サイクル時間評価による命令レベル並列処理マシンの性能比較

原 哲也[†1] 安藤秀樹[†2]
中西 知嘉子[†3] 中屋雅夫[†4]

近年，高性能なマイクロプロセッサはスーパスカラ技術を多く取り入れている．これに対して，ハードウェアが複雑で性能改善が困難なスーパスカラ・マシンに代わって，簡単なハードウェアで実現できるVLIWマシンが期待されている．しかし，単純なVLIWマシンでは投機的実行が大きく制限され，本当にスーパスカラ・マシンの性能を上回ることができるかどうかは疑問である．そこで我々は，プレディケーティングと呼ぶ投機的実行を支援する機構を提案した．この機構は，VLIWマシンにおける投機的実行に関して従来コンパイラが課せられていた制限を取り除くことができ，評価の結果，実行サイクル数を大幅に削減できることが分かった．種々の命令レベル並列処理マシンの性能を比較するには，従来の研究で行われてきたサイクル数による評価だけでなく，アーキテクチャがサイクル時間に与える影響も評価しなければならない．本論文では，スーパスカラ・マシン，単純なVLIWマシン，プレディケーティングを備えたVLIWマシンについて，実行サイクル数とサイクル時間の両方を評価し，総合的な性能評価を行った．その結果，単純なVLIWマシンはスーパスカラ・マシンの性能を僅かしか上回ることはできない．これに対して，プレディケーティングを備えたVLIWマシンは，スーパスカラ・マシンの性能を大きく上回り，1.41倍高速であることが分かった．

Performance Comparison of Parallel Instruction Execution Machines with Cycle Time Evaluation

TETSUYA HARA,[†1] HIDEKI ANDO,[†2] CHIKAKO NAKANISHI[†3]
and MASAO NAKAYA[†4]

Recent high-performance microprocessors use superscalar techniques. Further performance improvement is difficult because of the hardware complexity. A VLIW machine which can be implemented with simple hardware is expected as a next generation microprocessor. Yet speculative execution is severely restricted in a simple VLIW machine. Thus, it is a question whether the simple VLIW machine really outperforms the superscalar machine. Recently we proposed a mechanism called predicating that supports speculative execution. Predicating removes constraints imposed on the compiler's speculative code motion. Our evaluation showed significant reduction of cycle counts. To compare performance of parallel instruction execution machines, it is required to evaluate how much each machine architecture affects cycle time as well as cycle counts; previous studies evaluate only cycle count speedup. This paper evaluates overall performance of a superscalar machine, a simple VLIW machine, and a VLIW machine with predicating through evaluation of both cycle counts and cycle time. Our evaluation shows that the simple VLIW machine slightly outperforms the superscalar machine while the VLIW machine with predicating achieves a significant speedup of 1.41× over the superscalar machine.

1. はじめに

最近のマイクロプロセッサ[1]は，種々のスーパスカラ技術[2)~4)]を用いて高い性能を実現している．スーパスカラ・マシンでは，命令レベルの並列性（ILP: Instruction-Level Parallelism）を抽出するために，命令の並べ変え，すなわち，命令のスケジューリングを行うだけでなく，分岐方向が判明する前に，その分岐に依存する命令の実行を行う**投機的実行**を行っている．投機的実行をともなう命令スケジューリングによって，

†1 三菱電機株式会社マイコン・ASIC事業統括部
Microcomputer & ASIC Division, Mitsubishi Electric Corporation
†2 名古屋大学大学院工学研究科電子情報専攻
Department of Information Electronics, School of Engineering, Nagoya University
†3 三菱電機株式会社システムLSI開発研究所
System LSI Laboratory, Mitsubishi Electric Corporation
†4 三菱電機株式会社システムLSI開発部
System LSI Development Division, Mitsubishi Electric Corporation

スーパスカラ・マシンは，実行サイクル数の大幅な削減を実現しているが，非常に複雑なハードウェアが必要であり，これによりサイクル時間が増加するという問題を持っている．

これに対して，VLIW（Very Long Instruction Word）マシンでは，実行に先立ちコンパイラがスケジューリングを行うため，単純なハードウェアしか必要としない．このため，サイクル時間に悪影響を与えない．さらに，コンパイラは，プログラムの広い範囲を対象に最適なスケジューリングが可能である．これらの理由で，VLIW マシンは潜在的にはスーパスカラ・マシンより高い性能を得ることができると考えられる．

しかし，コンパイラだけで投機的実行により生じる問題[5]を完全に解決することができないので，投機的実行のハードウェア支援が少ない単純な VLIW マシンでは，コンパイラは大きな制約が課せられる．このため，実際には高い並列性の達成は難しい．よって，単純な VLIW マシンがスーパスカラ・マシンの性能に勝るかどうかは分からない．

近年，コンパイラに課せられている投機的実行に関する制限を取り除くハードウェア機構の研究が行われている[6)~9)]．我々は，**プレディケーティング**[10),11)]と呼ぶ，VLIW マシンにおける投機的実行を支援するハードウェア機構を提案している．実行サイクル数比較による評価を行った結果，プレディケーティングを用いれば高い並列性を得ることができることを確認している．また，プレディケーティングは簡単なハードウェアで実現できるので，サイクル時間を単純な VLIW マシンと同程度にできると考えられる．しかし，サイクル時間の詳細な評価は行っていないので，真にどの程度性能が向上するかはまだ分かっていない．

そこで本論文では，スーパスカラ・マシン，単純な VLIW マシン，プレディケーティングを組み込んだ VLIW マシンについて，実行サイクル数とサイクル時間の両方の評価を行うことにより，これらを総合した性能を評価する．以下，これら命令の並列実行を行うマシンのことを **ILP マシン**と呼ぶこととする．本論文は以下，まず 2 章で，評価を行った 3 つの ILP マシンのアーキテクチャについて説明する．3 章では，これらのアーキテクチャにおいて，命令レベル並列に起因するハードウェアの複雑さについて議論する．4 章では評価したマシン・モデルについて説明し，5 章でサイクル数，サイクル時間の評価結果，そして，それらを総合した性能を示す．最後に 6 章で本論文をまとめる．

2. ILP マシンのアーキテクチャ

2.1 Simple VLIW マシン

評価を行った単純な VLIW マシンを以下，**Simple VLIW マシン**と呼ぶ．Simple VLIW マシンは，複数命令の実行に必要な機能ユニット，レジスタ・ファイル・ポート数等を単に増やしたハードウェア構成である．ハードウェアが単純なので，評価した ILP マシンの中では最も高速に動作すると考えられるが，投機的実行に対するハードウェア支援が十分でないので，ILP を生かすことができない．以下，投機的実行に対するコンパイラの対処について説明する．

コンパイラは，分岐を越える投機的な命令移動を行う際，その命令の書き込みにより他の分岐方向で参照されるレジスタを破壊し，プログラムの意味が変わることを回避しなければならない．このために，**レジスタ・リネーミング**を用いる．すなわち，書き込みレジスタを空のレジスタに変更し命令移動を行い，後に，その変更した書き込みレジスタから元のレジスタに値をコピーする命令を挿入する．レジスタ・リネーミングによるプログラムの意味の維持に関する対処法は，レジスタ値に関しては適用できるが，メモリ値に関しては適用できない．

コンパイラはこれに加え，例外を起こす可能性のある命令を投機的に移動する場合に，注意を払わなければならない．投機的に実行された命令が実行中に例外を起こした場合（**投機的例外**と呼ぶ），その例外を通常の例外と同様に即座に処理すると，実行を不正に停止させるか，あるいは，性能を大きく低下させる可能性がある．これは，その例外処理が真に必要かどうか不明であるにもかかわらず行われるためである．この問題に対しては，コンパイラは対処する方法がない．すなわち，例外を起こす可能性のある命令を投機的に移動することはできない．

以上述べたコンパイラへの制約を緩和するために，我々の Simple VLIW マシンはパイプライン無効化機構を組み込んだ．すなわち，投機的命令が実行されないことが分かった時点で，その実行をパイプライン内でキャンセルする機構を備える．コンパイラはこの機構を用いて，メモリ書き込みおよび例外発生の可能性がある命令を，パイプライン内で無効化が可能な範囲に限定して投機的な命令移動を行うことができる．パイプライン無効化機構は単純なので，サイクル時間に悪い影響を与えることはない．

2.2 プレディケーティング VLIW マシン

プレディケーティングは，2.1 節で述べたコンパイ

```
      if (c1)
i1:     r1 = load r2;
        if (c2)              i1': c1    ? r1 = load r2
i2:       r3 = r1 + 1;       i2': c1&c2 ? r3 = r1 + 1
      } else {               i3': !c1   ? r4 = r1 & r2
i3:     r4 = r1 & r2;
      }
```

(a) (b)

図1 プレディケーティングにおけるコード例
Fig. 1 Code example with predicating.

