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

2007－ARC－172 (6)
2007－HPC－109 (6)
2007／3／1

スラック命令数を増加させるスラック共有化手法

小林 良太郎† 市川 彰孝† 島田 俊夫†

本稿では，依存関係に基づいてローカル・スラックを共有化する手法を提案する．ローカル・スラックとは，他の命令に影響を与えることなく，その命令の実行レイテンシを増加させることのできるサイクル数である．提案機構では，ある命令の持つローカル・スラックを，依存関係のある命令間で共有する．これにより，ローカル・スラックを持たない命令が，スラックを利用できるようになる．評価した結果，従来手法に対し，提案手法はスラック命令数を，同一 IPC で 8%，10%の IPC 低下で 35%増加させることができると分かった．

Slack Sharing Technique for Increasing the Number of Slack Instructions

Ryotaro Kobayashi,† Akitaka Ichikawa† and Toshio Shimada†

In this paper, we propose a mechanism that shares local slack based on dependences. Local slack of a dynamic instruction is the maximum number of cycles the execution latency of the instruction can be increased without affecting any other instruction. In our mechanism, sharing an instruction's local slack between dependent allows instructions that have not local slack to utilize slack. Our evaluation results show that slack sharing increases slack instruction rate to 8% with the same performance and 35% with the 10% performance degradation.

1. はじめに

近年，クリティカル・パスに関する情報を用いた，マイクロプロセッサの高速化や消費電力の削減に関する研究が数多く行われている[1),2),7),9),12),13)]．クリティカル・パスとは，プログラム全体の実行時間を決定する動的な命令列で構成されるパスである．クリティカル・パス上の命令の実行レイテンシをたとえ 1 サイクルでも増加させると，プログラム全体の実行サイクル数が増加する．しかし，クリティカル・パス情報は命令がクリティカル・パス上にあるかないかの 2 通りしかなく，クリティカル・パス情報は，適用範囲が狭くなってしまう．

これに対し，クリティカル・パスの代わりに，命令の**スラック**を用いる手法が提案されている[3),4)]．命令のスラックとは，プログラム全体の実行サイクル数を増加させることなく，その命令の実行レイテンシを増加させることのできるサイクル数である．命令のスラックが分かれば，各命令の実行レイテンシを，プログラムの実行に影響しない範囲で，どの程度増加させられるのかが分かる．

各動的命令のスラックは，ある範囲を持った値である．スラックの最小値は常に 0 である．一方，スラックの最大値（**グローバル・スラック**[3)]）は動的に決まる．ある命令のグローバル・スラックを求めるためには，ある命令の実行レイテンシの増加が，プログラム全体の実行サイクル数に与える影響を，プログラムの実行中に調べなければならない．そのため，グローバル・スラックを求めるのは非常に難しい．

この問題を解決するアプローチとして，**ローカル・スラック**[3)] を利用するという方法がある．命令のローカル・スラックとは，後続命令の実行に影響を与えないスラックの最大値である．ある命令のローカル・スラックは，依存関係のある後続命令に着目するだけで，容易に求めることができる．こうした利点があることから，これまでに，ローカル・スラックを予測する手法やその利用方法が提案されてきた[5),6),10),11)]．

しかし，従来の予測手法では，ローカル・スラックが 1 以上存在すると予測できる命令の数（**スラック命令数**）が少なく，スラックを利用できる機会が十分に確保できない．

そこで，本論文では，ある命令の持つローカル・スラックを，依存関係のある複数の命令間で共有する手法を提案する．この機構では，ローカル・スラックを持つ命令を始点として，ローカル・スラックを持たない命令間で，依存先から依存元へと，共有可能なスラックが存在するという情報を伝播させていく．そしてこの情報を基に，発見的な手法を用いて，各命令が利用するスラックの量を決定する．これにより，ローカル・スラックを持たない命令が，スラックを利用できるようになる．

2 章ではスラックについて説明する．3 章では従来のスラック予測機構について説明する．4 章ではスラック命令数を増加させる手法を提案する．5 章では提案手法を実現するための機構を示し，6 章で評価を行う．最後に 7 章でまとめる．

2. スラック

スラックの説明に用いるプログラムを**図 1** に示す．図において，ノードは命令を示し，エッジは命令間のデータ依存関係を示す．縦軸は命令を実行したサイクルを示す．ノードの長さは命令の実行レイテンシを示す．実行レイテンシは，命令 `i1`，`i4`～`i7`，`i11` が 2 サイクル，その他が 1 サイクルである．

