スレッドレベル投機的並列処理アーキテクチャにおけるデータ依存制約緩和手法の効果

目次勝彦[†] 村上和彰[††]

プログラムを並列実行する際に並列度を決定する要因となるのは，主に命令間のデータ依存と制御依存である．本稿では 3 つの制御依存制約の緩和手法（分岐予測に基づいた投機的実行，分岐命令が及ぼす制御依存関係の正確な解析，複数の制御フローの同時実行）の利用により高い並列度を得ることが可能であるスレッドレベル投機的並列処理（TLSP: Thread-Level Speculative Parallel processing）アーキテクチャにおいて 3 つのデータ依存制約の緩和手法（メモリ・アドレスの一致/不一致解析，リネーミング，値予測）を利用することによる効果の測定を行った．その結果，メモリ・アドレスの一致/不一致解析がリネーミングの効果に与える影響の大きさ，および値予測の効果の高さが判明した．

Effects of Relaxing Constraints due to Data Dependences on Thread-Level Speculative Parallel Processing Architecture

Katsuhiko Metsugi[†] and Kazuaki Murakami[††]

Two fundamental restrictions that limit the amount of instruction-level parallelism extracted from sequential programs are control flow and data flow. TLSP (Thread-Level Speculative Parallel processing) architecture gains high parallelism using three techniques (speculation with branch prediction, control dependence analysis, executing multiple flows of control) which relax constraints due to control dependences. In this paper, we evaluate the effects of three techniques (memory disambiguation, renaming, value prediction) which relax constraints due to data dependences on TLSP architecture. We have two major results. First, memory disambiguation affects renaming. Second, value prediciton has large effects on TLSP architecture.

1. はじめに

プログラム単体の実行時間を短縮する手法の 1 つに「並列処理」があるが，その効果はプログラムが本質的に有する「制御依存（control dependence）」および「データ依存（data dependence）」により以下の制約を受ける．

- **制御依存：**分岐命令が存在すると，それ以降のすべての命令は当該分岐命令が分岐するか否かが決まるまでその実行を開始できない．
- **データ依存：**ある先行命令にデータ依存している後続命令は，当該先行命令の実行が完了するまでその実行を開始できない．

Lam らは 1992 年の ISCA で発表した論文 “Limits of Control Flow on Parallelism” [11]の中で，制御依存に起因する制約を緩和する以下の 3 手法について，それらが並列度に与える効果の上限値に関する報告を行った．

- 分岐命令に後続する命令に対する「分岐予測を用いた投機的実行（speculation with branch prediction）」
- 分岐命令が及ぼす制御依存関係の正確な解析（control dependence analysis）
- 複数の制御フローの同時実行（executing multiple flows of control）

上記 3 手法の並列度に与えた正効果，特に投機的実行と複数制御フロー同時実行の効果の大きさから，「**スレッドレベル投機的並列処理（TLSP: Thread-Level Speculative Parallel processing）アーキテクチャ**」に関する研究がさかんに行われるようになり，現在までに以下のようなアーキテクチャが提案されている．

† 九州大学大学院システム情報科学府情報理学専攻
Department of Informatics, Kyushu University
†† 九州大学大学院システム情報科学研究院情報理学部門
Division of Informatics, Kyushu University

表 1 Lam and Wilson の評価結果
Table 1 Results of Lam and Wilson's evaluation.

		awk	ccom	eqntott	espresso	gcc	irsim	latex	調和平均
Lam and Wilson [11] ⟨並列度⟩	BASE	2.85	2.13	1.98	1.51	2.10	2.31	2.71	2.14
	CD	3.24	2.51	2.05	1.54	2.55	2.66	3.17	2.39
	CD-MF	5.32	5.61	5.21	7.49	14.63	11.89	6.18	6.96
	SP	9.22	6.92	6.40	4.16	7.76	8.40	7.60	6.80
	SP-CD	12.89	9.83	18.09	19.55	13.18	15.82	9.72	13.27
	SP-CD-MF	41.88	18.05	225.90	402.85	66.29	45.86	18.65	39.62
	ORACLE	242.77	46.80	3282.91	742.30	174.50	265.42	131.69	158.26

表 2 TLSP アーキテクチャの評価結果
Table 2 Comparison of ILP on TLSP architectures.

		compress	eqntott	espresso	gcc	go	idct	li	vortex
Multiscalar [15] ⟨UICR⟩	1PE	0.74	0.85	0.93	0.86				
	2PEs	1.52	1.32	1.44	1.25				
	4PEs	2.15	2.18	1.96	1.56				
	8PEs	2.44	2.39	2.98	1.70				
	12PEs	2.52	2.43	3.74	1.73				
MUSCAT [4] ⟨性能向上比⟩	1PE	1.00	1.00				1.00		
	2PEs	1.15	1.18				1.98		
	4PEs	1.48	1.60				2.90		
	8PEs	1.80	2.18				3.00		
Hydra [8] ⟨性能向上比⟩	4PEs	1.00	0.58					1.04	0.62
SKY [2] ⟨IPC⟩	2PEs	1.2			2.4	2.7		2.8	3.1
	4PEs	1.9			2.4	3.0		2.9	3.2
	8PEs	3.3			2.5	3.1		2.9	3.2

UICR : Useful Instruction Completion Rate
IPC : Instruction Per Clock cycle
性能向上比：1PE に対する性能向上比