ラの投機的命令移動に関する制限を取り除く機構である．すなわち，プレディケーティングはコンパイラに対して，プログラムの意味を破壊する命令や例外を発生する可能性のある命令の投機的な移動を可能にする．以下，簡単にプレディケーティングについて説明する．詳細については，文献10)，11)を参照してほしい．

プレディケーティングでは，すべての命令はプレディケートを持ち，次のような形式である．

プレディケート ? 操作

プレディケートは，その命令が依存する分岐条件の論理式である．命令の意味は，「プレディケートが真であるときのみ，操作部で示された操作の結果が有効となる」である．

プレディケートが参照する分岐条件は，汎用レジスタではなく，**CCR**（Condition Code Register）と呼ぶレジスタに記憶する．CCRは複数のエントリを持ち，各エントリは異なる分岐条件の値を記憶する．コンパイラは、CCRのエントリ数と等しいかそれ以下の数の分岐を越える投機的移動が可能である．

図1にプレディケーティングにおけるコードの例を示す．図1(a)が通常のマシンに対するコードで，図1(b)がプレディケーティングにおける対応するコードである☆．命令 `i1` と `i1'`，`i2` と `i2'`，`i3` と `i3'` が対応している．`cn` はCCRの第 n エントリを示す．命令にプレディケートを付加することにより，複数の制御パスは単一の制御パスになる．

マシンは投機的実行を行うために，**逐次的状態**と**投機的状態**の2つのマシン状態を持つ．逐次的状態は，制御依存が解消している命令の実行結果により構成されるマシン状態である．投機的状態は，制御依存が解消していない命令の実行結果により構成されるマシン状態である．命令がどちらの状態のレジスタ値を参照するかはコンパイラが明示し，レジスタ番号に**投機データ指定子 .s** を付加して表す．

命令の発行点では，命令のプレディケートがマシン

☆ このコードはスケジューリングによって並べ変えられるので，図1(a)のコードが一意に図1(b)のコードになるわけではない．

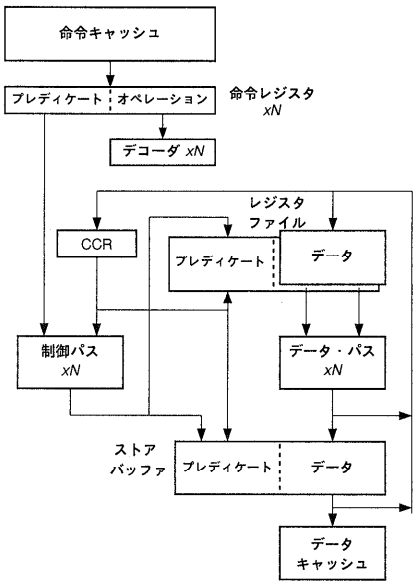


図2 プレディケーティングVLIWマシンのハードウェア構成
Fig. 2 Hardware organization of the predicating VLIW machine.

の分岐条件の値を参照して評価される．その結果，評価値が真であれば，実行を行い逐次的状態を更新する．一方，評価値が偽であれば，命令は無効化される．これらのいずれでもない場合，すなわち，プレディケート評価に必要な分岐条件の値の少なくとも1つが未定義であるために，プレディケートの評価値が定まらない場合は，命令の実行を行い投機的状態を更新する．

図2に N 命令発行のプレディケーティングVLIWマシンのハードウェア構成を示す．Simple VLIWマシンとの違いは，制御パス，プレディケート付きレジスタ・ファイル，および，プレディケート付きストア・バッファである．制御パスは，命令のプレディケートを評価し，実行結果を逐次的状態と投機的状態のどちらに書き込むか，あるいは，命令を無効化するかを制御する．プレディケートの評価を単純な論理で行うことができるように，プレディケートに許可する論理式として，分岐条件およびその否定の論理積に限定した．たとえば，プレディケート `c1&c2` や `c1&!c2` は許すが，`c1|c2` は許さない．さらに，このプレディケートを，真（1）か偽（0）か“don't care”（X）のいずれかの値を持つ要素の列（ベクタ）でエンコードする．プレディケートの論理式が `cn`（の否定でない）項を持つとき，ベクタの第 n 要素を `1`，`cn` の否定項 `!cn` を持つとき，ベクタの第 n 要素を0とする．プレディケートの論理式が `cn`，`!cn` のどちらの項も持たないとき，ベクタの第 n 要素をXとする．たとえば，CCRが

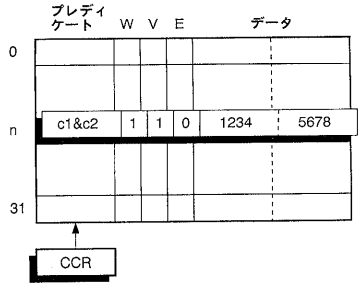


図3 プレディケート付きレジスタ・ファイル
Fig. 3 Predicated register file.

3つの分岐条件 c1，c2，c3 の値を保持する3つのエントリを持つ場合，プレディケート c1&!c2&c3 は 1，0，1，プレディケート c1&c3 は 1，X，1 とエンコードする．これにより，プレディケート評価の論理は単純に，分岐条件とのマスク付き一致演算で行うことができる．

図3にレジスタ・ファイルの構成を示す．各エントリは，2つのデータ記憶部，1つのプレディケート記憶部，3つのフラグ（W，V，E）を持つ．2つデータ記憶部のどちらか一方は，当該エントリに対応するレジスタ値に関する逐次的状態を保持し，他方は投機的状態を保持する．逐次的状態を保持しているレジスタを**逐次的レジスタ**と呼び，投機的状態を保持しているレジスタを**投機的レジスタ**と呼ぶ．フラグ W は，どちらのデータ記憶部が投機的レジスタであるかを示す．フラグ V は，フラグ W で指定された投機的レジスタの保持する値が有効であることを示す．フラグ E は，そのエントリに書き込みを行った命令が実行中に例外を起こしており，かつ，その処理が延期されていることを示す．

投機的実行の結果は，投機的レジスタに書き込まれるが，この際，フラグ V がセットされ，その命令のプレディケートはプレディケート記憶部に書き込まれる．各エントリのプレディケート記憶部は，CCR を参照しプレディケートの値を毎サイクル評価する専用のハードウェアを持つ．プレディケートを真と評価した場合は，投機的レジスタに格納された値で逐次的レジスタを更新し，偽と評価した場合は，投機的レジスタに格納された値を無効化する．投機的状態にある実行結果で逐次的状態を更新することを，投機的実行結果の**コミット**という．プレディケートを真と評価した場合，フラグ W を反転させ，フラグ V をリセットする．これは投機的実行結果のコミットに相当する．また，プレディケートを偽と評価した場合，フラグ V をリセットする．これは投機的実行結果の無効化に相当する．

投機的例外が発生した場合は，実行結果の書き込みにおいて，フラグ V をセットしプレディケートを書き込むほか，フラグ E をセットする．後に，フラグ V とフラグ E の両方がセットされているエントリのプレディケートを真と評価した場合，レジスタ・ファイルは投機的例外を検出したことを知らせる信号を出す．この後，まだコミットされていない値を生成した命令を選択し再実行する．先に例外を起こした命令は，再実行時に再び例外を発生し，これを処理する．

2.2.1 スーパスカラ・マシン

評価を行ったスーパスカラ・マシンは，投機的実行の支援ハードウェアを備え，動的スケジューリングと out-of-order 実行を行う．**リザベーション・ステーション**[2]を備えた複数の機能ユニットにより，複数命令の同時実行を可能とする．出力依存と逆依存への対処にはハードウェアによるレジスタ・リネーミングを，また，正確なマシン状態の保証には**リオーダ・バッファ**[3]を用いる．これらの機構を動的分岐予測と結合させることにより，スーパスカラ・マシンはフェッチした単一命令流からパイプラインがストールしないように動的にスケジューリングを行う．

3. 命令レベル並列処理の複雑さ

ILP マシンのハードウェアの複雑さは，複数命令の並列実行，命令スケジューリング，投機的実行に起因し，これらの複雑さはサイクル時間を延ばす原因となる．本章では ILP マシンの複雑さについて議論する．

3.1 並列実行

3.1.1 レジスタ・ファイル

並列実行では，複数命令のオペランドを供給するためにレジスタ・ファイルのポート数が増加する．ポート数増加はレジスタ・ファイル面積の増加をもたらすが，これによりワード線，ビット線の容量が増え，レジスタ・アクセス時間が長くなる．また，データを記憶するセルは，複数のビット線を駆動しなければならないので，これによりレジスタの読み出しはさらに遅くなる．評価を行うどの ILP マシンも，複数命令のオペランドを供給しなければならないので，レジスタ・ポート数増加によるペナルティはすべての ILP マシンに共通となる．

