九州大学学術情報リポジトリ
Kyushu University Institutional Repository

制御依存およびデータ依存制約を緩和することの効果 : Lam and Wilsonの評価結果の改訂

目次, 勝彦
九州大学大学院システム情報科学府情報工学専攻

村上, 和彰
九州大学大学院システム情報科学研究院情報工学部門

https://hdl.handle.net/2324/7566

出版情報：電子情報通信学会技術研究報告. CPSY, コンピュータシステム. 100 (248), pp.73-80, 2000-07-26. 電子情報通信学会
バージョン：
権利関係：

KYUSHU UNIVERSITY

制御依存およびデータ依存制約を緩和することの効果 －Lam and Wilsonの評価結果の改訂－

目次 勝彦† 村上 和彰‡

†九州大学大学院 システム情報科学府 情報工学専攻
‡九州大学大学院 システム情報科学研究院 情報工学部門
〒 816-8580 春日市春日公園 6-1
Email: ppram@c.csce.kyushu-u.ac.jp

あらまし 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．従来は依存解析範囲の問題から命令間のデータ依存の解析が重視されていたが，LSI の高集積化と大規模化により同時実行可能な命令数が増加したため広範囲の命令間の依存解析による並列度の向上が求められている．本論文ではデータ依存と制御依存による実行時の制約を緩和するためのの対処法を示し，その組合せの違いが並列度に与える影響を検証した．

キーワード 制御依存，データ依存，並列実行

Effects of Relaxing Constraints due to Control and Data Dependences - Updating the Result of Lam and Wilson's -

Katsuhiko METSUGI Kazuaki MURAKAMI

Department of Computer Science and Communication Engineering, Kyushu University,
6–1 Kasuga-koen, Kasuga, Fukuoka 816-8580 Japan.
Email: ppram@c.csce.kyushu-u.ac.jp

Abstract Two fundamental restrictions that limit the amount of instruction level parallelism extracted from sequential programs are control flow and data flow. In the past processors mainly use parallelism obtained by analyzing data flow dependences. But the increased density of transistor and the enlargement of LSI enable us to execute more instructions simultaneously and claim us to analyze larger area of instcutions for more parallelism. This paper describes the techniques to relax the constraints imposed by data dependence and define the abstract machine models using these techniques. Final destination of this work is to evaluate the effect of these models.

Key Words control dependence, data dependence, parallel execution

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 and Wilson は 1992 年の ISCA で発表した論文 “Limits of Control Flow on Parallelism”[1] 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**」に関する研究が盛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．

- UW-Madison の Multiscalar[2]
- NEC の MUSCAT[8](Merlot)
- Stanford 大の Hydra[3]
- 名古屋大学の SKY[9]
- UIUC の Speculative CMP (Chip MultiProcessor)[4]
- CMU の TLDS (Thread-Level Data Speculation)[5]
- 東京大学の OCHA-Pro[10]

上記のうち，MUSCAT は MP98[6] として既に実用化されている．

しかしながら，Lam and Wilson の評価 [1]，ならびに，これら TLSP に関する研究は次の課題を抱えている．

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見られる (表 1参照)．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，メモリに関わらず，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予め判明している」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如何で性能が異なる．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，その強力さにも関わらず，本手法が並列度に与える効果の上限値に関する評価は未だ成されていない．

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 の解決を試みる．以下，2章で制御依存への対処法を，また，3章でデータ依存への対処法をそれぞれ整理する．そして，

表 1: 性能比較

		awk	ccom	compress	eqntott	espresso	gcc	go	idct	irsim	latex	li	vortex	調和平均
Lam and Wilson[1] <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
Multiscalar [2] <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 [8] <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[3] <性能向上比>	4PEs			1.00	0.58							1.04	0.62	
SKY[9] <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

4章で評価対象とする抽象マシン・モデルすべてを定義し，5章で現在構築している評価環境について述べる．

2 制御依存制約の緩和法とその抽象マシン・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].