まず，`i3` のグローバル・スラックについて考える．

`i3` の実行レイテンシを 7 サイクル増加させた場合，それに

† 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University

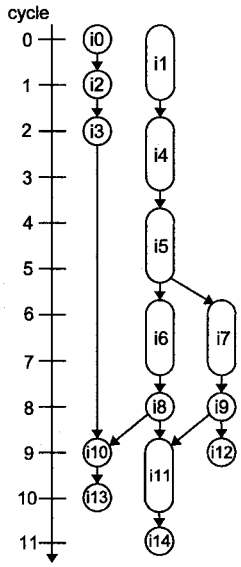


図 1　スラック

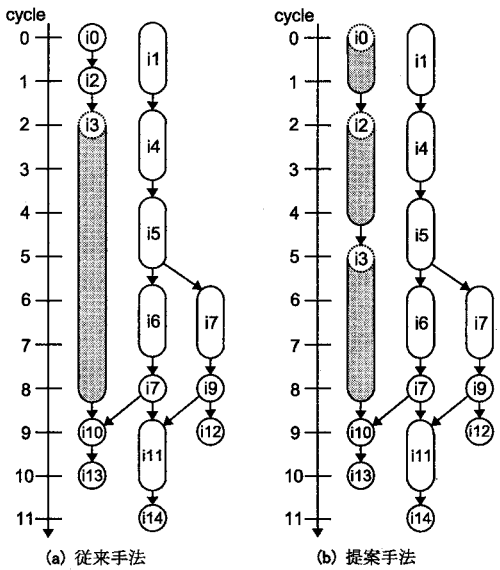


図 2　スラックの利用

直接的，間接的に依存する i10，i13 の実行が遅れる．その結果，i13 は，プログラム中もっとも最後に実行される i14 と同時刻に実行される．したがって，i3 の実行レイテンシをこれ以上増加させると，プログラム全体の実行サイクル数が増加する．つまり，i3 のグローバル・スラックは 7 である．このように，ある命令のグローバル・スラックを求めるためには，その命令の実行レイテンシの増加が，プログラム全体の実行に与える影響を調べる必要がある．そのため，グローバル・スラックの判定は非常に難しい．

次に，i3 のローカル・スラックについて考える．

i3 の実行を 6 サイクル増加させた場合，後続命令の実行に影響は与えない．しかし，これ以上実行レイテンシを増加させると，i3 に直接依存する i10 の実行が遅れる．つまり，i3 のローカル・スラックは 6 である．このように，ある命令のローカル・スラックを求めるには，その命令に依存する命令への影響に着目すれば良い．したがって，ローカル・スラックは比較的容易に判定することができる．

3. 従来のローカル・スラック予測機構

従来機構の概要について述べる．

時刻に基づく機構[5),11)] では，ある命令がデータを定義した時刻と，そのデータが他の命令によって参照された時刻の差からローカル・スラックを計算し，その値をスラック表と呼ばれるテーブルに記録する．そして，次回その命令をフェッチするときにスラック表を参照し，スラック表から得た値を，その命令のローカル・スラックであると予測する．また，この機構では，ローカル・スラックの動的な変化に対応するため，命令の実行を遅れさせたか否かという情報に基づいて，ローカル・スラックを計算する機能も備えている[5)]．

発見的手法に基づく機構[10)] も，時刻に基づく機構と同様に，スラック表を用いる．しかし，スラック表に記録する値を決定する方法が異なる．この機構では，命令実行時の振る舞い（分岐予測ミス，キャッシュ・ミス，フォワーディング）を観測しながら，予測するローカル・スラック（予測スラック）を増加させ，実際のローカル・スラック（実スラック）に近づけていく．また，この機構では，ローカル・スラックの動的な変化に対応するため，命令実行時の振る舞いに基づいて，予測スラックを減少させる機能や，予測スラックの増減の緩急を制御できる機能を備えている[10)]．

上記の手法は，いずれも，同程度の予測精度を達成できるが，スラック命令数が少ないという問題点がある．例えば，発見的手法では，4 命令発行のプロセッサにおいて，予測可能なスラック命令数は，全実行命令の 3〜5 割程度である（評価の詳細は 6 章で述べる）．スラック命令数が少なければ，スラックを利用する機会も制限されてしまう．そこで，スラック命令数を増加させる方策を考えることが重要となる．