3.1.2 オペランド・バイパス

オペランド・バイパスでは並列実行によって2つの処理の複雑さが増す．

1つ目の処理は，どの実行結果をバイパスするかを決定する**バイパス制御**である．バイパス制御では，ソー

ス・レジスタ番号と実行結果の書き込みレジスタ番号の比較を行いバイパスすべき実行結果を決定する．典型的な5ステージ・パイプライン構成[12]のスカラ・マシンでは，実行結果を得ているがまだ書き込みを終了していない2つの命令の書き込みレジスタ番号と，オペランド読み出しを行っている1つの命令の2つのソース・レジスタ番号の比較を行えばよいので，4つの比較器が必要なだけである．これに対して，同一パイプライン構成の4命令発行のVLIWマシンでは，パイプライン内にある8つの命令の書き込みレジスタ番号と，オペランド読み出しを行っている4つの命令の8つのソース・レジスタ番号の比較を行わなければならないので，64もの比較器が必要である．これに加え，最新の実行結果を選ぶために優先順位付きで選択を行う必要があり，複雑度が増す．

スーパスカラ・マシンでは，実行結果をデコード中の命令に加えて，すべてのリザベーション・ステーションのエントリに転送する必要がある．各リザベーション・ステーション・エントリでは，すべての実行結果に対してタグ比較を行いオペランドの選択を行うので，複雑さはリザベーション・ステーション・エントリ数の増加にともない増加する．

2つ目の処理は，**実行結果ドライブ**である．ILPマシンでは，実行結果のバイパス先が，スカラ・マシンに比べて大幅に増加するので，実行結果のドライブ時間が増加する．バイパス先は，スカラ・マシンでは2つであるが，4命令発行のVLIWマシンでは8つとなる．スーパスカラ・マシンでは，これに加えリザベーション・ステーションの全エントリに対してもバイパスする必要があるので，評価を行ったマシンでは，バイパス先が合計48（リザベーション・ステーションへ40，デコーダへ8）にもなる．この値は，スカラ・マシンの24倍，4命令発行VLIWマシンの6倍にもなる．実行結果ドライブ時間はALUの実行時間に付加されるので，影響は非常に大きい．

一般的には，処理が複雑になり遅延時間が増加する場合，その処理により多くのパイプライン・ステージを割り当てることによって，サイクル時間に与える影響を緩和することが可能である．たとえば，レジスタ読み出しに半サイクルを割り当てているMIPS R3000[13]に対して，レジスタ読み出しが遅いVLIWマシンでは，1サイクルを割り当てることによりサイクル時間の増加を防ぐことができる．ただし，このような深いパイプラインへの変更は他のペナルティを招くことになる（たとえば，分岐ペナルティの増加）．したがって，慎重にトレードオフを検討しなければならない．

ALU実行に複数のステージを割り当て，実行結果ドライブによって長くなったALU実行時間のサイクル時間への影響をなくす方法が考えられるが，得策ではない．なぜならば，連続するALU操作命令間にデータ依存が存在する場合は非常に多いので，実行サイクル数を許容できないほど増加させてしまうからである．したがって，ALU実行時間に付加される実行結果ドライブの遅延時間の増加は，サイクル時間に直接影響する可能性がある．

3.1.3 分岐実行

多くのスカラ・マシンは分岐方式として遅延分岐方式を採用している．しかし，遅延スロットが複数になるVLIWマシンでは，有効な命令を埋めることは明らかに困難であり☆，VLIWマシンでは有効な方法ではない．そこで，評価を行ったVLIWマシンの分岐方式としては，**分岐先バッファ**[14]による動的分岐予測を用いた．また，動的スケジューリングを行うスーパスカラ・マシンも動的分岐予測を用いる．動的分岐予測は，分岐予測のヒット判定と，その結果に従って次にフェッチする命令のアドレス選択を行う必要がある．これらの処理はサイクル時間を延ばす原因となる．

評価したVLIWマシンは複数分岐命令の同時実行が可能とした．これは，分岐命令の実行頻度が高い非数値計算プログラムにおいて，実行サイクル数を減らすためである．Simple VLIWマシンは，複数の分岐が同時にtakenと判定された場合，最も若い命令アドレスの分岐命令の実行結果を用い分岐するとした．この機能を実現するには，複数の分岐先アドレスの中から，優先順位付きで次のPCの選択を行う必要があるので，複雑度が増加する．これに対して，プレディケーティングでは，通常の分岐命令はプレディケート付きの無条件ジャンプ命令として実現するので，複数の分岐命令が同時にスケジュールされても，実際に実行される分岐命令はたかだか1つである．したがって，分岐先アドレスの優先順位付き選択を行う必要はない．

3.2 命令スケジューリング

VLIWマシンは命令スケジューリングをコンパイラで行うので，ハードウェアにペナルティは課せられない．

これに対して，スーパスカラ・マシンにおける命令スケジューリングには，非常に複雑なハードウェアが必要である．具体的には，データ依存解析，タグ割付け，および，リザベーション・ステーションからの命令発

☆ 単一の遅延スロットを有効に埋めることができる確率でさえ50%程度である[12]．

行を，命令フェッチから命令実行までの間，すなわち，通常の5段パイプラインでは命令デコード・ステージで行われなければならない．この処理の複雑さを考えると，動作速度を低下させないためには，少なくとも2ステージに割当てる必要がある．そこで，データ依存解析とタグ割り付けを1ステージに，命令発行を次のステージに割り当てた．このように2ステージに処理を分けても，なお命令発行は複雑である．なぜならば，命令発行では，リザベーション・ステーション・エントリ内のオペランド・タグと実行結果タグとの一致比較を行いオペランドの有効性を判定した後，オペランドが揃い実行可能となった命令から実際に実行する命令を選択する必要がある．これら命令発行にともなう処理は，1サイクルで行う必要があり，サイクル時間に影響する可能性がある☆．

今回の評価では，データ依存解析とタグ割付け，および，3.3節で述べるリオーダ・バッファの処理にそれぞれ1サイクル割り当てることにより，これらの複雑さはサイクル時間に影響を与えないとした．したがって，デコード・ステージに2サイクルを割り当てることとなり，分岐予測ミス時のペナルティは3サイクルになる．つまり，動的スケジューリングによる複雑さの一部は，分岐予測ミスのペナルティとして性能に反映されることになる．

3.3 投機的実行

Simple VLIWマシンは投機的実行の支援としてパイプライン無効化を採用している．ハードウェアの負担は少なく，サイクル時間に影響を及ぼすことはない．

プレディケーティングVLIWマシンは，投機的実行の支援としてプレディケート付きレジスタ・ファイルが必要であり，レジスタのアクセスが複雑になる．2.2節で述べたように，レジスタ・ファイルの各エントリは2つのデータ記憶部を持つので，このどちらを読み出すかのセレクタ（**ロケーション・セレクタ**と呼ぶ）が必要になる。**図4**に示すように，ロケーション・セレクタは投機データ指定子とレジスタ・エントリのWフラグを参照してデータを選択する．ロケーション・セレクタはアドレス・デコーダの後に挿入されるため，アドレス・デコードからのレジスタ読み出しのパス（**Path-1**と呼ぶ）の遅延時間が増加する．さらに，フラグWは各エントリで行うプレディケート評価結果によって更新されるので，これを含む，プレディケート評価，フラグ更新，ロケーション選択，

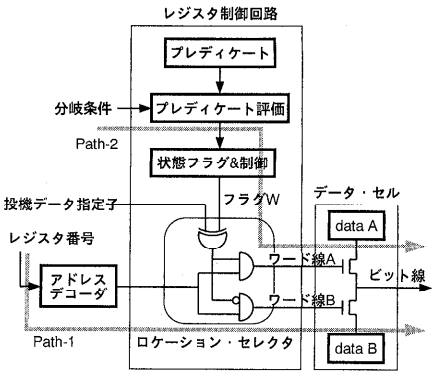


図4 プレディケート付きレジスタ・ファイルのレジスタ制御回路
Fig. 4 Control logic in the predicated register file.

