計算機アーキテクチャ 139－20
（２０００．８．４）

オンチップマルチプロセッサにおける命令フェッチ方式

嶋田幸子　鳥居淳　松下智　鈴木研司†　西直樹

NEC システムデバイス・基礎研究本部　†NEC 情報システムズ

Merlot(MP98 Ver.1) における命令フェッチ方式を提案する。細粒度で並列化されたプログラムの実行においてキャッシュのヒット率の向上と容量を確保するため、Merlot では単ポート共有命令キャッシュを使用している。各プロセッサエレメントに十分な命令供給能力を得るため、命令プリフェッチ、分岐予測の前倒しを可能とする機能の他、PE 毎命令アラインバッファをミニキャッシュとしても利用できるよう実装し、命令キャッシュ競合を低減している。パイプラインシミュレータを用いた評価により、理想的な構成である 4 ポート共有キャッシュに対して本方式では 98 %の性能となり、本方式の有効性が確かめられた。

Instruction Fetch Mechanism for On-chip Multi Processor

Sachiko Shimada　Sunao Torii　Satoshi Matsushita
Kenji Suzuki†　Naoki Nishi

NEC System Devices and Fundamental Research　†NEC Infomatec Systems,Ltd.

We have developed an on-chip multi-processor "Merlot(MP98 Ver.1)". Merlot adopts a single-port shared instruction cache to reduce area impact while achieving high instruction cache hit ratio. Each processing element(PE) fetchs 32byte-instruction data to instruction align buffers(ALBs) at one cycle, to satisfy average instruction consumption by 2 way superscalar issue on each PE. To reduce instruction bubble, instruction cache lines are prefetched as soon as ether set of ALBs becomes empty. Moreover, Merlot provides third decode slot to remove branch bubble. The ALBs also work as a mini cache system to decrease the number of instruction cache accesses. Simulation results show that Merlot's performance is about 98 % compared to an ideal 4-ports cache.

1　はじめに

近年のハイエンドマイクロプロセッサはスーパースカラや VLIW などの技術により命令レベルの並列化を行っている。しかしながら、これらの技術による並列化にはデータ依存などによる限界が見えて来ている。一方オフチップにおけるマルチプロセッサではプロセッサ間の通信コストが大きいためスレッドの粒度を細かくして並列化を行うことが難しい。これに対し我々の提案してきた MP98 アーキテクチャ [2] ではオンチップマルチプロセッサにおける PE 間の低い通信コストを有効に活用し、細粒度の命令スレッドレベルで並列化を行いより多くの並列性を用いることにより性能向上を実現している。

オンチップマルチプロセッサにおける命令キャッシュの構成は、プロセッサ毎にキャッシュを持つ分散命令キャッシュと、オンチップであることを利用した共有命令キャッシュの二種類が考えられる。MP98 アーキテクチャではスレッドの粒度が細かいため、1 チップ上の複数のプロセッサエレメント (PE) 間でキャッシュを共有することによりヒット率の向上が期待できる。また他の PE により先行してキャッシュリフィルが行われるためキャッシュミスペナルティを隠蔽できる。このため Merlot では共有命令キャッシュを選択し、更に以下の点を目標として命令キャッシュの設計を行った。

- 実装コストを削減しつつ十分な容量を確保する。
- 各 PE に十分な命令供給能力を確保する。
- 消費電力低減のため，キャッシュアクセス回数を低減する。

本報告では Merlot における命令キャッシュ及び命令フェッチ方式について述べ、パイプラインシミュレータによる性能評価結果についてまとめる。

2　関連研究

分散命令キャッシュを用いたオンチップマルチプロセッサ、あるいはスレッドの並列化を行うマルチスレッドプロセッサの研究として Olukotum ら [5], Sohi ら [6] の報告がある。これは命令キャッシュがデータキャッシュに比べ小容量でも高いヒット率を稼げること、共有命令キャッシュではアクセス競合の点で問題があるとの見解からである。

一方 Nayfeh ら [7] をはじめ共有命令キャッシュを

提唱する報告もある。これは十分に並列なスレッドが存在しない場合分散命令キャッシュではキャッシュがPE独立に存在するため使用されないものが生じること、プロセッサ間の交信度が高いプログラムを複数プロセッサで並列実行した場合や単一プログラムの並列実行を行う場合には命令キャッシュをPE間で共有するとヒット率が向上することが理由である。