- UW-Madison の Multiscalar [15]
- NEC の MUSCAT（Merlot）[4]
- Stanford 大学の Hydra [8]
- 名古屋大学の SKY [2]
- UIUC の Speculative CMP（Chip MultiProcessor）[10]
- 東京大学の OCHA-Pro [3]

上記のうち，MUSCAT は MP98 [14] としてすでに実用化されている．しかしながら，Lam and Wilson の評価，ならびに，これら TLSP アーキテクチャに関する研究は次の課題をかかえている．

(1) Lam and Wilson の評価結果と各種 TLSP アーキテクチャの性能評価結果との間には，得られる並列度の値に著しい乖離がみられる（**表 1** および**表 2** 参照）．しかしながら，その原因に対する考察はほとんど行われていない．

(2) Lam and Wilson の評価では，データ依存に対する対処法として 1 種類しか評価していない．これは，「オペランドの所在場所がレジスタ，メモリにかかわらず，真のデータ依存関係（フロー依存）のみ遵守する，しかもメモリ・オペランドに関する依存関係の有無もあらかじめ判明している」という理想的な対処法である．しかしながら，現実にはフロー依存以外に逆依存や出力依存といった偽のデータ依存関係も存在し，これらに対する対処法いかんで性能が異なる．また，フロー依存に関しても，レジスタと違ってオペランドがメモリに存在する場合は，依存関係の有無が実行時になるまで判明しないという「memory ambiguity」問題が残る．結局のところ，Lam and Wilson の評価は「制御依存に対する対処法が並列度に与える効果の上限値を求める」という意味では有効であるが，「データ依存に対する対処法と制御依存に対する対処法との間の相関関係」に関しては何ら情報を提供していない．

(3) 真のデータ依存関係であるフロー依存に起因する制約を緩和する手法として，一部の TLSP アーキテクチャでは「値予測（value prediction，data speculation）」が採用されている．

いわゆる，「**スレッドレベル・データ投機型並列処理（TLDSP: Thread-Level Data-Speculative Parallel processing）アーキテクチャ**」である．これは，Lam and Wilson の評価で用いられたフロー依存に対する対処法よりもさらに強力な対処法である．しかしながら，その強力さにもかかわらず，本手法が並列度に与える効果の上限値に関する評価はいまだなされていない．

我々は上記の課題の解決を目指して研究を行っている．まず，本稿により，上記課題 (2) と (3) の解決，つまり TLSP アーキテクチャに対してデータ依存制約の緩和手法を使用することにより得られる効果の上限値に関する報告を行う．

以下，2 章で評価のための抽象マシン・モデルの定義を行い，3 章で評価を行うための環境について述べる．最後に 4 章で評価結果について議論を行い，5 章でまとめる．

2. 評価のための抽象マシン・モデル

2.1 制御依存制約の緩和手法と抽象マシン・モデル

制御依存に起因する制約を緩和する手法として，Lam and Wilson は先に述べたように以下の 3 手法を評価の対象にした．

(1) **分岐予測に基づいた投機的実行（speculation with branch prediction）：**分岐命令が存在する場合，本来ならそれ以降のすべての命令は当該分岐命令が分岐するか否かが決まるまでその実行を開始できない．しかしながら，分岐するか否かを予測して，その分岐予測に従って予測した側のパスの命令を投機的に実行することは可能である．もし分岐予測が正しければ，制御依存に起因する制約を大きく緩和することになる．

(2) **分岐命令が及ぼす制御依存関係の正確な解析（control dependence analysis）：**ある分岐命令に注目した場合，その後続命令のすべてが当該分岐命令に制御依存しているわけではない．したがって，個々の制御依存関係を正確に解析できれば，分岐命令が存在していても，当該分岐命令に制御依存していない命令の実行は当該分岐命令の実行とは無関係に独立に行うことが可能となる．

(3) **複数の制御フローの同時実行（executing multiple flows of control）：**1 個のプログラムの中でも分岐命令の存在により複数の制御フローが存在しうる．これら複数の制御フローを同時に実行（たとえば，複数のプロセッサによりそれぞれ異なる制御フローを実行）することができれば，制御依存に起因する制約を緩和することが可能である．

これらの手法を使用することで，次のような分岐命令に対する制御依存による制約を緩和することができる．

(1) **SP（分岐予測に基づいた投機的実行）：**分岐予測に成功する分岐命令

(2) **CD（分岐命令が及ぼす制御依存関係の正確な解析）：**制御依存しない分岐命令

(3) **MF（複数の制御フローの同時実行）：**逐次実行において先行して実行される分岐命令

上記の SP，CD，MF は Lam らが論文 11) で行った評価のために定義した抽象マシン・モデルに使用され，それぞれが制御依存制約の緩和手法を使用することを示すものである．論文 11) で使用された抽象マシン・モデルは表 1 で示すとおり 7 種類であり，それぞれの名前が示す制御依存制約の緩和手法を使用している．特に BASE マシンは制御依存制約の緩和手法を使用しない抽象マシン・モデルであり，ORACLE マシンは分岐予測がすべて成功すると仮定する抽象マシン・モデルである．

本稿で行う評価においては，TLSP アーキテクチャにおける命令の実行をモデル化したものである SP-CD-MF マシンを使用する．SP-CD-MF マシンは ORACLE マシンを除く抽象マシン・モデルの中で最も制御依存制約の緩和の効果が高いモデルであり，その定義は以下のようになる．