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得る．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) することが出来れば，制御依存に起因する制約を緩和することが可能である．

本稿でも，Lam and Wilson の例にならって，下記の「抽象マシン・モデル (abstract machine model)」を定義する (ただし，表記法は若干異なる).

cBASE: ある命令の実行は，それに先行するすべての分岐命令の実行が完了するまで行えない．

CD: ある命令の実行は，それが制御依存しているすべての分岐命令の実行が完了するまで行えない．また，分岐命令はイン・オーダーに，かつ，同時には高々 1 個しか実行できない．

CD+MF: ある命令の実行は，それが制御依存しているすべての分岐命令の実行が完了するまで行えない．分岐命令に関しては CD モデルと異なり，複数の分岐命令を同時に，かつ，アウト・オブ・オーダーに実行してもよい．

cSP: ある命令の実行は，それに先行するすべての「分岐予測に失敗した分岐命令」の実行が完了するまで行えない．言い換えると，ある命令に注目したとき，それに先行する分岐命令の分岐予測が成功している限り，当該命令の実行はこれら分岐命令の存在により何ら妨げられることはない．分岐命令に関して言い換えると，ある分岐命令の実行は，それに先行するすべての「分岐予測に失敗した分岐命令」の実行が完了するまでは行えない．

cSP+CD: ある命令の実行は，それが制御依存しているすべての「分岐予測に失敗した分岐命令」

の実行が完了するまで行えない．分岐命令に関しては，それに先行するすべての「分岐予測に失敗した分岐命令」の実行が完了するまで行えない．

cSP+CD+MF: ある命令の実行は，それが制御依存しているすべての「分岐予測に失敗した分岐命令」の実行が完了するまで行えない．分岐命令もこれと同様．

cORACLE: 制御依存に起因する制約は一切ない．

上記 7 モデルの「制御依存制約に対する緩和効果の強さ」に関する二項関係をまとめると，図 1に示すような半順序関係となる．図 2(a) に示す制御依存グラフを有するプログラムに対して図 2(b) のように実行時に制御が流れた場合，各モデル上では図 3に示すように命令が実行されることになる．

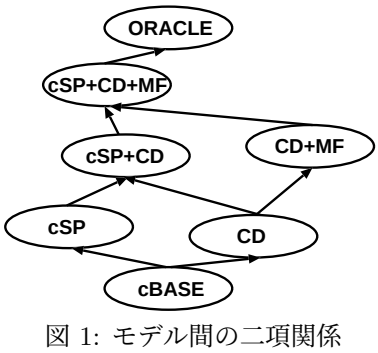

図 1: モデル間の二項関係

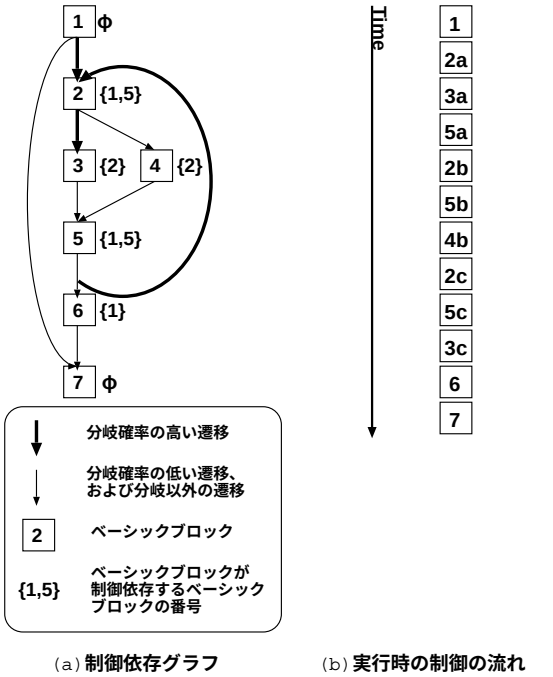

図 2: 制御依存グラフと実行時の制御の流れ

3 データ依存制約の緩和法とその抽象マシン・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／不一致が予め解析できれば，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取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．