4. スラック命令数を増加させる手法

本章では，ある命令の持つローカル・スラックを，当該命令だけでなく，それ以外の命令も利用する（**共有化する**）手法を提案する．スラックの共有化によって，ローカル・スラックを持たなかった命令が，スラックを利用できるようになれば，スラック命令数を増加させることができる．

ある命令の持つローカル・スラックを利用できるのは，その命令に対し，直接的，間接的にオペランドを供給する命令である．

例えば，図 1 において，i3 は，ローカル・スラックを持つ命令である．そして，i0，i2 は，i3 に直接的，間接的にオペランドを供給する命令である．したがって，i3 の持つローカル・スラックは，i0，i2，i3 の間で共有化できるということになる．

それでは，**図 2** を用いて，どのようにしてスラックの共有化が行われるのかを説明する．

図 2(a)，(b) は，それぞれ，スラックを共有化しない従来手法の場合，スラックを共有化する提案手法の場合を示す．従来手法では，i3 のローカル・スラックを利用するのは，i3 のみである．一方，提案手法では，i3 のローカル・スラックを，i3 だけでなく，i0，i2 も利用していることが分かる．

なお，各命令の利用するスラックの量は，スラックの予測方法によって決まるため，必ずしも図の例のようにはなるわけではないことに注意されたい．

つぎに，スラックを共有化する命令を求める方法について考える．

一般に，プログラムには参照の局所性が存在しており，一部の静的命令が，繰り返し何度も実行されることが知られている．例えば文献 8) によれば，Spice プログラムにおいて，10%弱の静的命令が動的実行命令数の 90%を占めることが示されている．我々はこの性質に着目し，共有化を行う命令を一回の実行で求めるのではなく，複数回の実行に分けて求めることにより，命令間の依存関係をたどる機構を比較的単純に実現できると考えた．

上記の考えに基づき，我々は，ローカル・スラックを持つ命令を始点とし，依存先から依存元へと依存関係を逆にたどりながら，共有可能なスラックが存在するという情報（**共有化情報**）を，複数回に分けて徐々に伝播させていくという方法を提案する．例えば，図 2(b) では，1 回目の実行において，ローカル・スラックを持つ `i3` から `i2` へと共有化情報を伝播させ，2 回目の実行において，`i2` から `i0` へと共有化情報を伝播させる．直接的な依存関係さえ分かれば，共有化情報を伝播させることができるため，比較的容易に実現できると考えられる．

また，ローカル・スラックの動的な変化に対応するため，共有化情報の伝播の速度を変化させることができるようにする．具体的には，命令は，自身の予測スラックがある閾値（伝播の閾値）以上になったら，共有化情報を伝播させることとする．これ以降，伝播の閾値のことを Pth と表す．

なお，共有化情報が有効になるのは，命令が全ての依存先から共有化情報を受け取ったときだけであることに注意されたい．共有化情報を伝播しない依存先がある場合，そこから依存関係をたどっていった先に，ローカル・スラックを持つ命令は存在しないので，当該命令がスラックを利用すると，プログラムの実行に悪影響を与えてしまう．

例えば，図 1 の `i9` は，ローカル・スラックを持つ `i12` から共有化情報を受け取るが，`i11` からは共有化情報を受け取らない．この場合，`i9` がスラックを利用してしまうと，直接的，間接的に依存する `i11`，`i14` の実行が遅れ，プログラム全体の実行サイクル数が増加してしまうので，共有化情報は有効にならない．

最後に，スラックの予測方法について考える．

予測には 2 種類ある．ローカル・スラックの予測と，共有化情報を受け取った命令が利用するスラック（**共有化スラック**）の予測である．

まず，ローカル・スラックの予測について考える．ローカル・スラックは動的に変化するが，共有化を行うため，その変化は更に複雑になる．これに対応するため，ローカル・スラックの予測手法として，予測スラックの増減の緩急を制御できる，発見的ローカル・スラック予測[10] を用いることとする．

つぎに，共有化スラックの予測について考える．共有化スラックは，相関関係のあるローカル・スラックと同様，複雑に変化する．そこで，ローカル・スラックと同じ考え方を用いて，共有化スラックも予測することとする．ただし，命令実行時の振る舞いではなく，共有化情報に基づいて，予測スラックを決定することとする．