- **SP-CD-MF**：ある命令の実行は，分岐予測に失敗し，かつ制御依存する分岐命令にのみ依存して行われる．

すなわち，SP-CD-MF とは，SP，CD，MF の条件により緩和できる制御依存関係による制約をすべて緩和できる抽象マシン・モデルであり，命令は分岐命令の実行順序，および分岐予測に成功する分岐命令または制御依存しない分岐命令の実行には関係なく実行が可能となる．

2.2 データ依存制約の緩和手法と抽象マシン・モデル

データ依存には，フロー依存，逆依存，出力依存の 3 種類が存在する．また，注目しているデータがレジスタ・オペランドかメモリ・オペランドかにより，そのデータ依存制約の緩和手法も異なる．本稿では以下の手法を評価の対象とする．

(1) **メモリ・アドレスの一致/不一致解析（memory disambiguation，alias analysis）：**一般に，ロード/ストア命令のメモリ・アドレスには実行時にならないと判明しないものが存在する．この場合，いま注目しているロード/ストア命令がその先行するロード/ストア命令に対してデータ依存関係を被っているか否か，あるいは，その後続するロード/ストア命令に対してデータ依存関係を及ぼしているのか否か，実行時になるまで分からない．結局，あるロード/ストア命令に注目した場合，その先行するすべてのロード/ストア命令のメモリ・アドレスが判明し，かつ，当該ロード/ストア命令のメモリ・アドレスが判明してデータ依存関係の有無が確認されるまで当該ロード/ストア命令は実行できない．これに対して，何らかの方法でメモリ・アドレスの一致/不一致があらかじめ解析できれば，このような制約は緩和できる．

(2) **リネーミング（renaming）：**データ依存のうち逆依存と出力依存は，データ格納場所（レジスタやメモリ）を再利用することから生じる偽のデータ依存である．偽のデータ依存は，真のデータ依存（フロー依存）とは異なり，データ格納場所の名前の付け替え（レジスタ・リネーミング，ストア・バッファリング，等）により消去することが可能である．

(3) **値予測（value prediction，data speculation）：**フロー依存は真のデータ依存であり，フロー依存関係を被っている命令は，当該フロー依存関係を及ぼしている先行命令の実行が完了するまでその実行を開始できない．しかしながら，フロー依存関係の原因となっているデータがとるであろう値を何らかの方法で予測できるなら，その予測された値を用いて，フロー依存関係を被っている命令を投機的に実行することが可能となる．もし値予測が正しければ，フロー依存に起因する制約を大きく緩和することになる．

Lam らが行ったモデル化と同様に，上記のデータ依存制約の緩和手法に関する 3 つの抽象マシン・モデルを定義し，それぞれ次のような依存関係による制約を緩和できるものとする．

(1) **MD（メモリ・アドレスの一致/不一致解析）：**異なるアドレスにアクセスする先行のメモリ・アクセス命令と後続のメモリ・アクセス命令間の依存関係

(2) **RN（リネーミング）：**出力依存および逆依存関係

(3) **dSP（値予測）：**値予測に成功した命令が及ぼしていたフロー依存関係

上記の 3 手法はそれぞれ単体もしくは任意の組合せで評価のための抽象マシン・モデルとして使用することが可能であり，その組合せは 7 通りとなる．また，上記 3 手法をまったく使用しない抽象マシン・モデルを dBASE，データ依存に関する制約がまったくない抽象マシン・モデルを dORACLE とする．dBASE および dORACLE の定義は以下のとおり．

- **dBASE**：命令間のすべてのデータ依存関係を遵守して命令の実行が行われる．また，すべてのメモリ・アクセス命令間には依存関係があるものとする．
- **dORACLE**：データ依存に起因する制約はいっさいない．

ここで，dORACLE マシンにおいてはすべての命令間にデータ依存関係が存在しないため，分岐命令を除くすべての命令が 1 クロックサイクルで終了することを表し，意味のある評価とはならない．したがって本稿では dORACLE マシンの評価は行わない．

本稿の評価で使用する抽象マシン・モデルは，次のようなデータ依存関係による制約を緩和可能である．

- **MD**：
  - 異なるアドレスにアクセスする先行のメモリ・アクセス命令と後続のメモリ・アクセス命令間の依存関係
- **RN**：
  - 出力依存および逆依存関係（レジスタのみ）
- **dSP**：
  - 値予測に成功した命令が及ぼしていたフロー依存関係
- **MD+RN**：
  - 異なるアドレスにアクセスする先行のメモリ・アクセス命令と後続のメモリ・アクセス命令間の依存関係
  - 出力依存および逆依存関係（レジスタおよびメモリ）

  これは Lam らの評価で用いられたモデル．
- **RN+dSP**：
  - 出力依存および逆依存関係（レジスタのみ）
  - 値予測に成功した命令が及ぼしていたフロー依存関係
- **MD+dSP**：
  - 異なるアドレスにアクセスする先行のメモリ・