また平田ら[8]は単ポート共有命令キャッシュを用い、各PE個別に十分な容量の命令バッファを持たせることを提案している。命令バッファの容量はPE発行命令数 × PE数を想定している。更に命令キャッシュアクセス競合が生じた際、複数PEからのアクセス要求アドレスが同一であればその要求をマージし、同時に複数PEへのフェッチを実現とすることにより命令バッファの容量削減が可能であると報告している。

3 Merlot 仕様

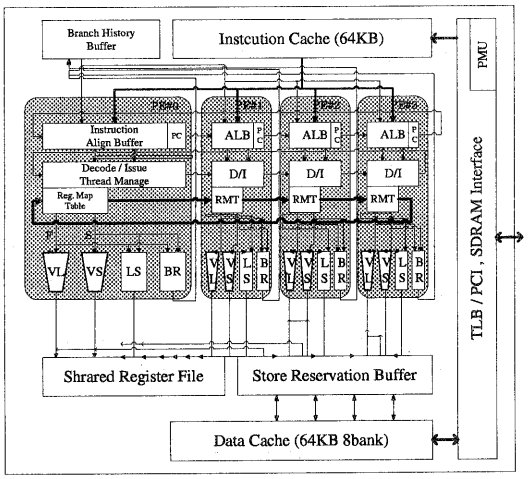

図 1: Merlot ブロック図

表 1: Merlot 仕様

命令セット	2B/4B 可変長命令
スーパースカラ度	2 命令同時発行 / 完了 (in-order)
PE数	4PE
キャッシュ	I/D とも 4PE 間共有,64kB, 4way set-associative
演算パイプライン	PE 毎に ALU×2, L/S パイプ ×1, 分岐処理 ×1

Merlot の仕様を表 1についてまとめた。1チップに 2way スーパースカラの PE を 4 台集積し、命令キャッシュの他、BHB、データキャッシュ、レジスタファイル等を PE 間共有のリソースとして持つ。パイプラインステージは 8 ステージであり、各ステージでは表 2に示す処理を行う。

表 2: パイプラインステージ

IS	命令キャッシュスケジューリング
IF	命令フェッチ
IA	命令アライメント、分岐アドレス計算
DM	デコード、命令発行
RA	レジスタアクセス
E1	実行ステージ 1
E2	実行ステージ 2
WB	ライトバック

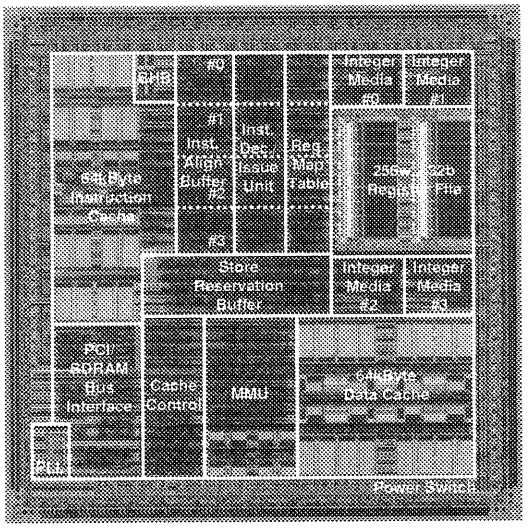

図 2: Merlot die プロット図

4 Merlot 命令フェッチ方式

MP98 アーキテクチャではスレッド粒度が細かいため、ヒット率の向上が期待できる共有命令キャッシュを使用することが優位であると考えた。また配線増加により集積度が悪化すると十分な容量の命令キャッシュが実現できないため単ポートで実装することとした。

さらに Merlot では単ポート命令キャッシュを用いつつ各 PE に十分な命令供給を行うため、以下の機能を実装している。

- 命令フェッチ幅として4サイクルに1回のフェッチで十分な命令数である8命令分(32B)を用意する。
- フェッチ幅に等しい32Bの命令アラインバッファ(ALB)をPE個別に2セットずつ用意する。
- 分岐時のフェッチ要求前倒しを行うため、2wayスーパースカラの発行系に対し命令スロットを3組用意する。
- ALBをミニキャッシュとしての利用する。