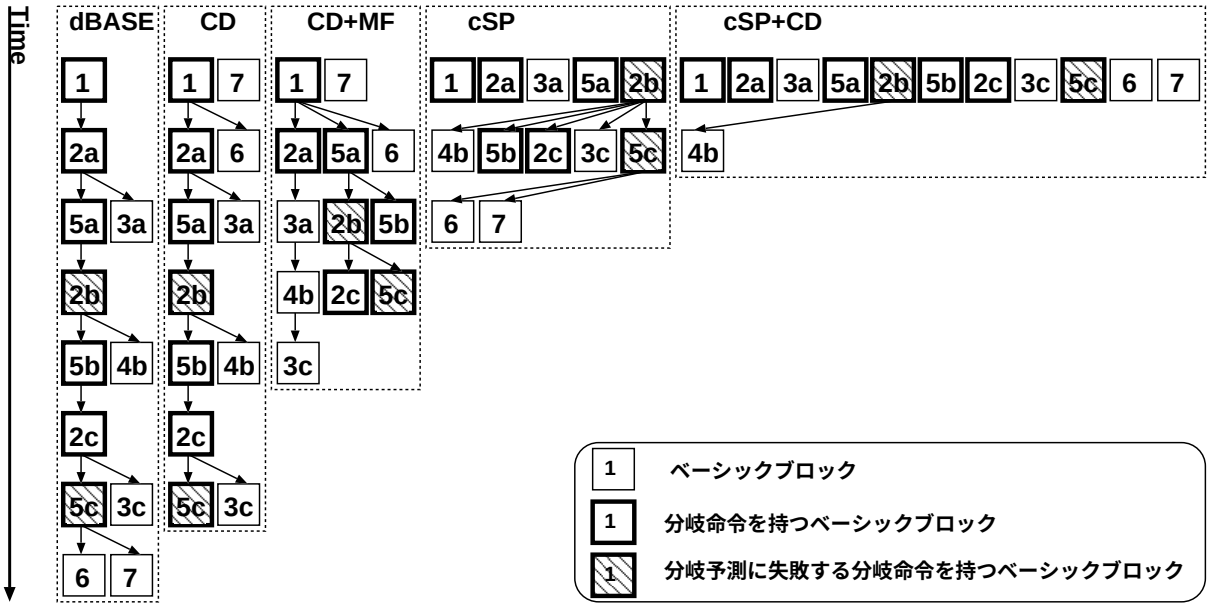

図 3: 各モデル上での命令実行の様子

制御依存制約の緩和法同様，データ依存制約の緩和法に対しても下記の「抽象マシン・モデル (abstract machine model)」を定義する．

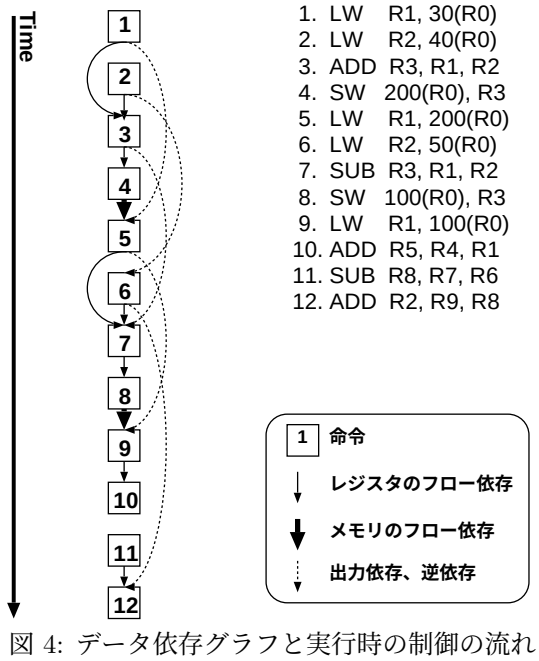

図 4: データ依存グラフと実行時の制御の流れ

dBASE: ある命令の実行は，それがデータ依存しているすべての命令の実行が完了するまで行えない．さらに，ロード／ストア命令に関しては，それに先行するすべてのロード／ストア命令の実行が完了するまでその実行を行えない．

RN: ある命令の実行は，それがフロー依存しているすべての命令の実行が完了するまで行えない．さらに，ロード／ストア命令に関しては，それに先行するすべてのロード／ストア命令の実行が完了するまでその実行を行えない．

RN+dSP: ある命令の実行は，それがフロー依存しているすべての「値予測に失敗した命令」の実行が完了するまで行えない．さらに，ロード／ストア命令に関しては，それに先行するすべてのロード／ストア命令の実行が完了するまでその実行を行えない．