データ読み出しのパス（**Path-2**と呼ぶ）が，新たに，レジスタ読み出しのクリティカル・パスに加わる．

スーパスカラ・マシンでは，リオーダ・バッファのエントリ割付け，リオーダ・バッファ読み出し，および，命令無効化が投機的実行のために必要な処理である．3.2節で述べたように，リオーダ・バッファの処理は1サイクルを割り当てることにより，サイクル時間に影響しないとした．命令無効化がサイクル時間に影響を与えないことは明らかである．

4. マシン・モデル

評価のベースとなるスカラ・マシンを，MIPS R3000とした．ILPマシンの命令デコーダ，機能ユニット，レジスタ・ファイル・ポート，データ・キャッシュ・ポートは，MIPS R3000のハードウェア資源を複写したものとした．

我々は，並列度が4のILPマシンについて評価を行った．**ILPマシンの並列度**とは，1サイクルにデコードする命令数のことである．これは，VLIWマシンでは同時に発行する命令数であり，スーパスカラ・マシンではリザベーション・ステーションに割り当てる命令の最大数となる．ILPマシンのスカラ・マシンに対する性能向上は，マシンの並列度によって上限が与えられる．並列度が2，4，8のスーパスカラ・マシンについて実行サイクル数の評価を行ったところ，並列度4では2の場合に比べて32%の性能向上が見られたが，並列度8にしても僅か3%しか向上しなかった．VLIWマシンにおいても同様な傾向が見られた．このことから，並列度4の場合について評価を行った．

本章では以下，評価を行ったILPマシン・モデルについて説明する．

☆ 1サイクルで行わなければ，たとえばALUの実行結果を次のサイクルですぐに使用することができない．

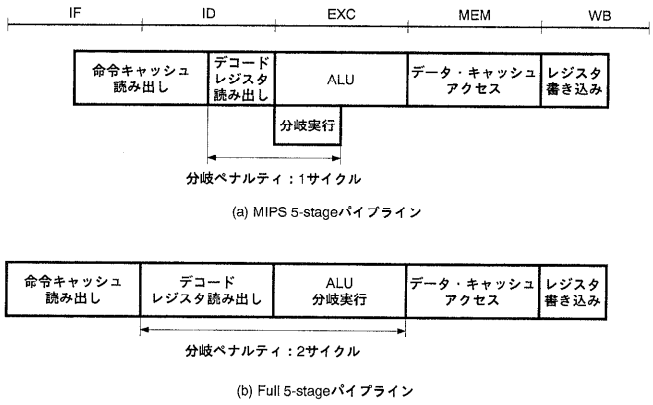


図5 VLIWマシンのパイプライン構成
Fig. 5 Pipeline organization of the VLIW machine.

4.1 Simple VLIW マシン・モデル

評価したSimple VLIWマシンは，4つのALU，4つの分岐ユニット，2つのロード・ユニット，1つのストア・ユニットを持ち，同時に4命令を発行する．レジスタ・ファイルのポート数は書き込みが4，読み出しが8であり，データ・キャッシュのポート数は2である．命令セットはMIPS R3000とほぼ等しい．命令のレイテンシはMIPS R3000と同じであり，ロード命令が2サイクル，その他の整数命令は1サイクルである．分岐方式は，MIPSアーキテクチャがcompare-and-branchを用いているのに対して，Simple VLIWマシンではbranch-on-conditionとした．分岐予測には2Kエントリで2ビット履歴の分岐先バッファを採用した．パイプライン無効化はMEMステージ（**図5**参照）内の命令までを無効化可能とするので，メモリ書き込みおよび例外発生の可能性がある命令は，依存する分岐命令を越えた1サイクル前まで投機的命令移動が可能である．

パイプライン構成の違いで2つのSimple VLIWマシンを評価した．**VLIW-BP1**は，1図5(a)に示すように，MIPS R3000のパイプライン構成と同一であり，命令デコードと分岐実行にそれぞれ半サイクルを割り当てる構成である．このパイプライン構成を，**MIPS 5-stageパイプライン**と呼ぶこととする．この構成では，分岐予測ミス・ペナルティは1サイクルである．3章で述べたように，並列実行の複雑さはレジスタ・ファイル読み出し，オペランド・バイパス，分岐実行に反映されるので，これらの処理を半サイクルで行う必要があるVLIW-BP1ではサイクル時間が延びる可能性がある．そこで図5(b)に示すように，デコードと分岐実行にそれぞれ1サイクルを割り当てる**Full 5-stageパイプライン**構成を持つ**VLIW-BP2**についても評価した．このモデルでは複雑さがサイクル時間に与える影響はないと考えられるが，分岐予測ミス・ペナルティが2サイクルに増加する．VLIW-BP1とVLIW-BP2の性能を比較することにより，クロック速度向上と分岐予測ミス・ペナルティによる性能低下のトレードオフを評価した．

Simple VLIWマシンに対しては，命令スケジューリングは次のようにして行った．まず，実行プロファイルを用いてプログラムより高い頻度で実行される単一のパス，すなわち**トレース**を選択する．その後，トレースの中で命令移動を行いスケジューリングする．レジスタ・リネーミングとハードウェアのパイプライン無効化を使い投機的命令移動を行う．レジスタ・リネーミングでは，コピー命令が挿入されるが，**コピー伝搬**[15]による最適化で，コピー命令へのデータ依存を除去する．さらに，コピー伝搬によってコピー命令の実行結果がもはや不要となった場合，コピー命令を取り除く[15]．その他命令スケジューリングの詳細については，文献16)を参照してほしい．

4.2 プレディケーティングVLIWマシン・モデル

評価したプレディケーティングVLIWマシンは，Simple VLIWマシン・モデルにプレディケーティングを加えた構成である．このマシンを**SPEV**（Speculative Execution VLIW）と呼ぶ．CCRのエントリ数を4とし，4つの分岐を越えた投機的命令移動を可能とする．SPEVはSimple VLIWマシンの複雑さに加えて，投機的実行支援のためにレジスタ・ファイル読み出しの複雑さが増加する．Simple VLIWマシンの場合と同様に，SPEVにも，MIPS 5-stageパイプライン構成をとる**SPEV-BP1**と，Full 5-stageパイプライン構成をとる**SPEV-BP2**の2つのモデルについて評価を行う．

SPEV では，複数のパスから移動された命令の投機的実行をハードウェアが支援するので，命令スケジューリングでは，Simple VLIW マシンの場合と異なり，プログラムから高い頻度で実行される複数のパスを選択し，その中で命令移動を行う．単一のパスをトレースと呼ぶのに対して，複数のパスを我々はリージョン[16]と呼んでいる．SPEV では，プレディケーティングによりリージョン内での自由な投機的命令移動が可能である．命令スケジューリングの詳細については，文献 16) を参照してほしい．

4.3 スーパスカラ・マシン・モデル

評価したスーパスカラ・マシンも VLIW マシン・モデルとほぼ同様の機能ユニットと同一の分岐先バッファを備える．**図 6** にハードウェア構成を示す．スカラ・マシンのコードを実行するので，分岐方式は compare-and-branch である．スーパスカラ・マシンの並列度は 4 であるが，リザベーション・ステーションを用いて最高 9 命令の同時発行が可能である．デコーダからリザベーション・ステーションに割り当てる命令数が 1 サイクルに最大 4 命令であるので，結果バス数も 4 とした．機能ユニットからの実行結果の書き込みは最大 8 つあるので，衝突の可能性があるが，評価の結果，これによる性能低下は僅かに 1%であり，結果バス数を 4 とすることに問題はない．また，リオーダ・バッファのエントリ数は 16 とした．これについてもエントリ数不足による性能低下が考えられるが，32 のエントリを持つ場合に比べて僅か 0.5%しか性能の低下はなく，16 エントリで十分である．

リザベーション・ステーションのエントリ数は，実行サイクル数とサイクル時間に大きく影響する．一般に，スケジューリング対象となる命令の数が多いほど良いスケジューリングが行える．スケジューリングを行う対象となるプログラムの一部を**命令ウィンドウ**という．スーパスカラ・マシンの場合，リザベーション・ステーションにある命令がスケジューリングの対象となるから，これらの命令が命令ウィンドウ内の命令ということができる．より多いエントリ数はより大きな命令ウィンドウを提供するので，実行サイクル数の削減をもたらすが，オペランド・バイパスおよび命令発行の複雑さを増加させ，サイクル時間を延ばす原因となる．そこで，リザベーション・ステーションのエントリ数の違いで 2 つのモデルを評価した．**SS-RS2** は，各 ALU，シフタ・ユニット，各ロード・ユニットのリザベーション・ステーションに 2 つのエントリを，各ストア・ユニット，分岐ユニットのリザベーション・ステーションに 4 つのエントリを備える．分岐ユニットに 4 エントリを備えることにより，SPEV と同様に 4 つの分岐を越えた投機的命令移動が可能である．これに対して，**SS-RS1** は，各リザベーション・ステーションのエントリ数を SS-RS2 の半分とした．このため，スケジューリング能力が SS-RS2 に比べて劣るが，より速いクロックで動作することができる．**表 1** にこれらのモデルのリザベーション・ステーションのエントリ数についてまとめる．

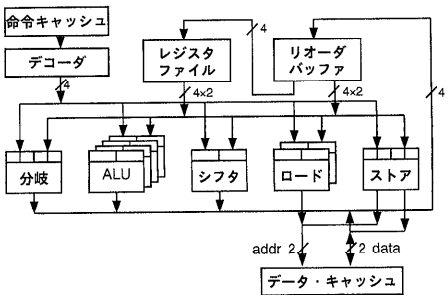


図 6 スーパスカラ・マシンのハードウェア構成
Fig. 6 Hardware organization of the superscalar machine.

表 1 リザベーション・ステーションのエントリ数
Table 1 Number of reservation station entries.