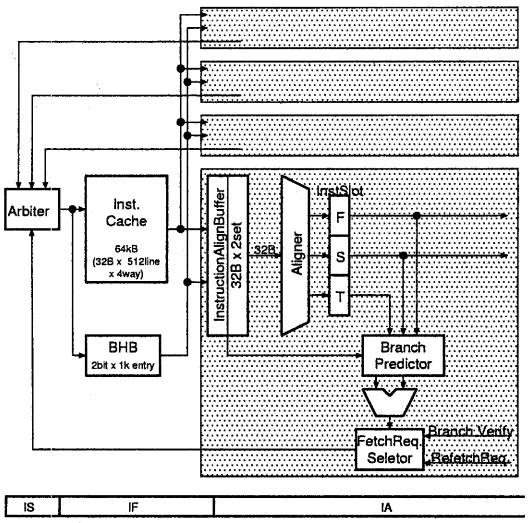

図 3: Merlot命令フェッチ系ブロック図

それぞれの項目について以下の節でその目的と有効点について述べる。

4.1 命令プリフェッチ

Merlotでは共有命令キャッシュを単ポートで構成するため、各PEは命令キャッシュに平均4サイクルに1回のアクセスしかできない。1回のアクセスで4サイクル分の命令を供給できることが最低条件と考え、命令フェッチ幅を2way SuperScalar×4cycle=8命令とした。またMerlotの命令セットは2B/4B混在の可変長命令であることから命令フェッチ幅を4B×8命令=32Bに決定した。

各PEには1サイクルで読み出した命令データを格納するためフェッチ幅と同量の32B命令アラインバッファ(ALB)を用意した。これにより命令キャッシュへのアクセスが平均4サイクルに1回のみであっても各PEはALBから命令を供給できる。

しかしながら、単一のALBは空になることが確認されるまで次のフェッチを行うことができず、分岐がなくても命令供給にバブルが生じる。また単ポート共有命令キャッシュにおける命令プリフェッチ機能は以下の点で重要である。これは分散命令キャッシュや多ポート共有命令キャッシュと比べ、命令競合調停に負けた場合フェッチ要求が発行されてからALBへ命令供給が完了するまでの遅延が長くなるからである。早期にフェッチ要求を行うことはこれらペナルティを減らすことにつながる。

このためMerlotでは32BのALBを2セット用意して一方のALBに空きができた時点で命令のプリフェッチを行うこととした。これにより分岐によりプログラムフローが乱れない限り命令キャッシュのアクセス競合が生じた際も連続して命令を供給することができる。

また二重化したALBは命令がキャッシュラインを跨いだ場合の処理を簡単にする。Merlot命令セットは可変長命令であるため命令キャッシュラインを跨ぐ命令が存在する。このような命令を発行系に供給するためには、キャッシュラインを跨いだ後半がフェッチされるまで命令の前半を予備バッファに保持しておかなければならない。二重化したALBでは常に一方が他方の予備バッファの役割を兼ねることとなり、キャッシュラインを跨ぐ命令に対しても余計なハードウェアリソースを費やさずに解決することができる。

更にMerlotでは、分岐が起こらない場合二重化したALBを利用して異なるキャッシュラインからの2命令同時供給を可能としている。すなわちFスロットへは一方のALBから、Sスロットへは他方からの命令供給を可能としている。これにより連続するプログラムフローであれば、キャッシュラインを跨いでも2組の発行スロットへの供給レートを最大限に取ることができる。

4.2 分岐予測の前倒し

分岐命令などによりプログラムフローが乱れるとALBに格納されていた命令はキャンセルされ新たな命令フェッチを行わなければならない。このためALBの二重化だけでは命令のプリフェッチに対応することができず、バブルが生じる。バブルによるペナルティを低減するためMerlotでは各PEが2wayスーパースカラであるにも関わらず命令スロットは3セット(F,S,T)用意した。

3セット用意した命令スロットのうち発行系への供給はF,Sスロットからのみ行う。これに対しアドレス加算器へのディスプレースメントの供給はF,Sスロットからだけでなく Tスロットからも行う。命令スロットのデータ幅は4B命令長に合わせており、常にF,Sスロットから発行系へ命令供給ができるようAlignerでは合計8Bのデータ処理が可能である。