MD: ある命令の実行は，それがデータ依存しているすべての命令の実行が完了するまで行えない．ロード／ストア命令も同様．

MD+RN: ある命令の実行は，それがフロー依存しているすべての命令の実行が完了するまで行えない．ロード／ストア命令も同様．Lam and Wilson の評価 [1] で用いられたモデル．

MD+RN+dSP: ある命令の実行は，それがフロー依存しているすべての「値予測に失敗した命令」の実行が完了するまで行えない．ロード／ストア命令も同様．

dORACLE: データ依存に起因する制約は一切ない．

上記 7 モデルの「データ依存制約に対する緩和効果の強さ」に関する二項関係をまとめると，図 5に示すような半順序関係となる．図 4(a) に示すデータ依存グラフを有するプログラムに対して図 4(b) のように実行時に制御が流れた場合，各モデル上では図 6に示すように命令が実行されることになる．

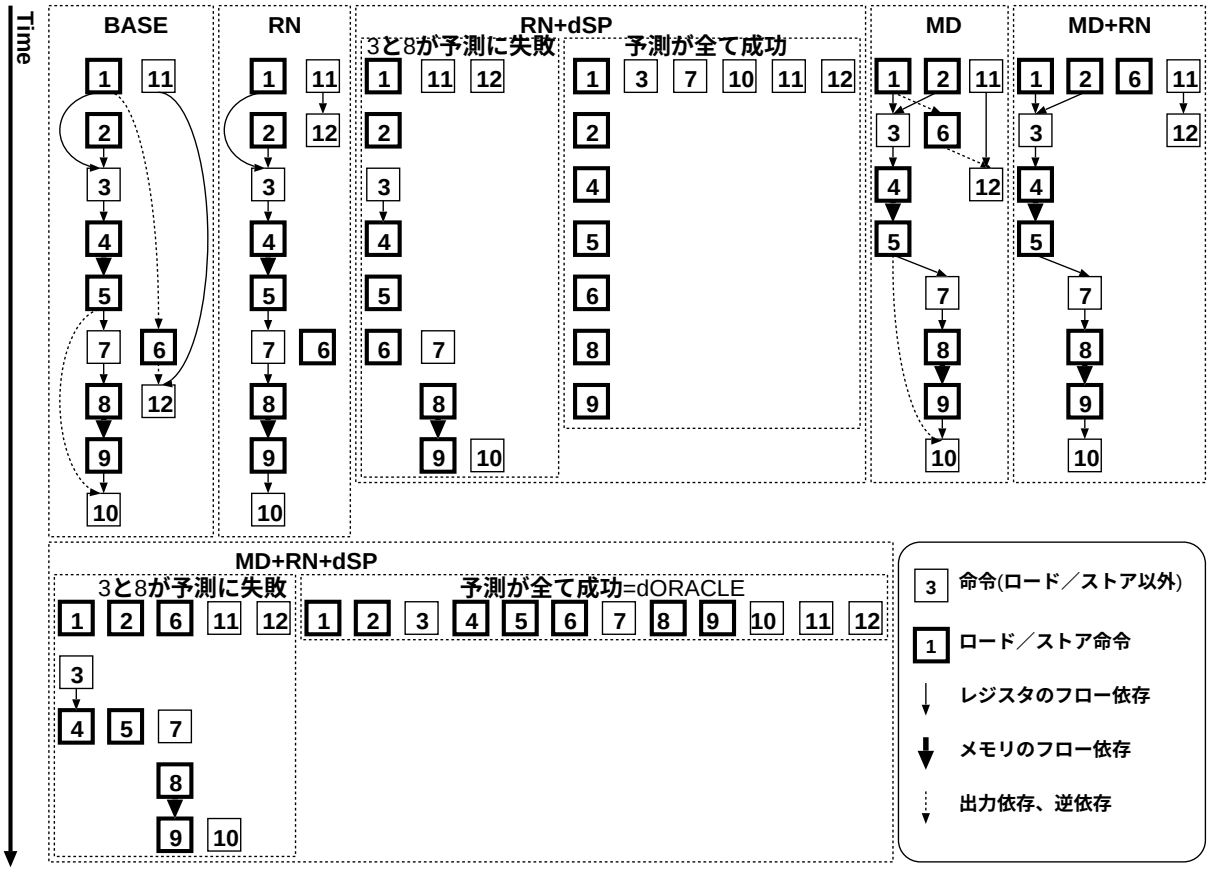

図 6: 各モデル上での命令実行の様子

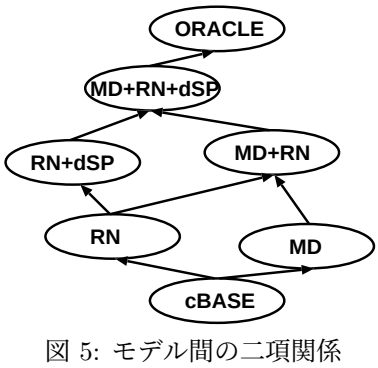

図 5: モデル間の二項関係

4 評価空間

2章および 3章で定義したそれぞれの抽象マシン・モデル集合は互いに直交関係にあるので，表 2に示すように計 49(=7 × 7) 通りのモデルの組み合わせが可能である．各欄の上段にはモデル名を，中段にはそのモデルに対応する具体的なアーキテクチャ名を，そして下段には当該モデル上で得られる並列度を記している．現時点では，Lam and Wilson の評価結果 [1] のみを記している．我々の当面の目標は，表 2のすべての欄に対して，対応するモデル上で得られる並列度を求めることである．

5 評価環境

表 2で定義した各モデル上で得られる並列度を測定するために，Lam and Wilson[1] と同様の方法論に基づいて下記の評価環境を構築している．

- **アドレストレースの作成：**シミュレータの入力として必要となるアドレストレースの作成環境を構築する．QPT[7] により，SPARC アーキテクチャ向けにコンパイルされた実行ファイルのアドレストレースが可能である．QPT により作成されたデータはシミュレータの入力に適した形式への変換を行う．

- **ベーシックブロックのアドレスおよび実行回数の測定：**QPT により作成されたアドレストレースを基に，ベーシックブロックの先頭アドレスおよび実行回数を測定するプログラムを作成する．作成されたデータはシミュレータの入力として使用される．