5. 提案機構

本章では前章で示した提案手法を実現する機構について説明する．まず，提案機構の概要について述べる．次に，提案機構の各構成要素について説明する．最後に，全体の動作について詳しく説明する．

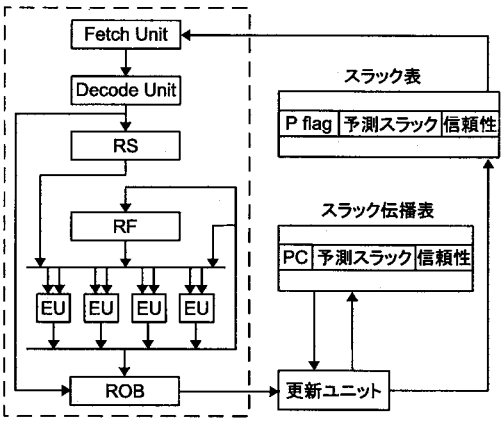


図 3　提案機構の全体図

5.1 概　　要

提案機構を備えたプロセッサの例を**図 3** に示す．図中の点線で囲まれた部分は，プロセッサを簡略化して示したもので，フェッチ・ユニット，デコード・ユニット，命令ウインドウ（RS），レジスタ・ファイル（RF），実行ユニット（EU），リオーダ・バッファ（ROB）を持つ．プロセッサの右側は，提案機構を示す．

提案機構は，以下 3 つの要素で構成される．

- スラック表
- スラック伝播表
- 更新ユニット

スラック表は，命令ごとに，P flag☆，予測スラック，信頼性を保持する．P flag は，予測の内容を示す．0 の場合，ローカル・スラックの予測を行っていることを示し，1 の場合，共有化スラックの予測を行っていることを示す．ローカル・スラックを予測した後でなければ，共有化情報は伝播できないため，P flag の初期値は 0 とする．

スラック伝播表は，共有化情報を伝播させるために用いる．この表は，論理レジスタ番号をインデクスとする．各エントリは，対応するレジスタを定義した命令の PC，予測スラック，信頼性を保持する．

更新ユニットは，プロセッサからコミットした命令に関する情報を受け取った後，ローカル・スラックと共有化スラックを予測する．その際，スラック表とスラック伝播表にアクセスする．

スラック予測動作の概要を以下に示す．

- ステップ 1（スラック表の参照）
  プロセッサは，フェッチ時に，PC をインデクスとしてスラック表を参照し，対応するエントリからフェッチした命令の予測スラック，P flag，信頼性を得る．
- ステップ 2（命令実行時の振舞の観測）
  プロセッサは，命令実行時の振舞を観測し，命令の R flag☆☆）をセット／リセットする．R flag は，予測スラックが実スラックに到達しているときに 1，そうでないときに 0 となるフラグである．

☆ predict flag の略
☆☆ reach flag の略．

- ステップ 3（ローカル・スラックの予測）
  更新ユニットは，コミットした命令のローカル・スラックを予測し，スラック表を更新する．
- ステップ 4（共有化情報の伝播）
  更新ユニットは，スラック伝搬表を用いて，共有化情報の伝播を行い，コミットした命令の先行命令に対し，S flag☆をセット／リセットする．S flag は，全ての依存先から共有化情報を受け取ったときに 1，そうでないときに 0 となるフラグである．
- ステップ 5（共有化スラックの予測）
  更新ユニットは，S flag をセット／リセットした命令の共有化スラックを予測し，スラック表を更新する．

以下では，ステップ 2〜5 を詳細に説明する．ステップ 1 は，上記の概要で十分であるため省略する．

5.2 命令実行時の振舞の観測

プロセッサは，命令実行時の振る舞いを観測し，下記のいずれかに該当した場合，予測スラックは実スラックに到達した（実スラック到達条件が成立した）と判定し，命令の R flag を 1 にセットする．そうでない場合，成立していないと判定し，R flag を 0 にリセットする．

- 分岐予測ミス
- キャッシュ・ミス
- 後続命令に対するオペランド・フォワーディング
- 後続命令に対するストア・データ・フォワーディング

そして，命令コミット時に，その命令の PC，P flag，信頼性，予測スラック，R flag を，更新ユニットに送る．

なお，従来のローカル・スラック手法との違いは，P flag が存在する点のみである．

5.3 ローカル・スラックの予測

更新ユニットは，コミットした命令の P flag が 0 であれば，発見的手法[10] に基づいて，ローカル・スラックを予測する．具体的には，信頼性と予測スラックを計算し，スラック表を更新する．P flag は変更しない．

