォワーディング回路を追加することにより，クラスタ化の度合いを低下している場合と非常に近い構成となる．たとえば 8*1 構成において隣接 PE の協調を行うと，4*2 構成に近い構成となる．両者の構成を分ける要素はレジスタファイルの構成である．これは，1 つの PE が独立して 1 つのレジスタファイルを備えていると想定しているためである．クラスタ化を行うほど，レジスタファイルも分散され，小面積で低消費電力かつ高速な動作が期待される半面，通信による性能低下が予想される．

クラスタ型プロセッサにおいて，レジスタファイルを分散することによる影響を調べるために，レジスタファイルの動作速度，消費エネルギー，ならびに必要となる面積の見積りを行った．レジスタファイルの動作周波数と消費エネルギーは CACTI-2.0 モデル[17] を用いて見積もった．半導体の設計ルールは 0.07 μm を想定した．また，レジスタファイルの面積は各 PE のレジスタファイルに必要なポート数とレジスタ数を入力として Lee の手法[18),19)] により算出した．ここで，$2\lambda = 0.07\,\mu$m と等しいとして計算した．

クラスタ型プロセッサの構成，レジスタファイルの動作速度，消費エネルギー，および面積の関係を**表 6** に示す．各構成の命令ステアリング手法は，8*1AF 構造は FO2，そのほかは!ready 手法とした．X*Y 構成の各 PE のレジスタファイルにおいて，必要なポート数は PE 内の FU のための 2Y リード Y ライトポートと PE 間通信の共有バスのための Y リードポートとした．各構成のレジスタファイルのレジスタ数については，クラスタ化を行わない構成の 128 を基準とし，クラスタ化を行った構成については 128 を PE の個数分に分割し，PE ごとの利用可能なレジスタ数の不均衡を是正するために分割されたレジスタファイルに 16 個を加えるとした．8*1AF 構成のレジスタファイルの構成は 8*1 構成と同一となる．また，8*1AF 構成は 4*2 構成とフォワーディング経路の個数は同一であるが，レジスタファイルが異なる．

表 6 にクラスタ型プロセッサの構成ごとにレジスタファイルと IPC の評価を行った結果を示す．レジスタファイルの動作周波数は，高度にクラスタ化を行うほど高速化が可能ということが分かる．8-way のスーパースカラプロセッサを 8 個の PE にクラスタ化した場合にレジスタファイルの動作周波数は，クラスタ化を行わない構成（1*8）と比べて 2.17 倍，4 個の PE にクラスタ化した場合には 1.78 倍となる．また，クラスタ型プロセッサ全体のレジスタファイルの面積やレ

ジスタアクセスのために必要なエネルギーは，高度にクラスタ化を行うほど大幅に減少することが分かる．面積と消費エネルギーはほぼ同等の傾向を示し，8 個の PE にクラスタ化した場合はクラスタ化を行わない場合と比べて 12 分の 1 程度に，4 個の PE にクラスタ化した場合は 5 分の 1 程度となる．

同時発行可能な命令数が増加するにつれて，レジスタファイルの消費電力の増大がプロセッサを実装するうえでの大きな制約となる．Zyuban らは[20]，8 命令同時発行可能なスーパースカラプロセッサにおいてレジスタファイルの消費電力はキャッシュを除くコア全体で約 4 割を占めることを示している．レジスタファイルのクラスタ化を行うことによりデータパスにおける消費電力の 4 割分を 12 分の 1 に削減できるとすると，データパスの消費電力は元の 6 割程度となる．したがって，クラスタ化によりレジスタファイルのアクセスごとの消費エネルギーを大幅に削減することは，プロセッサの低消費電力化に大きく貢献するといえる．

クラスタ化によりレジスタファイルを分割することにより IPC は低下する．IPC について，MediaBench の平均を比較したところ，8*1 構成は 1*8 構成から 26%ほど IPC が低下したが，8*1AF 構造においては 1*8 構成より 19%ほどの低下に抑えられることが分かる．一方，4*2 構成は 1*8 構成から 14%ほど IPC が低下している．

クラスタ化された構成において，どの程度の IPC が期待できるかを調べるために，ベンチマークを通しての IPC の最大値であるピーク IPC を比較した（表 6 下段）．クラスタ化を行っていない 8-way 構成のスーパースカラである 1*8 構成のピーク IPC は 3.59 であり，8*1AF 構成は 1*8 構成と比べて 17%だけピーク IPC が低下している．8*1AF 構成は 4*2 構成のピーク IPC をわずかだが凌駕している．この結果は，クラスタ化の度合いを増加させても，隣接間 PE の協調により効率良く処理を行えば性能低下を回避可能であることを示唆している．本実験においては，コンパイラの出力するバイナリコードをそのまま入力としたが，アセンブリレベルの最適化を行うことにより，隣接する PE の協調を支援することによりさらに高い IPC を達成できると考えられる．