Merlot は可変長命令であるため 8B の命令データ内に 3 命令以上のデータを含むことが多い。この場合、3 命令目を T スロットに転送し、この命令が分岐命令であれば T スロットからアドレス計算をする。これにより Aligner のハードウェアリソースを増やすことなく分岐時に命令のプリフェッチが可能となる。今回評価を行ったプログラムでは 2B 命令を 45–60 % 程度含むため、T スロットを使用できる割合が高くなることが予想される。

図 4(a) は分岐命令のアドレス計算が F スロットで行われる場合のパイプラインステージ図を示している。分岐先のアドレス計算と分岐先アドレスのフェッチ要求は IA ステージで行われる。このため競合調停の結果命令フェッチ要求が即時に受理されたとしても分岐後の命令供給には 1 サイクルのバブルが生じる。S スロットで分岐アドレス計算を行う場合も同様である。

これに対し T スロットでアドレス計算が行われる場合のパイプラインステージ図を図 4(b) に示す。T スロットにおいてアドレス計算とフェッチ要求が行われ、その後分岐命令は F または S スロットから発行系へ供給される。このため分岐命令のフェッチ要求は (a) の命令発行に比べ 1 サイクル早くなる。この結果
フェッチ要求が PE 間の競合調停に勝つとバブルなしで命令供給を行うことができる。

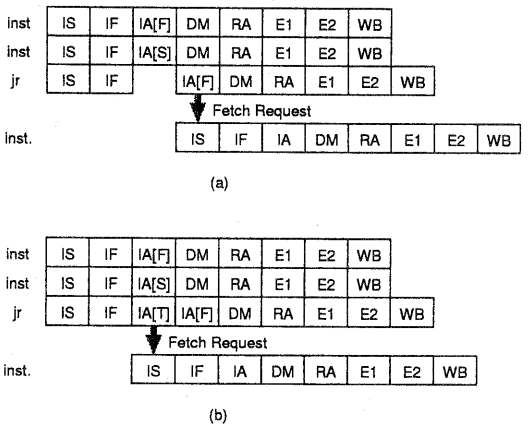

図 4: T スロットの利用

4.3 アラインバッファのミニキャッシュ化

上記プリフェッチにより自 PE への命令供給の充実となる一方、PE 間の命令キャッシュアクセス競合を増加させる。

そこで Merlot では命令キャッシュアクセス競合を減らすことを目的として ALB をミニキャッシュとしても使用することとした。ミニキャッシュは遅延上の問題から正常パスでは使用せず、データキャッシュのバンクコンフリクト時、データキャッシュミス時などのパイプラインフラッシュ時のみに使用する。

これらのパイプラインフラッシュ時には命令キャッシュにアクセスする前に ALB 内のデータの有無を確認し、ヒットした場合には ALB から命令供給を行う。命令キャッシュへのアクセスは ALB にミスしたときのみとなる。

ミニキャッシュにヒットした場合には、その PE は命令キャッシュへのアクセスを行わないため、命令キャッシュアクセス競合を低減することが可能となる。

5 評価及び考察

5.1 評価環境

評価にはパイプラインシミュレータ IMAsim を用いた。これは Merlot の実装をシミュレートすることができる。ただし、キャッシュミスペナルティは実機の振るまいと異なり 125MHz において 16 コアクロックと固定している。この値は Merlot に内蔵した高速 SDRAM の I/F を用いた場合の平均値に近く、十分現実的なものである [3]。

命令フェッチ系の重要な役割は命令供給を十分に行うことである。命令が供給できない要因としては以下の点が考えられる。

- 命令キャッシュミス
- 命令キャッシュ競合
- 分岐命令によるバブル

特に命令キャッシュ競合調停による供給能力低下の問題は、分散型命令キャッシュ方式、多ポート共有命令キャッシュ方式のいずれにおいても存在せず、単ポート共有命令キャッシュ特有である。そこで多ポートの共有命令キャッシュ方式における命令供給能力を理想値として単ポート共有命令キャッシュの評価を行った。

5.2 評価プログラム

実装した機能の有効性を検証するためアプリケーションプログラムを評価した。評価プログラムは MPEG2 伸長プログラム MPEG2dec (Mediabench)、音声符号化プログラム adpcm (Mediabench)、JPEG 圧縮伸長プログラム ijpeg (SPECint95) 及び音声認識処理の HMM 計算部を抜粋した Beam を使用した。各プログラムは Merlot 向け並列化コンパイラによりコード生成したものを用いた [4]。