以下に，ローカル・スラック予測に関係するパラメータとその内容を示す．

- $Vmax\_l$: 予測スラックの最大値
- $Vinc\_l$: 予測スラックの 1 回あたりの増加量
- $Vdec\_l$: 予測スラックの 1 回あたりの減少量
- Cth_l: 信頼性の閾値
- $Cinc\_l$: 信頼性の 1 回あたりの増加量
- $Cdec\_l$: 信頼性の 1 回あたりの減少量

上記のパラメータを用いて，信頼性と予測スラックを計算する手順を説明する．なお，予測スラックと信頼性の最小値は，いずれも 0 である．実スラック到達条件が成立していれば（R flag が 1 であれば），信頼性を $Cdec\_l$ 減少させ，R flag が 0 であれば $Cinc\_l$ 増加させる．信頼性が Cth_l 以上になったら，予測スラックを $Vinc\_l$ 増加させ，信頼性を 0 にリセットする．一方，信頼性が 0 になったら，予測スラックを $Vdec\_l$ 減少させる．

なお，従来のローカル・スラック手法との違いは，P flag に応じて，予測するかどうかを決定している点のみである．

これ以降の動作はすべて，スラックの共有化のために，本論文で新たに導入したものである．

5.4 共有化情報の伝播

まず，共有化情報を伝播させるかどうかを決定するため，コミットした命令の予測スラックと，伝播の閾値 Pth を比較する．

予測スラックが Pth 未満の場合，共有化情報は伝播させないので，コミットした命令の依存元は，共有化情報を受け取らない．この場合，まず，コミットした命令のソース・レジスタ番号でスラック伝播表を参照し，依存元の情報（PC，予測スラック，信頼性）を読み出すとともに，参照したエントリをクリアする．また，依存元の S flag を 0 にリセットする．そして，スラック伝播表から得た依存元の情報と，0 にリセットした S flag を，更新ユニットに送る．

一方，予測スラックが Pth 以上である場合，コミットした命令の依存元は，共有化情報を受け取ることが分かる．しかし，4 章で説明したように，共有化情報が有効になるのは，命令が全ての依存先から共有化情報を受け取るときだけである．そのため，この時点では何も行わない．

命令が，全ての依存先から共有化情報を受け取る（S flag が 1 である）ことが決定するのは，S flag が一度も 0 にリセットされたことがない状態で，デスティネーション・レジスタの再定義が行われるときである．そこで，そのような場合を検出するために，コミットした命令のデスティネーション・レジスタ番号でスラック伝播表を参照し，同じレジスタを定義した先行命令の情報（PC，予測スラック，信頼性）を読み出す．もし，参照したエントリがクリアされていなければ，S flag が 0 にリセットされたことは一度もないということなので，先行命令の S flag を 1 にセットする．そして，スラック伝播表から得た命令の情報と，1 にセットした S flag を，更新ユニットに送る．

上記の処理が終了した後，スラック伝播表の更新を行う．

コミットした命令のうち，ローカル・スラックを持たない命令を，スラック伝播表に登録する．命令がローカル・スラックを持たないのは，P flag が 1 である場合，あるいは，P flag，予測スラック，信頼性がいずれも 0 である場合である．そこで，これらの場合，スラック伝播表のデスティネーション・レジスタに対応するエントリに，コミットした命令の PC，予測スラック，信頼性を書き込む．これら以外の場合，対応するエントリをクリアする．

5.5 共有化スラック予測

スラック伝播表からデータ（命令の PC，信頼性，予測スラック，S flag）が送られてくると，更新ユニットは共有化スラックの予測を行う．

4 章で述べたように，ローカル・スラック予測と同じ考え方を用いて信頼性と予測スラックを計算し，これらの計算結果に応じて P flag を決定した後，スラック表を更新する．

以下に，共有化スラック予測に関係するパラメータとその内容を示す．

- $Vmax\_s$: 予測スラックの最大値
- $Vinc\_s$: 予測スラックの 1 回あたりの増加量
- $Vdec\_s$: 予測スラックの 1 回あたりの減少量
- Cth_s: 信頼性の閾値
- $Cinc\_s$: 信頼性の 1 回あたりの増加量
- $Cdec\_s$: 信頼性の 1 回あたりの減少量