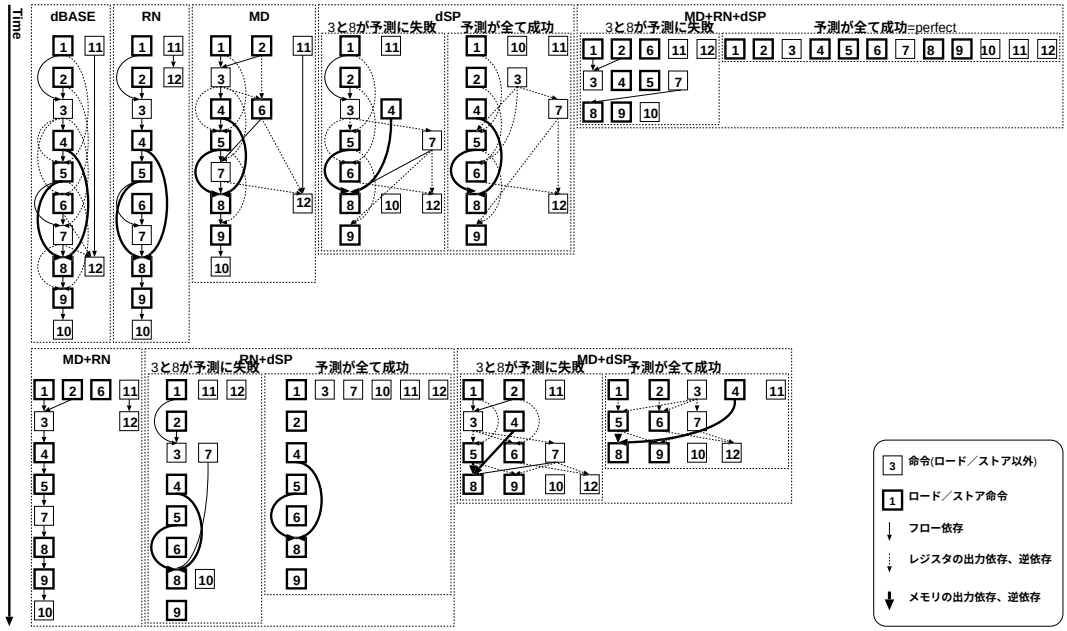

図 2 各モデルにおける命令実行の様子
Fig. 2 Execution on each machine model.

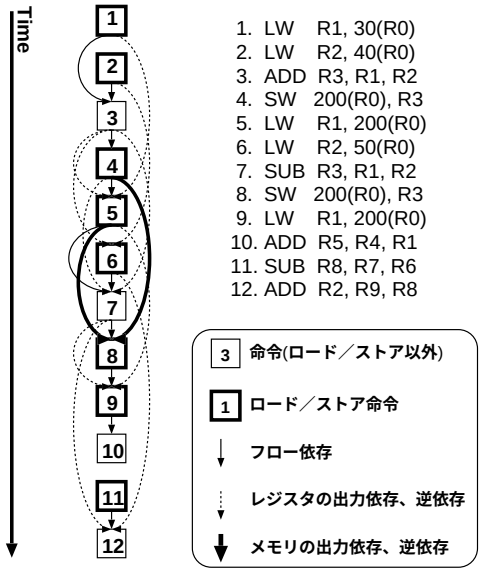

図 1 データ依存グラフと実行の流れ
Fig. 1 Data flow graph and execution.

アクセス命令と後続のメモリ・アクセス命令間の依存関係
- 値予測に成功した命令が及ぼしていたフロー依存関係

- **MD+RN+dSP**：
  - 異なるアドレスにアクセスする先行のメモリ・アクセス命令と後続のメモリ・アクセス命令間の依存関係
  - 出力依存および逆依存関係
  - 値予測に成功した命令が及ぼしていたフロー依存関係

MD を同時に使用しない場合，RN によるリネーミングの効果はレジスタ上のみに制限される．これは，MD による異なるアドレスにアクセスするメモリ・アクセス命令間の依存関係が緩和されないため，たとえメモリ・アクセスを行う命令においてリネーミングが行われたとしてもその効果はゼロであるためである．

これまでに定義したデータ依存制約の緩和手法の抽象マシン・モデルにより，どのような命令間の依存関係による制約が緩和できるのかを示すため，**図 1** のようにデータ依存グラフの命令が番号 1 から 10 まで順番に実行される場合を考える．これまでに定義した抽象マシン・モデルでは，その定義によりデータ依存関係による実行の制約が緩和され，**図 2** のように実行が行われる☆．

これらの抽象マシン・モデルは，そのデータ依存制約の緩和の効果に関して**図 3** に示すような二項関係

☆ 図 1 の例におけるメモリ・アクセス命令はすべてレジスタ R0 による相対アドレスを使用するが，R0 が一度も更新されないことからディスプレースメントの異なる命令間の依存関係がないことが容易に判明する．しかしながら，このような簡単な判定もメモリ・アドレスの一致/不一致解析であるため，メモリ・アドレスの一致/不一致解析が行われることがなければこれらの命令間の依存関係は判明しない．

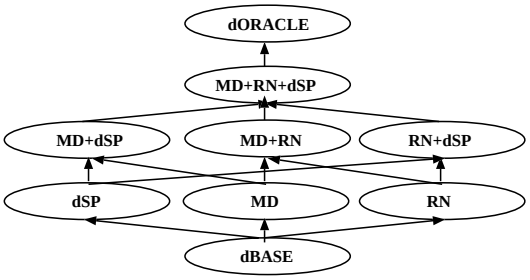

図 3 データ依存制約の緩和手法におけるモデル間の二項関係
Fig. 3 Binominal relations of abstract machine models on data dependences.