機能ユニット（その数）	SS-RS2	SS-RS1
ALU (4)	2 × 4	1 × 4
シフタ (1)	2 × 1	1 × 1
ロード (2)	2 × 2	1 × 2
ストア (1)	4 × 1	2 × 1
分岐 (1)	4 × 1	2 × 1
合計	22	11

なお，SS-RS2 以上にリザベーション・ステーションのエントリ数を増加させても，サイクル数での性能向上はほとんどなく，SS-RS2 の倍のエントリ数を与えても，サイクル数は 1%しか改善しなかった．

図 7 にスーパスカラ・マシンのパイプライン構成を示す．3 章で述べたように，デコード・ステージとして 2 サイクルを割り当てている．ID1 ステージは，命令デコード，レジスタ・ファイルとリオーダ・バッファの読み出し，データ依存解析，リオーダ・バッファ・エントリの割付け，および，タグ割付けを行う．ID2 ステージでは，リザベーション・ステーションへの命令の書き込みと命令の発行を行う．

5. 性能評価

本章では，実行サイクル数とサイクル時間について評価した後，これらによって計算されるプログラムの実行時間を比較して総合性能を評価する．

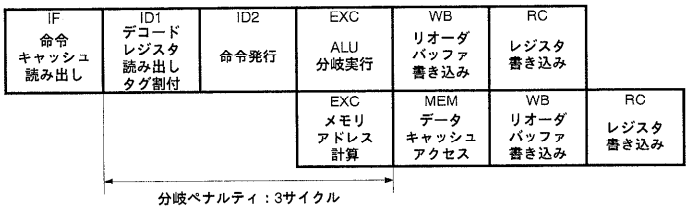


図7 スーパスカラ・マシンのパイプライン構成
Fig. 7 Pipeline organization of the superscalar machine.

表2 ベンチマーク・プログラム
Table 2 Benchmark programs.

プログラム	行数	R3000 サイクル	機能
comp	1,557	21.4 M	データの圧縮
eqn	3,441	27.5 M	真理値表の生成
esp	13,511	11.4 M	PLA の最適化
grep	430	15.7 M	文字列の検索
li	7,429	15.0 M	Lisp 翻訳
nroff	7,276	19.9 M	文書の清書

5.1 実行サイクル数比較

5.1.1 評価方法

表2に示すベンチマーク・プログラムを用いて，実行サイクル数の評価を行った．基本モデルである MIPS R3000 での実行サイクル数は，pixie を用いて求めた．pixie とは，MIPS マシンで実行したプログラムに対する動的な統計データを採取するユーティリティである．メモリ・システムは理想的なものと仮定している．各 ILP マシンでの実行サイクル数は，トレース駆動シミュレータを作成し求めた．MIPS R3000 に対するサイクル数による性能向上は，R3000 での実行サイクル数を各 ILP マシンでの実行サイクル数で割った値である．以下，サイクル数による性能向上を**並列性向上**と呼ぶこととする☆．

5.1.2 実行サイクル数評価結果

図8に並列性向上の評価結果を示す．スカラ・マシンに対する並列性向上は幾何平均で，SS-RS2 で 2.09 倍，SS-RS1 で 1.87 倍であった．SS-RS1 は命令ウインドウ・サイズが小さくなるため，SS-RS2 に対して 10.5%性能が低下する．

VLIW-BP1 の並列性向上は 1.87 倍，VLIW-BP2 は 1.77 倍となった．VLIW-BP2 は分岐予測ミス・ペナルティが大きいので VLIW-BP1 に対して 5.3%の並列性低下となっている．また，VLIW-BP1 の並列性は SS-RS2 に対して 10.5%低下している．これは，

☆ 並列性の向上と実行サイクル数の減少による性能向上は必ずしも対応しないが，ここでは便宜上こう呼ぶこととする．

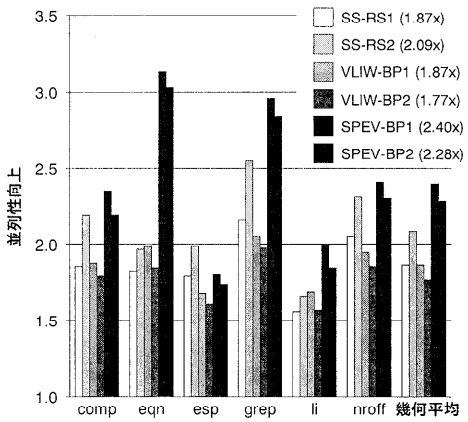


図8 並列性向上
Fig. 8 Parallelism improvement.

Simple VLIW マシンの命令スケジューリングは，スーパスカラ・マシンの動的命令スケジューリングに比べて，より大きな命令ウィンドウより複数の基本ブロックを越えたデータ依存情報を基にして行う点では優れているが，投機的実行が大きく制限されているためである．

SPEV-BP1 の並列性向上は 2.40 倍，SPEV-BP2 は 2.28 倍となった．SPEV-BP2 は分岐予測ミス・ペナルティが大きいので SPEV-BP1 に対して 5.0%の並列性低下となっている．SPEV では投機的実行へのハードウェア支援により，コンパイラは複数の基本ブロック境界を越えた複数のパスから自由に投機的命令移動を行うことが可能であるので，スケジューリング能力を最大限発揮することができる．これにより，SPEV-BP1 は SS-RS2 に対して 14.8%，VLIW-BP1 に対して 28.3%の大きな並列性向上を達成している．

5.2 サイクル時間

5.2.1 評価方法

サイクル時間に影響するクリティカル・パスについて遅延時間を求め，サイクル時間の評価を行った．まず，インバータの遅延値を基準としてそれぞれのゲート，加算器，レジスタ・ファイル，および，キャッシュ

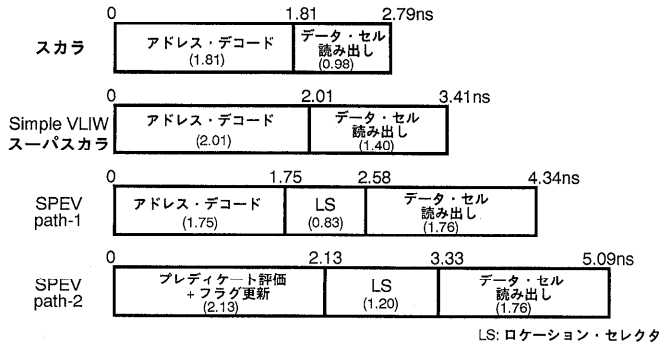


図9 レジスタ・ファイルの読み出し時間
Fig. 9 Delay time of register file read.

等の遅延値を設定して粗い遅延見積を行い，クリティカル・パスの選定を行った．その後，クリティカル・パスの詳細な論理設計を行い，ファンアウトおよび配線長から正確な負荷容量を計算して配線に付加した．遅延時間に影響を与える配線長は，各ユニットのトランジスタ数に他のチップ設計で得られたトランジスタ密度を適用して求めた面積に基づいて計算した．この回路にEPIC社製の静的遅延解析ツールPathMillを使用して遅延値を求めた．こうして得られた結果に基づいて，トランジスタ・サイズおよび論理回路の最適化を繰り返し，最小の遅延値を求めた．使用したプロセスは0.5 μm CMOS，3層配線である．我々の持つハードウェア・ライブラリの値より，16 KBキャッシュのアクセス時間は6.0 ns，32 bit加算器は3.9 nsとした．なお，マルチ・ポートのレジスタ・ファイルは我々のライブラリに存在せず，しかも，その遅延値は回路およびレイアウトに大きく依存するため，実際にデータ・セルのレイアウト設計を行い，正確な遅延値を求めた．

5.2.2 複雑さの評価

各ILPマシンに対するサイクル時間を評価する前に，3章で述べた個々の複雑さが原因で生じる遅延時間の増加について評価する．

レジスタ・ファイル

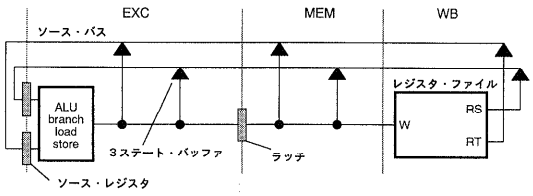
アドレス・デコードからデータ読み出しのパスは，明らかに，レジスタ・ファイルにおけるクリティカル・パスである．プレディケート付きレジスタ・ファイルでは，このほかに，プレディケート評価を含むパスがクリティカルとなる．**図9**にレジスタ・ファイル読み出し時間の評価結果を示す．

Simple VLIWマシンのレジスタ・ファイル読み出し時間は，スカラ・マシンのそれより0.62 ns（22.2%）増加したが，これはポート数増加によるものである．また，SPEV path-1（図4参照）の遅延時間は，Simple VLIWマシンのレジスタ・ファイル読み出し時間より0.93 ns（27.3%）増加したが，これはプレディケート付きレジスタ・ファイルが1エントリに2つのレジスタを持つことによるものである．また，path-1よりpath-2の方が遅延時間が長いことが分かった．この結果，プレディケート付きレジスタ・ファイルの読み出し時間は，Simple VLIWマシンのそれより1.68 ns長く，49.3%増である．