上記のパラメータを用いて，信頼性と予測スラックを計算する手順を説明する．なお，予測スラックと信頼性の最小値は，いずれも 0 である．命令の S flag が 1 であれば信頼性を $Cinc\_s$ 増加させ，そうでなければ信頼性を $Cdec\_s$ 減少させる．信頼性が Cth_s 以上になったら，予測スラックを $Vinc\_s$ 増加させ，信頼性を 0 にリセットする．一方，信頼性が 0 になったら，予測スラックを $Vdec\_s$ 減少させる．

上記の手順は，基本的に，ローカル・スラック予測の場合と同じであるが，信頼性を増減させる際に，S flag を参照する点が異なる．

最後に，信頼性と予測スラックの計算結果から P flag を決定する手順について述べる．計算した信頼性，あるいは，

☆ share flag の略．

表 1 測定条件

フェッチ／発行幅	4 命令
命令ウィンドウ	32 エントリ
ROB	64 エントリ
機能ユニット数	iALU 4, iMULT/DIV 2, fpALU 3, fpMULT/DIV/SQRT 2
命令キャッシュ	8KB, 2 ウェイ, 32B ライン, ミスペナルティ6 サイクル
データキャッシュ	32KB, 2 ウェイ, 32B ライン, 2 ポート, ミスペナルティ6 サイクル
2 次キャッシュ	2MB, 4 ウェイ, 64B ライン, ミスペナルティ100 サイクル
ストアセット	4K エントリ SSIT, 128 エントリ LFST
分岐予測機構	gshare 6 ビット履歴, 8K エントリ PHT, 予測ミスペナルティ5 サイクル

予測スラックが 1 以上であった場合，現在，その命令に対して共有化スラックの予測を行っていることを示すので，P flag を 1 にセットする．そうでない場合，P flag を 0 にセットする．

6. 評　　価

まず，評価環境について述べる．次に，ローカル・スラックの予測精度を調査する．最後に，提案機構の評価を行う．

6.1 評価環境

シミュレータには，SimpleScalar Tool Set のスーパスカラ・プロセッサ用シミュレータを用い，提案手法を組み込んで評価した．命令セットには MIPS R10000 を拡張した SimpleScalar/PISA を用いた．ベンチマーク・プログラムは，SPECint2000 の bzip2, gcc, gzip, mcf, paser, perl, votex, vpr の 8 本を使用した．gcc では 10 億命令，その他では 20 億命令をスキップした後，1000 万命令を実行した．測定条件として**表 1** を用いた．

以下のモデルに対して評価を行った．

- **NO-PRED モデル**
  スラックの予測を行わないモデルである．
- **Local モデル**
  発見的手法を用いて，ローカル・スラックのみ予測するモデルである．
- **Share モデル**
  Local モデルに，スラックの共有化を導入したモデルである．

Local モデルと Share モデルで用いるスラック表の構成は，ローカル・スラック予測精度がほぼ飽和する，4K エントリ，2 ウェイとした．Share モデルで用いるスラック伝播表は，論理レジスタ番号をインデクスとするので，エントリ数を 64 とし，ダイレクトマップ方式でアクセスするとした．

スラック表の更新に関するパラメータの組合せは膨大な数になるので，幾つかのパラメータをある値に固定する．

まず，$Cth\_l/Cinc\_l$ と $Cth\_s/Cinc\_s$ は，どちらも，スラックを増加させる頻度を表すので，$Cinc\_l = Cinc\_s = 1$ に固定し，Cth_l と Cth_s を変化させることとする．つぎに，予測スラックの減少を迅速に行うため，$Cdec\_l = Cth$, $Cdec\_s = Cth\_s$ とする．また，限られたスラックを効率良く共有化するために，$Vmax\_l > Vmax\_s$ とする．最後に，予測スラックの値をできるだけ細かく制御するために，$Vinc\_l$, $Vinc\_s$, $Vdec\_l$, $Vdec\_s$ を全て 1 に固定する．

以上より，本章では，$Vmax\_l$, $Vmax\_s$, Cth_l, Cth_s, Pth だけを変化させて，提案方式の評価を行うこととする．しかし，依然としてパラメータの組合せの数が多いので，パラメータの上限を 7 とし，各パラメータの取り得る値を，2 通りか 3 通りに制限する．なお，$Vmax\_l < Pth$ となる場合は，共有化情報を伝播することができないので，評価しない．