にある．

これらのデータ依存制約の緩和手法における抽象マシン・モデルを TLSP アーキテクチャに対して適用することで，データ依存制約の緩和手法それぞれが TLSP アーキテクチャに与える効果について測定を行う．

3. 評価環境

2.1 節で定義を行った TLSP アーキテクチャを表す制御依存制約の緩和手法の抽象マシン・モデル SP-CD-MF を使用し，これに 2.2 節で定義を行ったデータ依存制約の緩和手法の抽象マシン・モデルのすべてを組み合わせて評価を行う．以降，2.2 節で定義を行った抽象マシン・モデル名のみを使用した場合には TLSP アーキテクチャを表す SP-CD-MF マシンに対して当該抽象マシン・モデルを適用した場合であることを前提として議論を行っていく．

ベンチマークプログラムとして SPEC CPU2000 整数ベンチマークの中から**表 3** に示すプログラムと入力を使用する．評価は実行開始から 5,000 万命令までの実行のトレースを使用し，シミュレーションにより行う．シミュレータは Lam らが評価に使用したシミュレータを本稿の評価に使用できるように改良したものである．トレースの作成には SimpleScalar バージョン 2.0[6] およびその付加プログラムである Data Value Predictors を，本稿の評価に必要な改良を行って使用する．

3.1 評価アルゴリズム

ベンチマークプログラム上のある命令の実行は，制御依存に関しては 2.1 節で定義した抽象マシン・モデル SP-CD-MF のアルゴリズムに従い，データ依存に関しては 2.2 節で定義した抽象マシン・モデルの各アルゴリズムに従う．各命令は，制御依存およびデータ依存する命令の実行が完了した時刻に実行の開始が可能となり，その時刻を各命令の実行時刻として記録する．すべての命令は 1 クロックサイクルのレイテンシで実行され，これにはメモリ・アクセスを行う命令も含まれる．本稿の評価は，TLSP アーキテクチャにおいてデータ依存制約の緩和手法が並列度に与える効果の上限値を測定することが目的である．したがって，無限の命令が同時に実行可能であり，実行すべき命令はトレース中のすべての命令の中から選択されるとする．また，分岐予測および値予測を行う場合の予測の失敗によるペナルティはないものとする．

表 3 ベンチマークプログラム
Table 3 Benchmark programs.

Program	Description	Input
gzip	Compression	ref/input.random
mcf	Combinatorial Optimization	ref/inp.in
parser	Word Processing	ref/ref.in
vortex	Object-oriented Database	ref/bendian1.raw
bzip2	Compression	ref/input.graphic

評価結果として用いるのは，トレースの全命令の開始から終了までにかかる時間のシミュレーション結果をトレースの全命令数で割ることにより得られる並列度である．

3.2 プログラムの変形

プログラムは，プロシージャ・コールおよびループによる不要な依存関係が存在する．プロシージャ・コール時およびリターン時にはスタックポインタの操作を行うため，スタックポインタを媒介とした依存関係が生成される．しかしながらスタックポインタの操作はプログラム本来の意味にはまったく関係せず，プログラムの実行のために便宜的に行われる操作である．我々の評価では，スタックポインタ操作するための命令による不要な依存関係を取り除いた．同様に，ループ変数を操作するための命令による不要な依存関係を取り除いた．

3.3 分岐予測

Lam らは，分岐予測の手法として静的な分岐予測手法を使用した．その分岐予測手法とは，トレースを解析し，分岐命令ごとに分岐方向とその回数を記録することによって分岐した回数が多かった方向を予測するという手法である．この分岐予測手法により 90%程度の分岐予測成功率を達成できると Lam らは報告している．しかし，今日までに，実行時に動的に分岐予測を行うさまざまな分岐予測器が提案され，静的な分岐予測手法と比較して高い予測成功率を実現している．本稿の評価には，その中から最も一般的な構成である gshare 分岐予測器[13] を使用する．gshare 分岐予測器は，分岐命令のアドレスとグローバルな分岐の履歴を

表 4 評価結果
Table 4 Parallelism for each machine model.

	gzip	mcf	parser	vortex	bzip2	調和平均
dBASE	2.63	6.39	2.06	2.55	2.37	2.72
MD	3.62	6.89	2.52	3.27	3.52	3.55
RN	3.32	9.61	4.99	5.98	2.98	4.51
dSP	3.42	18.40	3.27	3.86	3.27	4.10
MD+dSP	4.28	21.67	6.29	9.00	3.30	5.86
RN+dSP	4.84	28.99	5.96	8.71	7.37	7.58
MD+RN	60.08	15.91	14653.38	1039.49	59.22	51.33
MD+RN+dSP	60.08	31.82	29090.15	2077.41	118.43	87.67
MD+RN+dSP (perfect)	60.08	31.82	29449.76	3575.24	118.44	87.98