5.3 Merlot 性能

Merlot と理想的な 4 ポート共有キャッシュにおける各プログラムの 4PE 合計 IPC と単体での命令発行不可要因を表 3に示す。命令発行不可要因は、各 PE でスレッドを実行しているサイクル当たりの発行できなかった平均命令数であり、このうち命令供給がされていないことを原因とする値を示している。2 命令同時発行されたサイクルではこの値はゼロとなり、命令キャッシュミスなどにより ALB が空であるサイクルでは値は 2 となる。この値が低いほど命令供給能力は高いことを示している。

表 3: IPC と命令発行不可要因

	単ポートキャッシュ		4 ポートキャッシュ	
	IPC	命令発行不可要因	IPC	命令発行不可要因
MPEG2dec	1.54	0.74	1.55	0.72
adpcm	1.50	0.29	1.50	0.29
ijpeg	1.64	0.47	1.65	0.44
Beam	2.58	0.35	2.63	0.33

理想的な 4 ポート共有命令キャッシュにおける命令発行不可要因と比べ Merlot のそれは 3–7 %程度増加し、命令供給能力の低下がみられる。特に Beam など並列性の高いプログラムでは命令キャッシュへのアクセス回数増加、すなわち PE 間のアクセス競合増加が起こり、命令供給能力の低下を引き起こす。

しかしながら最も性能に差のある Beam においても Merlot は理想的な 4 ポート共有命令キャッシュと比較して 98 %程度の IPC を出すことができる。

5.4 命令フェッチ幅

図 5,6にフェッチ幅を変化させたときの命令未供給による IPC と命令発行不可要因を示す。命令供給能力が足りないために性能が十分に出ない場合には、命令発行不可要因の値を低減することにより IPC をあげることができる。

図より命令フェッチ幅を広げることにより命令発行不可要因の値を低減し、IPC が上がることがわかる。しかしながら、32B の命令フェッチ幅において MPEG2dec, adpcm, ijpeg では命令発行不可要因の値は 16B に比べ低減しているにも関わらず、IPC はあまり変化しない。これはプログラムの並列性が高くないことや、スレッド間の同期などにある。プログラムの並列性が高い Beam では 32B の命令フェッチ幅にすることにより、IPC を更に上げる。

並列性の高いプログラムでは 32B 以上のフェッチ幅が有効であること、32B 以上で命令発行不可要因の値が飽和していることから十分な命令供給を行うために命令フェッチ幅は 32B 必要である。

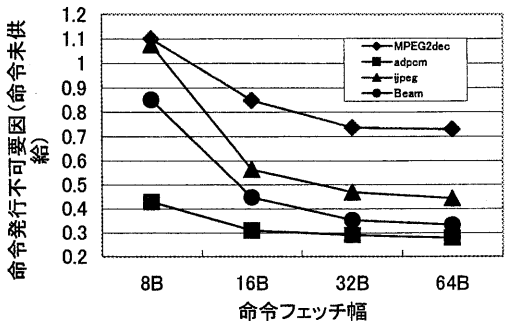

図 5: フェッチ幅に対する発行不可要因 (命令未供給)

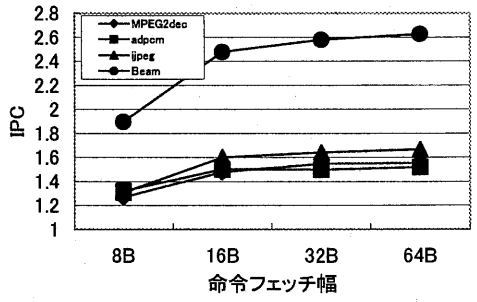

図 6: フェッチ幅に対する IPC

5.5 分岐前倒しの効果

分岐時の命令フェッチ要求を前倒しを行うため実装した T スロット方式について評価を表 4にまとめる。IPC と命令供給不可要因について T スロット使用の有無についてまとめた。 adpcm, ijpeg, Beam

表 4: T スロット効果 (IPC 及び命令供給不可要因)