- **トレース駆動シミュレーション：**トレース駆動型シミュレータを使用することで，命令間およびベーシックブロック間の静的な制御依存解析

表 2: 評価空間

	dBASE	RN	RN+dSP	MD
cBASE	(cBASE) × (dBASE)	(cBASE) × (RN)	(cBASE) × (RN+dSP)	(cBASE) × (MD)
CD	(CD) × (dBASE)	(CD) × (RN)	(CD) × (RN+dSP)	(CD) × (MD)
CD+MF	(CD+MF) × (dBASE)	(CD+MF) × (RN)	(CD+MF) × (RN+dSP)	(CD+MF) × (MD)
cSP	(cSP) × (dBASE)	(cSP) × (RN)	(cSP) × (RN+dSP)	(cSP) × (MD)
cSP+CD	(cSP+CD) × (dBASE)	(cSP+CD) × (RN)	(cSP+CD) × (RN+dSP)	(cSP+CD) × (MD)
cSP+CD+MF	(cSP+CD+MF) × (dBASE)	(cSP+CD+MF) × (RN)	(cSP+CD+MF) × (RN+dSP)	(cSP+CD+MF) × (MD)
cORACLE	(cORACLE) × (dBASE)	(cORACLE) × (RN)	(cORACLE) × (RN+dSP)	(cORACLE) × (MD)

	MD+RN	MD+RN+dSP	dORACLE
cBASE	(cBASE) × (MD+RN)	(cBASE) × (MD+RN+dSP)	(cBASE) × (dORACLE)
	2.14		
CD	(CD) × (MD+RN)	(CD) × (MD+RN+dSP)	(CD) × (dORACLE)
	2.39		
CD+MF	(CD+MF) × (MD+RN)	(CD+MF) × (MD+RN+dSP)	(CD+MF) × (dORACLE)
	6.96		
cSP	(cSP) × (MD+RN)	(cSP) × (MD+RN+dSP)	(cSP) × (dORACLE)
	ILSP	ILDSP	
	6.80		
cSP+CD	(cSP+CD) × (MD+RN)	(cSP+CD) × (MD+RN+dSP)	(cSP+CD) × (dORACLE)
	13.27		
cSP+CD+MF	(cSP+CD+MF) × (MD+RN)	(cSP+CD+MF) × (MD+RN+dSP)	(cSP+CD+MF) × (dORACLE)
	TLSP	TLDSP	
	39.62		
cORACLE	(cORACLE) × (MD+RN)	(cORACLE) × (MD+RN+dSP)	(cORACLE) × (dORACLE)
	158.26		