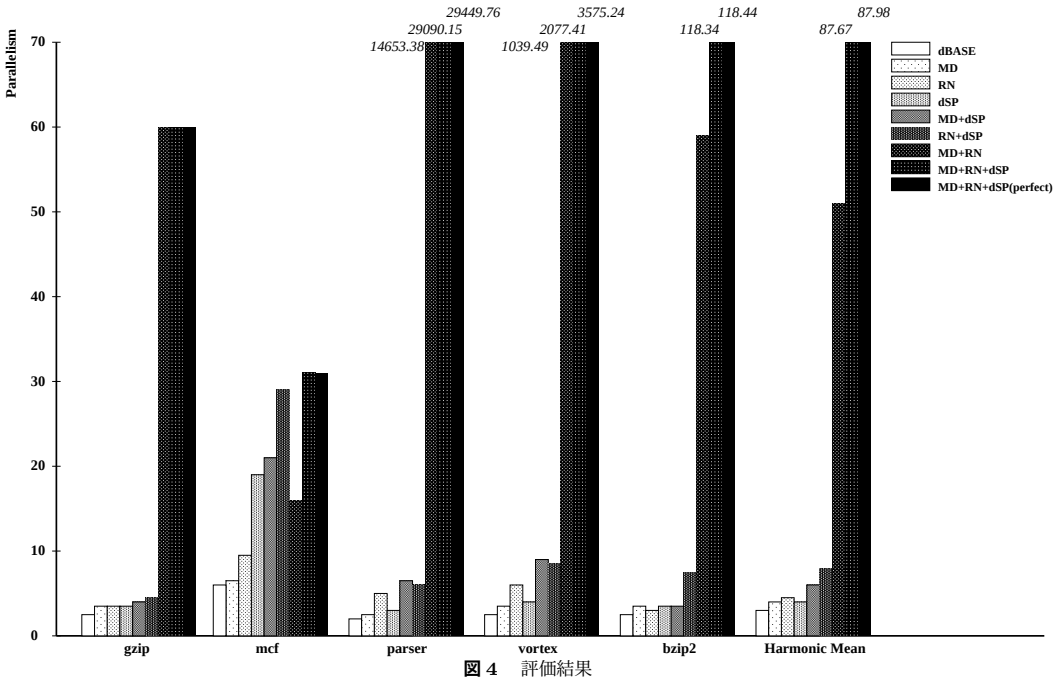

図 4 評価結果
Fig. 4 Parallelism for each machine model.

用いてパタン履歴テーブル（PHT: Pattern History Table）にアクセスし，分岐方向を予測するものである．本稿の評価では我々は PHT の容量を 128k エントリ，分岐の履歴は 12 個まで取得する構成の gshare 分岐予測器を使用する．これは，分岐予測の成功率が飽和するのに十分な大きさである[5),13)]．

3.4 値 予 測

本稿の評価では，データ依存制約の緩和のための手法として値予測を評価に使用する．値予測を実行時に動的に行う機構として，一般的で単純な構成であるストライド値予測器を使用する．ストライド値予測器は，ある命令の実行結果を予測するのに，前回の実行結果と前々回の実行結果の差分を保存しておき，それを前回の実行結果に足し合わせて予測値とするものである．本稿の評価では，命令の実行結果を格納する VHT（Value History Table）のエントリ数は 4k エントリとする．また，2 ビット飽和カウンタを用いることで値予測の信頼性を確認する．

4. 評 価 結 果

各抽象マシン・モデルを各ベンチマークプログラムに適用し，シミュレータ上で実行することにより得られた結果の並列度を**表 4** および**図 4** に示す．dBASE マシンは，TLSP アーキテクチャにおいてデータ依存制約の緩和に何の手法も使用しない場合の並列度を表し，TLSP アーキテクチャにおける基本となる．dBASE マシンは調和平均で 2.72 の並列度を持つ．

MD+RN+dSP（perfect）マシンは MD+RN+dSP

表 5 ストライド値予測器の挙動
Table 5 Result of value prediction.

	全命令数	値予測可能な命令数	値予測に成功する命令数	値予測に失敗する命令数	値予測を行わない命令数	値予測成功率	値予測失敗率
gzip	49,999,227	32,728,953	9,291,929	4,342	23,432,682	28.3%	0.0%
mcf	49,661,695	17,504,307	15,327,500	231,939	1,944,868	87.6%	1.3%
parser	45,264,275	29,988,903	11,988,277	1,875,084	16,135,542	40.0%	6.3%
vortex	46,710,540	19,235,391	12,658,705	767,207	5,809,479	65.8%	4.0%
bzip2	49,273,582	30,297,873	14,801,340	22,040	15,474,493	48.9%	0.1%

マシンにおいて，すべての値予測が成功した場合の性能を示し，その並列度は調和平均で 87.98 となった．

4.1 メモリ・アドレスの一致/不一致解析の効果

メモリ・アドレスの一致/不一致解析単体の効果は MD マシンによって表される．MD マシンは調和平均で 3.55 の並列度を持ち，これは dBASE マシンと比較して 30%程度の並列度の向上となる．しかしながら他の 2 手法（RN，dSP）単体で得ることのできる並列度と比較して少ない並列度しか得ることができていない．しかしながら，gzip および bzip2 では MD マシンが RN マシンより高い並列度を得ている．

4.2 リネーミングの効果

RN マシンが行うことのできるのはレジスタのリネーミングのみとなる．なぜなら，メモリ・アドレスの一致/不一致解析が行われなければメモリのリネーミングの有無にかかわらずメモリ・アクセスがインオーダに行われるからである．したがって RN マシンが示すのはレジスタ・リネーミングのみで得ることのできる並列度でありその調和平均は 4.51 である．これは 1 種類のデータ依存制約の緩和手法で得ることのできる並列度では最も高い並列度であり，最も低い並列度の MD マシンと比較して約 27%の並列度の向上がみられる．dBASE マシンと比較すると 66%もの並列度の向上であり，TLSP アーキテクチャの性能は逆依存や出力依存制約により大きく制限されることが分かる．