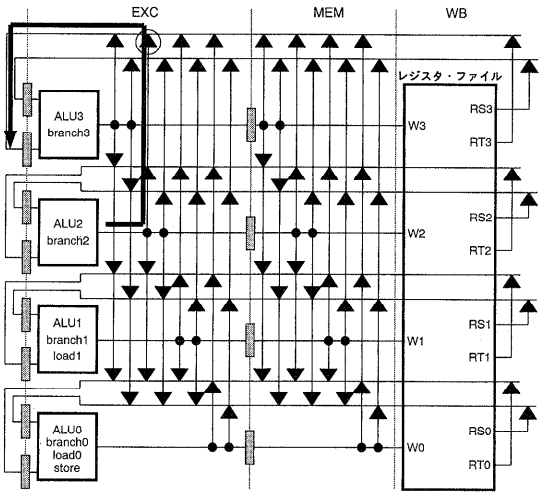
オペランド・バイパスと命令発行

図10に，各ILPマシンのバイパス・ネットワークを示す．Simple VLIWマシンとSPEVにおけるバイパス・ネットワークは，スカラ・マシンにおける回路の単純な拡張である．各パイプライン・ステージにおけるオペランドは，ソース・バスと呼ばれるバスを介して，機能ユニットに設けられた入力レジスタ（ソース・レジスタ）に転送される．たとえば，図10 (b)において，ALU2のEXCステージにある実行結果を，ALU3とbranch3のソース・レジスタに転送するためには，丸で示した3ステート・バッファをイネーブルにすればよい．この実現方法では，バイパスの結果ドライブとは，ソース・バスのドライブである．

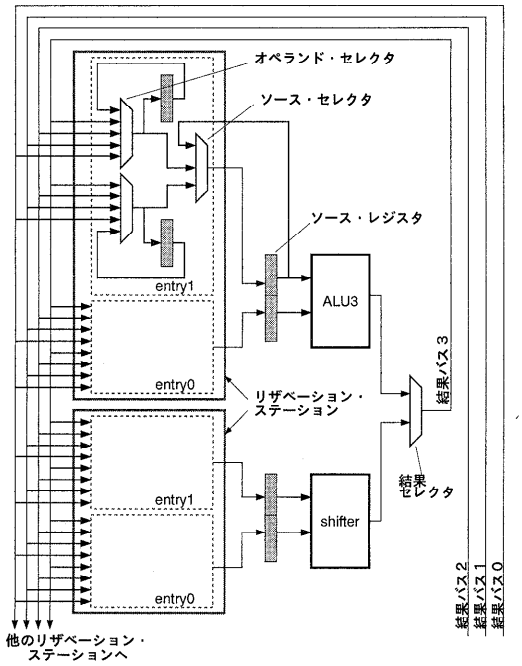
スーパスカラ・マシンでは，実行結果は，リザベーション・ステーションのすべてのエントリに転送されなければならず，転送先が非常に多い．このため，Simple VLIWマシンやSPEVと同様のネットワークでバイパス回路を構成しようとすると，非常に多くのバスが必要となり非現実的である（転送先の数だけバスが必要である）．このため，3ステート・バスを用いるのではなく，セレクタを用いてバイパス回路を構成した．図10 (c)に示すように，3つのセレクタが必要である．**結果セレクタ**は，各機能ユニットの実行結果から調停によって結果バスに出力するものを選択する．**オペラ**



(a) スカラ



(b) Simple VLIW と SPEV



(c) スーパスカラ (SS-RS2)

図10 オペランド・バイパス・ネットワーク
Fig. 10 Operand bypass network.

ンド・セレクタは，タグ比較に従って，リザベーション・ステーションのエントリに取り込むべきオペランドを選択する．**ソース・セレクタ**は，リザベーション・ステーションにあるオペランドから機能ユニットへ出力すべきオペランドを選択する．

3.1 節で述べたように，バイパスはバイパス制御と結果ドライブよりなる．**図11** にバイパス制御の遅延時間と命令発行時間を合わせた評価結果を示す．3.2 節で述べたように，スーパスカラ・マシンでは，バイパスされたデータによって実行可能となった命令から，実際に実行を開始する命令を決定し機能ユニットに送る，命令発行の処理を 1 サイクルで行わなければならないので，バイパス制御と命令発行の時間を合わせている．VLIW マシンでは，命令発行順はコンパイラが定めているので，命令発行時間は必要ない．

Simple VLIW マシンあるいは SPEV に対する遅延時間は，スカラ・マシンのそれより 1.38 ns (56.1%) 増加している．遅延時間増加のほとんどは，優先度付きで転送元の 3 ステート・バッファを選択するために必要な時間であり，これはバイパス先の増加に起因するものである．

SS-RS2 では，バイパス制御 (結果バス調停+タグ・ドライブ，タグ比較+オペランド選択) と命令発行 (オペランド有効性検査とソース・セレクタ制御) に 6.44 ns かかり，VLIW マシンにおけるバイパス制御時間より 67.7%長い．SS-RS1 では，結果タグの転送先が少なく，かつ，リザベーション・ステーションで実行可能となった命令から発行する命令の選択が簡単なので，遅延時間は短くなっている．この結果，遅延時間は 5.44 ns となったが，VLIW マシンに比べるとまだ 1.60 ns (41.7%) 長い．

図12 にバイパスにおける結果ドライブの時間を含めた ALU のレイテンシを示す．結果ドライブ時間を ALU レイテンシに含めたのは，ALU での実行結果を次のサイクルで他の命令で使用できるようにするためである．3.1 節で述べたように，結果ドライブ時間はバイパス先が増加することにより増加する．図 12 に示したように，VLIW マシンでは僅かな遅延時間増加しかなかったが，スーパスカラ・マシンでは非常に多いバイパス先数とセレクタの挿入によって遅延時間が大きく増加した．

分岐実行

図13 に分岐実行の遅延時間を示す．Simple VLIW マシンにおける分岐実行の遅延時間は，スカラ・マシンのそれより短い．これは，スカラ・マシンの compare-and-branch に必要なオペランド比較が，

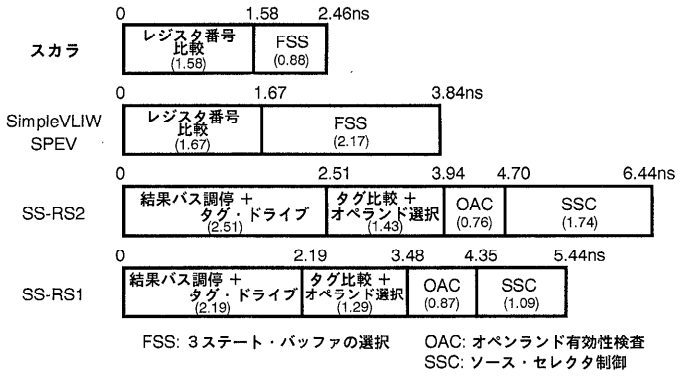


図11 バイパス制御と命令発行の遅延時間
Fig. 11 Delay time of bypass control with instruction issue.

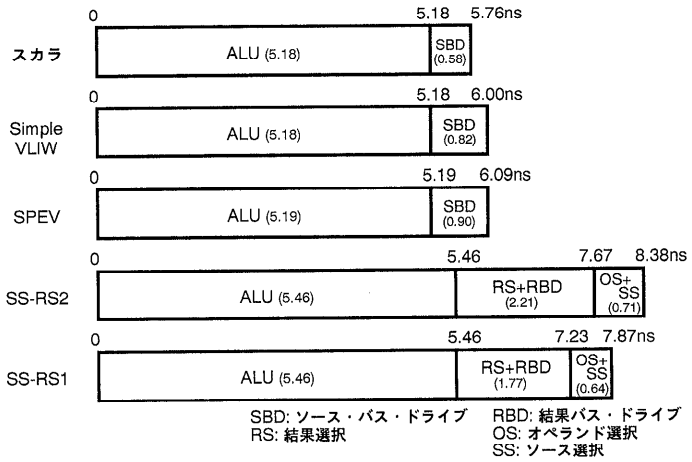


図12 結果ドライブの時間を含めたALUのレイテンシ
Fig. 12 ALU latency with result drive time.