ILSP: Instruction-Level Speculative Parallel processor
ILDSP: Instruction-Level Data-Speculative Parallel processor
TLSP: Thread-Level Speculative Parallel processor
TLDSP: Thread-Level Data-Speculative Parallel processor

とデータ依存解析が可能となる．シミュレータは SPARC アーキテクチャ向けにコンパイルされた実行ファイルおよびその実行ファイルのアドレストレース，ベーシックブロックのアドレスとその実行回数を入力とし，表 2の各モデルに対応した並列度を出力する．

6 おわりに

本稿では，プログラムが本質的に有する制御依存およびデータ依存による制約を緩和する手法を利用するスレッドレベル投機的並列処理 (TLSP) アーキテクチャに関する従来の研究が残した課題を述べ，その解決手法として制御依存およびデータ依存それぞれに関する制約の緩和手法のモデル化を行った．また，モデル化を行った手法の効果を測定するために構築している評価環境について述べた．

今後は評価環境を完成させ，モデル化を行った手法の有効性の実証とともに残された課題 1 について検証を行う．

謝辞

日頃から御討論頂く，九州大学大学院システム情報科学研究院 安浦寛人教授，岩井原瑞穂助教授，ならびに，安浦・村上・岩井原研究室の諸氏に感謝する．

本研究は一部，文部省科学研究費補助金基盤研究 (A)(2) 一般研究「スケーラブル・システム LSI アーキテクチャの設計手法に関する研究」(課題番号: 11308011) および文部省科学研究費補助金基盤研究 (A)(2) 展開

研究「システム LSI 向きカスタム化可能 IP コアのアーキテクチャおよび設計支援環境の開発」(課題番号: 12358002) による．

参考文献

[1] Lam, M. S. and Wilson, R. P., “Limits of Control Flow on Parallelism,” *19th International Symposium on Computer Architecture(ISCA-19)*, 1992.

[2] Sohi, G. S., Breach, S. E. and Vijaykumar, T. N., “Multiscalar Processors,” *22th International Symposium on Computer Architecture(ISCA-22)*, 1995.

[3] Hammond, L., Hubbert, B., Siu, M., Prabhu, M., Chen, M. and Olukotun, K., “The Stanford Hydra CMP,” *IEEE MICRO Magazine*, March-April 2000.

[4] Krishnan, V. and Torrellas, J., “Hardware and Software Support for Speculative Execution of Sequential Binaries on a Chip-Multiprocessor,” *12th International Conference on Supercomputing (ICS)*, July 1998.

[5] Steffan, J. G., and Mowry, T. C., “The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization,” *4th International Symposium on High-Performance Computer Architecture*, February 1998.

[6] Nishi, N., Inoue, T., Nomura, M., Matsushita, S., Torii, S., Shibayama, A., Sakai, J., Ohsawa, T., Nakamura, Y., Shimada, S., Ito, Y., Edahiro, M., Minami, K., Matsuo, O., Inoue, H., Manabe, T., Horiuchi, T., Motomura, M., Yamashina, M. and Fukuma, M., “A 1GIPS 1W Single-Chip Tightly-Coupled Four-Way Multiprocessor with Architecture Support for Multiple Control Flow Execution,” *International Solid-State Circuits Conference*, February 2000.

[7] Hill, M. D., Larus, J. R., Lebeck, A. R., Talluri, M. and Wood, D. A., “WARTS: Wisconsin Architectural Research Tool Set,” *http://www.cs.wisc.edu/~larus/warts.html*, University of Wisconsin - Madison.

[8] 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 年 5 月.

[9] 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.

[10] 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.