最後に，提案機構に導入されている，ローカル・スラック予測機構の精度を示す．本節の評価環境において，ローカル・スラック予測機構は，全実行命令の 31%～51%に対して，1 以上のローカル・スラックを持つと予測し，全実行命令の 2%～13%に対して，実スラックよりも大きいスラックを持つと予測する．

6.2 スラック命令数と性能

本節では，Local モデルと Share モデルにおけるスラック命令数と性能を測定する．なお，各パラメータは，6.1 節で説明した方針にしたがって定めている．具体的な数値については，6.1 節，および，評価結果の図を参照されたい．

図 4 に，各モデルのスラック命令数を評価した結果を示す．図の縦軸は，全実行命令数に占めるスラック命令数の割合をベンチマーク平均で示し，横軸は更新パラメータを示す．一方，**図 5** に，各モデルの IPC を測定した結果を示す．縦軸は，NO-PRED モデルの場合で正規化した IPC を，ベンチマーク平均で示し，横軸は更新パラメータを示す．図 4(a) と図 5(a) は，$Vmax\_l = 3, Vmax\_s = 2$ の場合を示し，図 4(b) と図 5(b) は，$Vmax\_l = 7, Vmax\_s = 3$ の場合を示す．4 本で組になった棒グラフは，左から順に，Local モデルの場合，Share モデルにおいて，Cth_s が 1, 3, 7 の場合である．

図 4 より，Cth_l と Pth が同一であるモデル同士を比較すると，Share モデルにおけるスラック命令数の割合は，Local モデルよりも，6%～36%ポイントも高くなる．また，スラック命令数の割合は，Cth_l, Cth_s, Pth が大きくなるほど減少し，$Vmax\_l$, $Vmax\_s$ が大きくなるほど増加することが分かる．

一方，図 5 より，Cth_l と Pth が同一であるモデル同士を比較すると，Share モデルは，Local モデルよりも，IPC が 0%～17%ポイント低くなることが分かる．また，IPC は，スラック命令数の場合と逆に，Cth_l, Cth_s, Pth が大きくなるほど増加し，$Vmax\_l$, $Vmax\_s$ が大きくなるほど減少することが分かる．

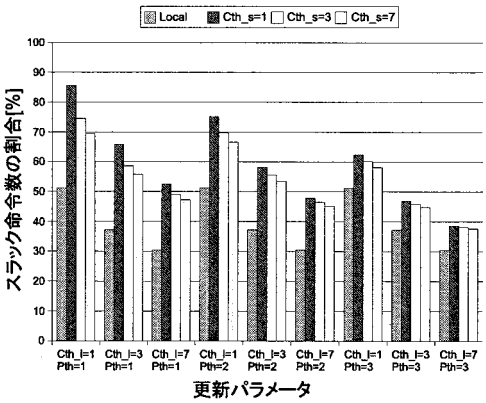
図 4 と図 5 より，Share モデルは，Local モデルに対して，性能を維持したまま，スラック命令数を増加させることができ，性能低下を許容するのであれば，スラック命令数をさらに増加させることができる．例えば，$Vmax\_l = 3$ の場合，Local モデルはスラック命令数の上限が 51%である．これに対して Share モデルは，性能を低下させること無く，スラック命令数を 59%まで増加させることができる．また，4%ポイント，10%ポイントの性能低下で，スラック命令数をそれぞれ 75%，86%まで増加させることができる．

なお，予測したスラックを応用して，効果的に高速化，低消費電力化を実現するためには，スラックの応用方法に関する検討が重要であるが，それは今後の課題である．

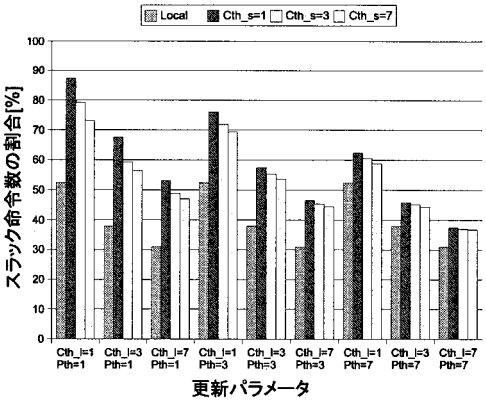
7. ま と め

今回我々は，依存関係に基づいてローカル・スラックを共有化する機構を提案した．提案機構では，ある命令の持つローカル・スラックを，依存関係のある命令間で共有するため，スラックを利用できる命令の数を増やすことができる．