以上より，8*1AF 構成は 4*2 構成と比べてレジスタファイルの消費電力を 2 分の 1 以上削減しつつ，同等レベルの IPC を達成するといえる．したがって，隣接間 PE の協調は，クラスタ化による低消費電力化という利点を最大限に活用すると同時に，高い IPC を達成する方式であるといえる．また，クラスタ化によりレジスタファイルの動作周波数の向上や面積の削減といった効果を期待できる点もクラスタ化の有効性をいっそう引き立たせる方式であるといえる．

6. 関 連 研 究

本研究においてベースとした未解決オペランドを持つ命令をデータ依存に基づきステアリングを行う!ready 方式をさらに拡張した命令ステアリング方式として，依存のある命令間の発行時間差に基づき命令ステアリングを行う方式（ldist）がある[21]．ldist 方式は，命令キューにおける発行時間差を算出するために，ステアリングされた命令に対して PE ごとに通し番号を付ける．この番号の差を Local distance（ldist）と呼ぶ．未解決のオペランドとその依存する先行命令の間の ldist が閾値を超えない場合，RMBS 方式と同様のステアリングを行う．閾値を超えた場合は，その命令は PE 間通信を挿入しても IPC には影響がないと判断され，負荷が最小の PE に割り付けられる．負荷の状況は PE 内の未発行命令数を用いて監視される．!ready 方式は，ldist 方式の距離の閾値を無限大として未解決オペランドの場合はつねに同一 PE に割り付けるとした場合と同等である．また，ldist 方式は，!ready 方式と比べて番号の生成や記憶，距離の算出，比較にハードウェアを追加する必要がある．

本論文で提案しているレジスタファンアウトに基づく命令ステアリングと服部らの ldist 方式[21] はある動的な指標に対して特定の閾値を与え命令をステアリングする点においては共通である．しかしながら，ldist 方式は隣接する PE に命令をステアリングする基準を設定するのが困難であるのに対し，我々の提案手法は命令の逐次性を意識しつつ隣接 PE に命令をステアリングを行うことが可能である．必要となるハードウェアの違いとしては，動的レジスタファンアウトを観測するためにはレジスタごとに使用された回数をカウントするカウンタを必要とする一方で，服部らの ldist 方式は発行時間差を算出するために命令キューの中のエントリ間の距離を測るハードウェアを必要とする．

服部らはクリティカルパス情報を過去の履歴から推測し，クラスタ型スーパースカラプロセッサの命令ステアリングに応用するという報告も行っている[22]．しかしながら，クリティカルパス情報だけでは不十分であり負荷情報を組み合わせることが重要と述べているので，適切な命令のステアリングの基準が重要であると考えられる．本論文においては隣接 PE 間の協調動作を行うためにオペランドの状態とレジスタファンアウトという指標を用いたが，ldist のような指標やク

リティカルパス情報から適切な指標を見つけ出し隣接PE間の協調動作を行うことも考えられる．命令ステアリングのための指標を得るためのハードウェアコストを含めて適切な命令ステアリング手法を確立することは今後の課題である．

7. 結　　論

本論文では，クラスタ型スーパースカラプロセッサの利点である低消費電力性をさらに引き出すために，プログラムに内在する逐次性に着目し，クラスタ化されたデータパス上において効率的に実行することを試みた．プログラム中の逐次性の指標として命令実行時にオペランドの状態とレジスタファンアウトを観測し，隣接するPEにおいて協調を行うことを提案した．評価実験の結果，隣接PE間の協調を行うことにより，従来手法よりIPCの低下を抑えつつ，レジスタファイルの消費電力を大幅に削減できることが分かった．