MD+RN マシンではメモリ・アクセスのインオーダ実行の制限がなくなるためにリネーミングの効果はメモリ・アクセスにおいても有効となる．MD+RN マシンは MD+RN+dSP を除くすべての中で最も高い並列度の 51.33 を示し，これは MD+dSP マシンや RN+dSP マシンと比較しても 6 倍以上の数値である．また，MD マシンと比較すると MD+RN マシンは 10 倍以上の並列度を示し，MD 単体や RN 単体の並列度と比較して非常に高いことが分かる．このことから，メモリ・アクセスにおけるリネーミングによる効果はレジスタ・アクセスにおけるリネーミング以上に高いことが分かる．

4.3 値予測の効果

dSP マシンによって示されるのは TLSP アーキテクチャにおいてフロー依存制約により制限されていた性能が値予測の使用にる緩和の可能性である．dSP マシンの並列度は 4.10 であり，これは 1 つのデータ依存制約の緩和手法により得られる並列度の中で 2 番目に大きい数字であり，dBASE マシンから 50%の向上がみられる．このことから TLSP アーキテクチャにおけるフロー依存が多く存在し，それは値予測によって大幅に緩和することが可能であるということを示している．

値予測を他のデータ依存制約の緩和手法と組み合わせた場合，RN+dSP マシンでは RN マシンと比較して 68%ものの並列度の向上がみられ，MD+dSP マシンでは MD マシンと比較して 65%もの並列度の向上がみられる．このことから，値予測は他のデータ依存制約の緩和手法との組合せにかかわらず，ある一定の並列度の向上の効果が期待できるということが分かる．

MD+RN+dSP マシンは，値予測を行わない MD+RN マシンと比較して 69%もの並列度の向上がみられる．このことから，すでに高い並列度を得ている TLSP アーキテクチャにおける MD+RN マシンにおいて，値予測によってさらに大幅な並列度の向上が期待できる．特に，vortex を除くプログラムでは MD+RN+dSP マシンは MD+RN+dSP（perfect）マシンで得ることのできる並列度とほぼ同じ並列度を得ることが可能となっており，今回使用したストライド値予測器により十分な効果が得られることが確認できる．また，vortex では MD+RN+dSP マシンにより得られる並列度には MD+RN+dSP（perfect）マシンと比較するとまだ並列度の向上の余地があり，値予測器の予測成功率の向上による並列度の向上が期待できる．

本評価における値予測の結果について**表 5** に示す．ここで値予測成功率は値予測に成功した命令数が値予測対象の命令数に占める割合を示し，値予測失敗率は値予測に失敗する命令数が値予測対象の命令数に占める割合を示す．値予測を行わない場合とは，VHT に当該命令に対応するエントリが存在しなかった場合と

2ビット飽和カウンタの条件が値予測を行うのに十分でなかった場合である．

値予測が非常に高い成功率である2つのプログラムmcfとvortexは対照的な結果を残している．mcfは値予測の高い成功率を背景にdSPを含む場合に非常に高い並列度を達成しているが，vortexは値予測の高い成功率にもかかわらずdSPを含む場合でもそれほど高い並列度を達成するわけではない．また，vortexは値予測の成功率の高さと比較してMD+RN+dSP（perfect）マシンとMD+RN+dSPマシンとの間の並列度の差が大きい．

値予測の失敗率が低いプログラムとしてgzipとbzip2がある．これらはともに1%以下の失敗率を示している．この非常に低い値予測の失敗率に対して，値予測の成功率はそれぞれ28.3%，48.9%と低くとどまっている．gzipはMD+RN+dSPマシンにおける値予測の効果が非常に低く，失敗率は低いものの値予測を行う価値は低いといえる．しかし，bzip2ではMD+RN+dSPマシンの値予測の効果が非常に高く，かつ値予測の失敗率が非常に低いため，値予測を行う価値が非常に高いといえる．

値予測の効果については，文献17)において報告がなされている．文献17)によると，本稿で使用したものと同様のストライド値予測器の予測成功率は40〜80%程度であると報告しており，本稿の値予測の結果にほぼ一致する．また，文献12)ではTLSPアーキテクチャにおいてスレッド間のレジスタおよびメモリの同期において値予測がすべて成功する場合に40%程度の並列度の向上がみられると報告しており，これは我々の評価においても値予測を行うマシンが値予測を行わないマシンと比較して60〜70%程度の並列度の向上がみられることと比較して，評価モデルの違い等を考慮に入れると妥当なものである．

5. おわりに

本稿ではTLSPアーキテクチャにおいてデータ依存制約の緩和手法が並列度に与える効果について実験を行い検証した．その結果，メモリ・アドレスの一致/不一致解析を行うことでリネーミングの効果を大幅に向上させることができることが判明した．また，値予測は，すでに他のデータ依存制約の緩和手法のすべてを使用して高い並列度が得られている場合に適用しても，さらに69%もの並列度の向上がみられた．また，すべての値予測が成功した場合の並列度は，通常に値予測における並列度と比較して大幅な向上はみられず，現状の値予測の効果が非常に高いことが判明した．