Simple VLIWマシンのbranch-on-conditionでは不要なためである．一方，Simple VLIWマシンでは，複数分岐命令実行と分岐予測のために余分な時間が必要であり（分岐選択と予測チェック/次PC選択），結果としては，スカラ・マシンの分岐実行時間に対して僅かに短くなるだけであった．SPEVでは，分岐命令のプレディケートの評価時間が，compare-and-branchにおけるオペランド比較時間をやや上回っている．さらに，分岐予測のための時間が必要であるが，SPEVでは，プレディケートが真となった分岐命令は実行されるので，条件テストはなく，結果として，スカラ・マシンの分岐実行時間を僅かに延ばすだけにとどまった．スーパスカラ・マシンでは，スカラ・マシンにおける分岐処理に加えて，分岐予測のチェックが必要で，分岐実行時間は20.7%増加した．

5.2.3 サイクル時間評価結果

各ILPマシンのサイクル時間を見積る．5.2.2項で評価した部品を用いるが，パイプラインに組み込むため，パイプライン・ラッチを挿入する．ノンオーバラップ2相クロックを仮定し，パイプライン・ラッチをサイクルの初めと中間点に挿入する．各ラッチにおけるセットアップ時間とホールド時間を満たすように挿入した．

個々のマシンのクリティカル・パスの遅延時間を調べる前に，まず，サイクル時間の下限を調べた．どのマシンにおいても，結果ドライブを含めたALUのレイテンシは，他の部品の遅延時間より長い．したがって，ALUのレイテンシがサイクル時間下限を与える．図12に示す評価結果を得た回路にラッチを挿入し，下限を求めた．**表3**にその結果を示す．

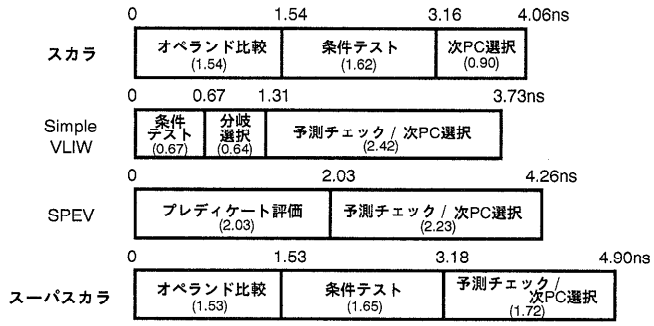


図 13 分岐実行の遅延時間
Fig. 13 Delay time of branch execution.

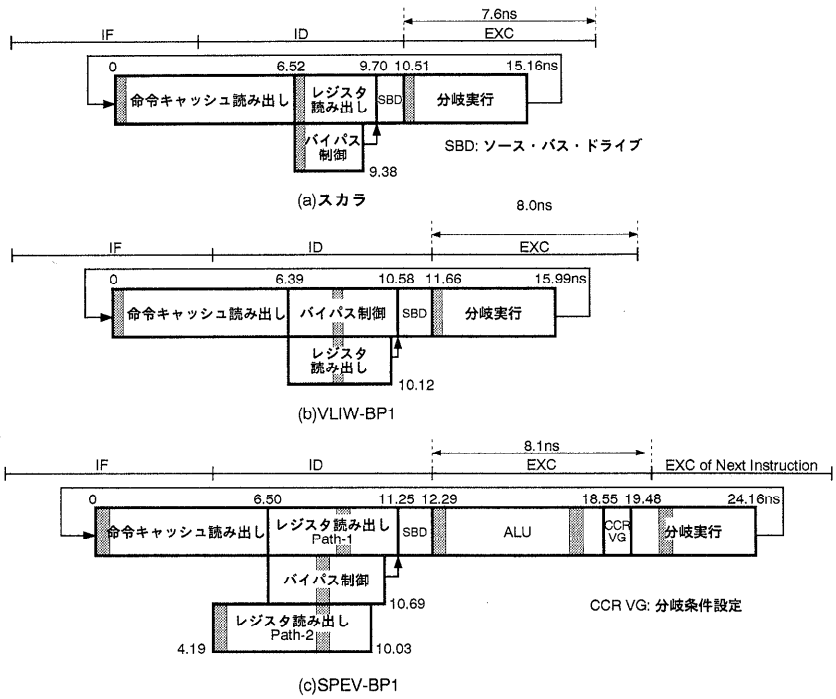


図 14 MIPS 5-stage パイプライン構成のモデルにおける
クリティカル・パスと遅延時間
Fig. 14 Critical paths and delay time in models with the MIPS five-stage pipeline.

表 3 サイクル時間の下限
Table 3 Lower limit of the cycle time.

モデル	スカラ	Simple VLIW	SPEV	SS-RS1	SS-RS2
サイクル時間の下限 (ns)	6.7	6.9	7.1	8.7	9.2

MIPS 5-stage パイプライン・モデル

図 14 に，スカラ，VLIW-BP1，SPEV-BP1 のクリティカル・パスと遅延時間を示す．図中のハッチングの部分は，パイプライン・ラッチを信号が通過する時間である．

スカラ・マシンのクリティカル・パスは，命令キャッシュ読み出し，レジスタ・ファイル読み出し，分岐実行であり，その遅延時間は 15.16 ns であった．この処理は 2 サイクルで行うので，サイクル時間は 7.6 ns となる．バイパス制御はレジスタ・ファイル読み出し時間より短く，サイクル時間に影響を与えなかった．

これに対して，VLIW-BP1 では，レジスタ・ファイル読み出し時間よりバイパス制御の遅延時間の方が長い．これにより，VLIW-BP1 のクリティカル・パスは，

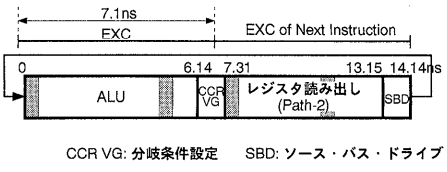


図15 SPEV-BP2のクリティカル・パスと遅延時間
Fig. 15 Critical path and delay time in the SPEV-BP2.

バイパス制御を含むパスとなり，遅延時間は15.99 nsとなった．この処理は2サイクルで行うので，サイクル時間は8.0 nsとなり，スカラ・マシンの5.3%増加である．

同一のパスは，SPEVではクリティカル・パスとはならない．これは，SPEVの分岐命令がレジスタ値を参照するのではなく，ALU実行によって生成されるCCRの値を参照して実行するためである．SPEVのクリティカル・パスは，命令キャッシュ読み出し，レジスタ・ファイル読み出し，ALU実行，分岐条件設定，分岐実行のパスとなる．レジスタ・ファイル読み出しの2つのパスのうち，Path-1がクリティカル・パスに含まれる．これは，Path-1で用いるレジスタ番号がサイクルの途中から有効となるのに対して，Path-2で参照する分岐条件の値は，サイクルの初めで有効となるからである．このクリティカル・パスの処理は3サイクルで行うので，サイクル時間は8.1 nsとなり，スカラ・マシンの6.6%増加である．

Full 5-stage パイプライン・モデル

Full 5-stageパイプラインのモデル（VLIW-BP2, SPEV-BP2）は，バイパス制御，レジスタ・ファイル読み出し，分岐実行にそれぞれ1サイクルを割り当てる構成である．このため，VLIW-BP1のクリティカル・パスの処理をVLIW-BP2では3サイクルで行うことになり，1サイクルあたり5.3 nsとなる．この値は表3のサイクル時間の下限より短いため，VLIW-BP2のクリティカル・パスはALU実行のパスとなる．よって，サイクル時間は6.9 nsとなり，VLIW-BP1の13.8%短縮である．同様に，SPEV-BP1のクリティカル・パスの処理はSPEV-BP2では4サイクルで行うのでサイクル時間に影響しない．クリティカル・パスとなるのは**図15**に示すように，ALU実行，分岐条件設定，レジスタ読み出し（Path-2）である．この処理を2サイクルで行うので，SPEV-BP2のサイクル時間は7.1 nsとなり，SPEV-BP1に比べ12.3%短縮となる．

SS-RS2およびSS-RS1では，5.2.2項で述べたように，レジスタ・ファイル読み出し，バイパス制御と命令発行の合計，および、分岐実行の各遅延時間は，ALUのレイテンシより短くサイクル時間を制限しない．したがって，ALUのレイテンシがサイクル時間となり，SS-RS2は9.2 ns，SS-RS1は8.7 nsとなる．

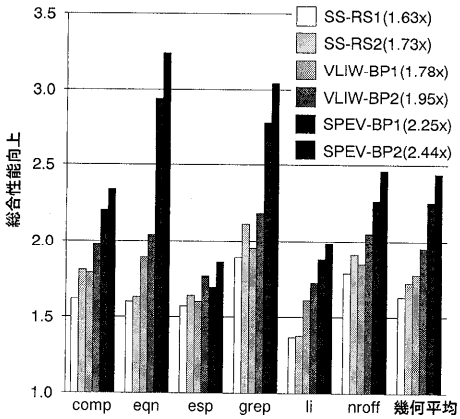


図16 ILPマシンの総合性能向上
Fig. 16 Overall performance improvement of ILP machines.