謝辞　本研究の一部は，株式会社半導体理工学センターとの共同研究により行った．

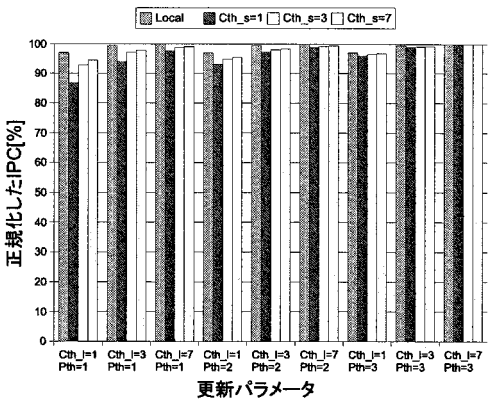


(a) $Vmax\_l = 3, Vmax\_s = 2$

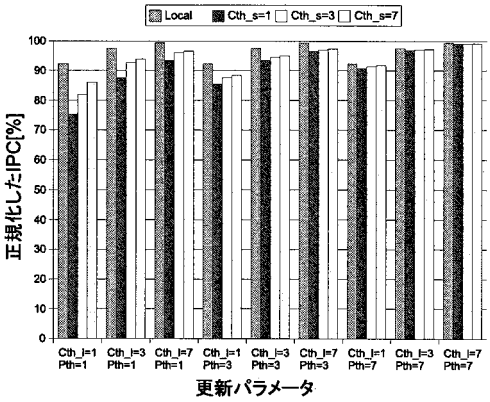


(b) $Vmax\_l = 7, Vmax\_s = 3$

図 4 スラック命令数



(a) $Vmax\_l = 3, Vmax\_s = 2$



(b) $Vmax\_l = 7, Vmax\_s = 3$

図 5 IPC

参 考 文 献

[1] 千代延昭宏 ほか：低消費電力プロセッサアーキテクチャ向けクリティカルパス予測器の提案，情報処理学会研究報告 2002-ARC-149, 2002 年 8 月.

[2] B. Fields, et al., "Focusing Processor Policies via Critical-Path Prediction," In *Proc. ISCA-28*, June 2001.

[3] B. Fields, et al., "Slack: Maximizing Performance under Technological Constraints," In *Proc. ISCA-29*, May 2002.

[4] B. Fields, et al., "Using Interaction Costs for Microarchitectural Bottleneck Analysis," In *Proc. MICRO-36*, Dec. 2003.

[5] 福山 智久 ほか：スラック予測を用いた省電力アーキテクチャ向け命令スケジューリング，先進的計算基盤システムシンポジウム SACSIS2005, 2005 年 5 月.

[6] 福山 智久 ほか：スラック予測を用いたクラスタ型スーパースカラ・プロセッサ向け命令ステアリング，情報処理学会研究報告 2006-ARC-169, 2006 年 8 月.

[7] 服部 直也 ほか：クリティカル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. SIG 6(ACS 6) pp.12-22, 2004 年 5 月.

[8] J. L. Hennessy, et al., *Computer Architecture: A Quantitative Approach, 2nd Edition*, Morgan Kaufmann Publishing Inc., San Francisco, CA, 1996.

[9] 小林良太郎 ほか：データフロー・グラフの最長パスに着目したクラスタ化スーパースカラ・プロセッサにおける命令発行機構，2001 年並列処理シンポジウム JSPP2001, 2001 年 6 月.

[10] 小林良太郎 ほか：発見的手法に基づくローカル・スラック予測機構，2006 年先進的計算基盤システムシンポジウム SACSIS 2006, pp.385-394, 2006 年 5 月.

[11] 劉 小路 ほか：クリティカリティ予測のためのスラック予測，先進的計算基盤システムシンポジウム SACSIS2004, 2004 年 5 月.

[12] J. S. Seng, et al., "Reducing Power with Dynamic Critical Path Information," In *Proc. MICRO-34*, Dec. 2001.

[13] E. Tune, et al., "Dynamic Prediction of Critical Path Instructions," In *Proc. HPCA-7*, jan. 2001.