今後は，今回の実験により得られた結果と，これまでに提案されてきたTLSPアーキテクチャにより得られる並列度の間に存在する大きな乖離の原因の考察を行っていく予定である．

謝辞 本研究は一部，日本学術振興会科学研究費基盤研究（A）（2）「システムLSI向けカスタム化可能IPコアのアーキテクチャおよび設計支援技術の開発」（課題番号12358002），日本学術振興会科学研究費基盤研究（A）（2）「ハードウェア構成を動的に最適化する『ハードウェア・モーフィング』技術の開発」（課題番号13308015）による．

参考文献

1) 朝生良教，高田 滋，森俊一郎，清水健士，林達也：マルチスーパースカラパイプラインによるブロック並列実行方式，情報処理学会アーキテクチャ研究会報告，I19-11 (1996).
2) 小林良太郎，岩田充晃，安藤秀樹，島田俊夫：制御依存解析複数命流実行を導入した投機的実行機構の提案と予備的評価，情報処理学会アーキテクチャ研究会報告，I25-24 (1996).
3) 玉造潤史，松本 尚，平木 敬：On Chip MIMDにおける大規模投機的実行機構，情報処理学会アーキテクチャ研究会報告，I19-11 (1996).
4) 鳥居 淳，近藤真己，本村真人，池野晃久，小長谷明彦，西 直樹：オンチップ制御並列プロセッサMUSCATの提案，情報処理学会論文誌，Vol.39, No.6, pp.1622–1631 (1998).
5) 野口良太，森 敦司，小林良太郎，安藤秀樹，島田俊夫：分岐方向の偏りを利用し破壊的競合を低減する分岐予測方式，情報処理学会論文誌，Vol.40, No.5 (1999).
6) Burger, D. and Austin, T.M.: The SimpleScalar Tool Set, Version 2.0, University of Wisconsin-Madison Computer Sciences Department Technical Report, #1342 (June 1997).
7) Dubey, P.K., O'Brien, K., O'Brien, K.M. and Barton, C.: Single-Program Speculative Multithreading (SPSM) Architecture: Compiler-assisted Fine-Grained Multithreading, *1st International Conference on Parallel Architectures and Compilation Techniques*, pp.109–121 (June 1995).
8) Hammond, L., Hubbert, B., Siu, M., Prabhu, M., Chen, M. and Olukotun, K.: The Stanford Hydra CMP, *IEEE MICRO Magazine*, pp.71–84 (March-April 2000).
9) Kemp, G.A. and Franklin, M.: PEWs: A Decentralized Dynamic Scheduler for ILP Processing, *International Conference on Parallel Processing*, Vol.1, pp.239–246 (Aug. 1996).

10) Krishnan, V. and Torrellas, J.: Hardware and Software Support for Speculative Execution of Sequential Binaries on a Chip-Multiprocessor, *12th International Conference on Supercomputing* (July 1998).
11) Lam, M.S. and Wilson, R.P.: Limits of Control Flow on Parallelism, *19th International Symposium on Computer Architecture*, pp.46–57 (June 1992).
12) Marcuello, P., Tubella, J. and González, A.: Value Prediction for Speculative Multithreaded Architectures, *32nd International Symposium on Microarchitecture*, pp.230–237 (Dec. 1999).
13) McFarling, S.: Combining Branch Predictors, *WRL Technical Note*, TN–36 (June 1993).
14) Nishi, N., et al.: A 1GIPS 1W Single-Chip Tightly-Coupled Four-Way Multiprocessor with Architecture Support for Multiple Control Flow Execution, *International Solid-State Circuits Conference* (Feb. 2000).
15) Sohi, G.S., Breach, S.E. and Vijaykumar, T.N.: Multiscalar Processors, *22nd International Symposium on Computer Architecture*, pp.414–425 (June 1995).
16) Wall, D.W.: Limits of Instruction-Level Parallelism, *4th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp.177–188 (Apr. 1991).
17) Wang, K. and Franklin, M.: Highly Accurate Data Value Prediction using Hybrid Predictors, *30th International Symposium on Microarchitecture*, pp.281–290 (Dec. 1997).
18) Yeh, T.Y. and Patt, Y.N.: A Comparison of Dynamic Branch Predictors that use Two Level of Branch History, *20th International Symposium on Computer Architecture*, pp.257–266 (May 1993).

(平成 14 年 1 月 25 日受付)
(平成 14 年 5 月 5 日採録)

目次 勝彦（学生会員）

1974 年生．1998 年九州大学工学部情報工学科卒業．2000 年同大学大学院システム情報科学研究科情報工学専攻修士課程修了．現在，同大学院システム情報科学府情報理学専攻博士後期課程に在学中．計算機アーキテクチャの研究に従事．

村上 和彰（正会員）

1960 年生．1982 年京都大学工学部情報工学科卒業．1984 年同大学大学院修士課程修了．同年富士通（株）本体事業部に入社，汎用計算機 M シリーズのアーキテクチャ開発に従事．1987 年九州大学工学部助手，1992 年同大学大学院総合理工学研究科講師，1994 年同助教授，2000 年九州大学大学院システム情報科学研究院情報理学部門教授，現在に至る．工学博士．電子情報通信学会，日本応用数理学会，ACM，IEEE，IEEE-CS 各会員．