5.3 総合性能評価

5.1節で求めた実行サイクル数と5.2節で求めたサイクル時間より実行時間を求め，各ILPマシン・モデルの性能を比較する．モデルiの実行時間$_i$は，

実行時間$_i$ = 実行サイクル数$_i$ × サイクル時間$_i$

で計算される．スカラ・マシンに対する総合性能向上は，

$$\text{総合性能向上}_i = \frac{\text{実行時間}_{scalar}}{\text{実行時間}_i}$$

である．

$$\text{並列性向上}_i = \frac{\text{実行サイクル数}_{scalar}}{\text{実行サイクル数}_i}$$

であるので，総合性能向上は以下の式で計算できる．

$$\text{総合性能向上}_i = \text{並列性向上}_i \times \frac{\text{サイクル時間}_{scalar}}{\text{サイクル時間}_i}$$

図16にILPマシンの総合性能向上を示す．SS-RS2はSS-RS1と比較して，サイクル時間が長いが，より多くの並列性を引き出すことができ，6.1%の総合性能向上となっている．これとは対照的に，VLIW-BP1はSS-RS2に対して並列性の点では劣っているが，サイクル時間が短いことにより，総合性能は僅かに（2.9%）向上している．VLIW-BP2は，サイクル時間が短いことによりさらに性能を向上させることができ，VLIW-BP1に比べて9.6%総合性能が高い．この結果より，VLIWマシンでは並列実行の複雑さがデコード・ステージに反映されるため，スカラ同様の半サイクルを

表4 評価結果のまとめ
Table 4 Evaluation summary.

モデル	並列度向上	サイクル時間	総合性能向上
スカラ	1.0	7.6 ns	1.0
SS-RS1	1.87×	8.7 ns	1.63×
SS-RS2	2.09×	9.2 ns	1.73×
VLIW-BP1	1.87×	8.0 ns	1.78×
VLIW-BP2	1.77×	6.9 ns	1.95×
SPEV-BP1	2.40×	8.1 ns	2.25×
SPEV-BP2	2.28×	7.1 ns	2.44×

割り当てるパイプライン構成はバランスが良くなく，分岐予測ミス・ペナルティを増加させても深いパイプライン構成にし，クロック速度を上げる方が得策であることが分かる．

SPEV は効率的な投機的実行支援によりサイクル時間の増加を抑えつつ，制限のない投機的命令移動モデルをコンパイラに提供することによって高い並列性を引き出すことが可能である，他の ILP マシンに比べて大きな性能向上を達成している．SPEV-BP2 の総合性能向上はスカラ・マシンに対して 2.44 倍となり，SS-RS2 の 1.41 倍，VLIW-BP2 の 1.25 倍である．**表4** に評価結果のまとめを示す．

6. まとめ

本論文では，スーパスカラ，Simple VLIW，プレディケーティング VLIW の3つの ILP マシンについて，並列性の向上とハードウェアの複雑さのトレードオフを定量的に比較した．スーパスカラ・マシンは動的スケジューリングを行うハードウェアが複雑であるため，3つの ILP マシンの中では最も性能が低い結果となった．Simple VLIW マシンは高速に動作できるが，コンパイラのスケジューリングに対する制約が大きく並列性向上が少ない．このため，スーパスカラ・マシンに対する性能向上は大きくない．プレディケーティングは投機的実行に関する制限が少ないため，大きな並列性向上を得ることができる．また，ハードウェアも単純でありサイクル時間の増加は少ない．この両点により大きな性能向上を得ることができた．プレディケーティング VLIW マシンは，スーパスカラ・マシンの 1.41 倍，スカラ・マシンの 2.44 倍の性能向上が達成できることを確認した．

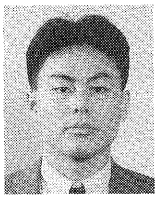
謝辞 本研究に対してご支援いただいたシステム LSI 開発研究所部長・角正氏に感謝致します．

参考文献

1) 浅見直樹，枝 洋樹：次世代マイクロプロセサ，スーパスカラと VLIW が融合，日経エレクトロニクス，No.626, pp.67–150 (1995).
2) Tomasulo, R.M.: An Efficient Algorithm for Exploiting Multiple Arithmetic Units, *IBM Journal*, Vol.11, No.1, pp.25–33 (1967).
3) Smith, J.E. and Pleszkun, A.R.: Implementation of Precise Interrupts in Pipelined Processors, *Proc. 12th Int. Symp. on Computer Architecture*, pp.36–44 (1985).
4) Murakami, K., Irie, N., Kuga, M. and Tomita, S.: SIMP (Single Instruction Stream/Multiple Instruction Pipelining): A Novel High-speed Single-processor Architecture, *Proc. 16th Int. Symp. on Computer Architecture*, pp.78–85 (1989).
5) Johnson, M.: *Superscalar Microprocessor Design*, Prentice Hall, Englewood Cliffs, NJ (1991).
6) Hsu, P.Y.T. and Davidson, E.S.: Highly Concurrent Scalar Processing, *Proc. 13th Int. Symp. on Computer Architecture*, pp.386–395 (1986).
7) Colwel, R.P., Nix, R.P., O'Donnell, J.J., Papworth, D.B. and Rodman, P.K.: A VLIW Architecture for a Trace Scheduling Compiler, *Proc. 2nd Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, pp.180–192 (1987).
8) Smith, M.D., Lam, M.S. and Horowitz, M.A.: Boosting Beyond Static Scheduling in a Superscalar Processor, *Proc. 17th Int. Symp. on Computer Architecture*, pp.344–355 (1990).
9) 小松秀昭，古関 聰，鈴木秀俊，深澤良彰：拡張 VLIW プロセッサ GIFT における命令レベル並列処理機構，情報処理学会論文誌，Vol.34, No.12, pp.2599 2610 (1993).
10) 安藤秀樹，中西知嘉子，原 哲也，中屋雅夫：プレディケート付き状態バッファリングによる投機的実行，並列処理シンポジウム JSPP'95，pp.107–114 (1995).
11) Ando, H., Nakanishi C., Hara T. and Nakaya, M.: Unconstrained Speculative Execution with Predicated State Buffering, *Proc. 22nd Int. Symp. on Computer Architecture*, pp.126–137 (1995).
12) Hennessy, J.L. and Patterson, D.A.: *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Mateo, CA (1990).
13) Kane, G.: *MIPS RISC Architecture*, Prentice Hall, Englewood Cliffs, NJ (1988).
14) Lee, J.K.F. and Smith, A.J.: Branch Prediction Strategies and Branch Target Buffer Design, *Computer*, Vol.17, No.1, pp.6 22 (1984).
15) Aho, A.V., Sethi, R. and Ullman, J.D.:

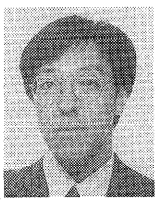
Compilers: Principles, Techniques, and Tools, Addison-Weslay, Reading, MA (1986).
16) 安藤秀樹，中西知嘉子，原　哲也，中屋雅夫：非数値計算応用におけるプレディケート実行向け命令スケジューリング，並列処理シンポジウムJSPP'96, pp.65–72 (1996).

(平成8年 9 月13日受付)
(平成8年12月 5 日採録)



原　哲也（正会員）

1966年生．1989年九州大学工学部情報工学科卒業．1991年同大学大学院総合理工学研究科情報システム学専攻修士課程修了．同年三菱電機（株）LSI研究所に入社し，細粒度並列処理アーキテクチャの研究に従事．1996年より信号処理プロセッサの開発に従事．現在，同社マイコン・ASIC事業統括部所属．



安藤　秀樹（正会員）

1959年生．1981年大阪大学工学部電子工学科卒業．1983年同大学大学院修士課程修了．同年三菱電機（株）LSI研究所入社．ISDN用ディジタル信号処理LSI，第5世代コンピュータ・プロジェクトの推論マシン用プロセッサの設計に従事．1991年Stanford大学に客員研究員として留学．1996年京都大学工学博士．1997年名古屋大学大学院工学研究科電子情報専攻・講師．計算機アーキテクチャ，コンパイラの研究に従事．

中西知嘉子

1988年大阪大学基礎工学部情報工学科卒業．同年三菱電機（株）LSI研究所に入社．以来，計算機用LSIの開発に従事．現在，同社システムLSI開発研究所所属．

中屋　雅夫（正会員）

1951年生．1974年早稲田大学理工学部電子通信学科卒業．1976年同大学大学院修士課程修了．1988年工学博士（早稲田大学）．1976年三菱電機（株）入社．以来，高速MOSゲートアレイ，ECLゲートアレイ，MOSA/D，D/Aコンバータ，三次元回路素子，通信用LSI，ISDN用LSI，並列処理プロセッサ，ATM-LAN用LSIなどのシステムLSIの研究開発に従事．現在，三菱電機（株）システムLSI開発部勤務．1993年度電子情報通信学会論文賞受賞．IEEE，電子情報通信学会各会員．