参 考 文 献

1) Palacharla, S., Jouppi, N.P. and Smith, J.E.: Complexity-effective superscalar processors, *Proc. 24th annual international symposium on Computer architecture*, pp.206–218 (1997).
2) 佐藤幸紀，鈴木健一，中村維男：プログラムにおける命令の並列性と逐次性について，情報処理学会研究会報告 2004-ARC-157，pp.121–132 (2004).
3) 佐藤寿倫：命令レベル逐次プロセッサ，情報処理学会研究報告 2006-ARC-169，pp.49–54 (2006).
4) Llosa, J., Valero, M. and Ayguade, E.: Non-Consistent Dual Register Files to Reduce Register Pressure, *Proc. 1st IEEE Symposium on High-Performance Computer Architecture*, pp.22–31 (1995).
5) Sato, Y., Suzuki, K. and Nakamura, T.: Power Estimation of Partitioned Register Files in a Clustered Architecture with Performance Evaluation, *IEICE Trans. Information and Systems*, Vol.E90-D, No.3, pp.627–636 (2007).
6) Desikan, R., Burger, D. and Keckler, S.W.: Measuring Experimental Error in Microprocessor Simulation, *Proc. 28th annual international symposium on Computer architecture*, pp.266–277 (2001).
7) Franklin, M. and Sohi, G.S.: Register traffic analysis for streamlining inter-operation communication in fine-grain parallel processors, *Proc. 25th annual international symposium on Microarchitecture*, pp.236–245 (1992).
8) Park, I., Powell, M.D. and Vijaykumar, T.N.: Reducing register ports for higher speed and lower energy, *Proc. 35th annual ACM/IEEE international symposium on Microarchitecture*, pp.171–182 (2002).
9) Hrishikesh, M.S., Burger, D., Jouppi, N.P., Keckler, S.W., Farkas, K.I. and Shivakumar, P.: The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays, *Proc. 29th annual international symposium on Computer architecture*, pp.14–24 (2002).
10) Sprangle, E. and Carmean, D.: Increasing processor performance by implementing deeper pipelines, *Proc. 29th annual international symposium on Computer architecture*, pp.25–34 (2002).
11) Sato, Y., Suzuki, K. and Nakamura, T.: An operand status based instruction steering scheme for clustered architectures, *Proc. 2005 International Conference on Computer Design* (*CDES'05*), pp.168–174 (2005).
12) Parcerisa, J.-M. and González, A.: Reducing wire delay penalty through value prediction, *Proc. 33rd annual international symposium on Microarchitecture*, pp.317–326 (2000).
13) Canal, R., Parcerisa, J.-M. and González, A.: A Cost-Effective Clustered Architecture, *Proc. 1999 International Conference on Parallel Architectures and Compilation Techniques*, pp.160–168 (1999).
14) Sankaralingam, K., Nagarajan, R., Keckler, S. and Burger, D.: A Technology-Scalable Architecture for Fast Clocks and High ILP, *5th Workshop on the Interaction Between Compilers and Computer Architectures* (2001).
15) Butts, J.A. and Sohi, G.S.: Characterizing and predicting value degree of use, *Proc. 35th annual ACM/IEEE international symposium on Microarchitecture*, pp.15–26 (2002).
16) Abella, J. and Gonzalez, A.: Inherently Workload-Balanced Clustered Microarchitecture, *Proc. 19th IEEE International Parallel and Distributed Processing Symposium*, p.20a (2005).
17) Reinman, G. and Jouppi, N.P.: CACTI 2.0: An Integrated Cache Timing and Power Model, Technical report, WRL Research Report 2000/7 (2000).
18) Lee, C.: Code Optimizers and Register Organizations for Vector Architectures, Ph.D. Thesis, Univ. of California at Berkeley (1992).
19) Lopez, D., Llosa, J., Valero, M. and Ayguade, E.: Cost-Conscious Strategies to Increase Performance of Numerical Programs on Aggressive VLIW Architectures, *IEEE Trans. Comput.*, Vol.50, pp.1033–1051 (2001).

20) Zyuban, V.V. and Kogge, P.M.: Inherently Lower-Power High-Performance Superscalar Architectures, *IEEE Trans. Comput.*, Vol.50, No.3, pp.268–285 (2001).
21) 服部直也，高田正法，岡部　淳，入江英嗣，坂井修一，田中英彦：発行時間差に基づいた命令ステアリング方式，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG11, pp.80–93 (2004).
22) 服部直也，高田正法，岡部　淳，入江英嗣，坂井修一，田中英彦：クリティカルパス情報を用いた分散命令発行型マイクロプロセッサ向けステアリング方式，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG6, pp.12–22 (2004).

(平成 19 年 1 月 22 日受付)
(平成 19 年 4 月 27 日採録)

佐藤　幸紀（正会員）

平成 13 年東北大学工学部機械知能工学科卒業．平成 15 年東北大学大学院情報科学研究科前期 2 年の課程修了．平成 18 年東北大学大学院情報科学研究科後期 3 年の課程修了，博士（情報科学）．同年ファインアーク株式会社入社．入社後も東北大学大学院情報科学研究科大学院研究生，民間等共同研究員研究として組み込みプロセッサシステムの低電力化に関する研究開発に従事．平成 19 年 4 月より北陸先端科学技術大学院大学助教（情報科学センター）．計算機アーキテクチャ，高性能計算システムやその応用に関する研究に従事．

鈴木　健一（正会員）

平成 4 年東北大学工学部機械工学科卒業．平成 6 年東北大学大学院情報科学研究科博士前期 2 年の課程（機械工学専攻）修了．平成 9 年同博士課程（情報基礎科学専攻）修了．同年，宮城工業高等専門学校情報デザイン学科助手に採用．平成 11 年同講師に昇任．平成 15 年から東北大学大学院情報科学研究科に講師として勤務．

中村　維男（正会員）

平成 6 年米国 Stanford 大学客員正教授，平成 19 年東北大学名誉教授，英国ロンドン大学インペリアル校教授・フェロー（平成 19 年秋開講），IEEE フェロー，IEEE Taylor L. Booth Award 2004 年受賞者（1 名），IEEE シンポジウム COOL Chips 組織委員会委員長．