	IPC			命令供給不可要因		
	なし	あり	比率	なし	あり	比率
MPEG2dec	1.54	1.55	1.00	0.74	0.74	0.99
adpcm	1.45	1.50	1.03	0.30	0.29	0.96
ijpeg	1.59	1.64	1.03	0.49	0.47	0.96
Beam	2.54	2.58	1.02	0.39	0.35	0.91

では IPC の向上は 2–3 %であったが、MPEG2dec ではほとんど性能向上は得られなかった。これは分岐命令の割合が MPEG2dec 以外のプログラムで 6–15 %であるのに対し MPEG2dec では 2.5 %と少な

いことが原因と考えられる。このため分岐のフェッチ要求を前倒しすることによる効果は得られない。命令供給不可要因ではMPEG2dec以外のプログラムで4–9 %低減しており、命令供給能力を高める効果としてTスロットの利用が有効であることが示された。

5.6 ミニキャッシュ化による効果

各PEからのキャッシュアクセス回数低減を目的として実装したALBのミニキャッシュ利用の効果を示すため、命令キャッシュアクセス回数を表5にまとめる。adpcm以外のプログラムにおいてはミニキャッシュの利用により3–6 %の低減され、キャッシュアクセス回数低減に有効であることが分かる。

表5: ミニキャッシュ化の効果 (命令キャッシュアクセス回数)

	なし	あり	アクセス低減率
MPEG2dec	4.34M	4.17M	0.96
adpcm	0.46M	0.46M	1.00
ijpeg	1.99M	1.94M	0.97
Beam	0.32M	0.30M	0.94

6 まとめ

制御並列アーキテクチャMUSCATを実装したオンチップマルチプロセッサMerlotにおける命令フェッチ系の実装について述べた。

プロセッサ間の交信頻度が高いプログラムを並列実行した場合、ヒット率が向上することからMerlotでは共有命令キャッシュを採用した。また集積度が高くなりキャッシュの容量を十分確保できることから単ポートでキャッシュを構成した。単ポート共有命令キャッシュにおける命令供給能力の問題を解決するために実装した手法について述べた。

パイプラインシミュレータにより評価を行った結果、十分な命令供給を行うには命令フェッチ幅としては8命令分(32B)が必要であること、命令スロットを3組用意することで実現した分岐予測の前倒しにより命令供給能力を上げ、結果として3 %程度の性能向上が得られることが分かった。更に、命令アラインバッファのミニキャッシュ化により3–6 %キャッシュアクセス回数を低減させることを確認できた。これらの機能を持ったMerlotは、共有キャッシュとして理想的な4ポートキャッシュと比較し98 %以上の性能を出すことを示した。

今後は更に並列性の高いプログラムに対しても十分な命令供給を行うことができる機能の提案、検討を行っていく予定である。

参考文献

[1] N.Nishi et al."A 1GIPS 1W Single-Chip Tightly-Coupled Four-Way Multiprocessor with Architecture Support for Multiple Control Flow Execution", IEEE Int'l Solid-State Circuits Conference(ISSCC),Feb 2000

[2] 鳥居淳他" 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 June 1998,pp.1622-1631

[3] S.Matsushita et al.,"Merlot: A Single-Chip Tightly Coupled Four-Way Multi-Thread Processor", COOL Chips III,2000,pp.63-74

[4] J.Sakai et al.,"Software Environment for Single-Chip Multi-Processor Merlot(MP98 Ver.1)",COOL Chips III,2000,pp.277

[5] K.Olukotun et al.,"The Case for a Single-Chip Multiprocessor", Proc. of the 7th Int'l. Symp. on Architectural Support for Parallel Languages and Operating Systems, Oct 1996,pp.2-11

[6] G.S.Sohi et al."Multiscalar Processors", 22th Int'l Symp. on Computer Architecture (ISCA-22),May 1995,pp.392-403

[7] B.A.Nayfeh el al."Evaluation of Design Altanatives for a Multiprocessor Microprocessor",23rd Annual Int'l Symp. on Computer Architecture(ISCA-23),May 1996, pp.67-77

[8] 平田博章他," マルチスレッドプロセッサおよび1チップマルチプロセッサのための命令キャッシュ構成・命令フェッチ方式の性能比較", 電子情報通信学会論文誌 Vol.J81-D-I No.6 June 1998, pp.718-727 並列・分散処理研究推進機構, 並列分散処理アーキテクチャ技術成果報告書,1998,pp.34-41