命令再構成型 VLIW プロセッサ V++ における 適応型再構成戦略

2C－1

金岡弘記　高木浩光　有田隆也　川口喜三男
名古屋工業大学

1. はじめに

VLIWプロセッサの問題点として、コード量が増加することと実行時間変動に対して弱いということが挙げられる。それらに対する解決手段として規定型再構成と適応型再構成を用いた、V++プロセッサアーキテクチャ[1,2]を我々は提案している。V++では、この2つの再構成の効果的な融合が重要となる。本稿では、適応型再構成のための同期方式として重複可能バリア同期[3]を用いた場合について、ソフトウェアパイプライニングと同様の効果をコード量を増加させないで実現する方法、基本ブロックの境界における同期タグを実現する方法、及び基本ブロックの境界における投機的実行の実現の方法について述べる。

2. V++

2.1 規定型再構成

規定型再構成とは、コンパイラ等によりあらかじめ命令に付加された実行タイミングの情報に基づいて、実行時に命令の再構成をおこなうものであり、各命令はフェッチ後指定されたクロック遅らされた後、実行される。これにより同時にフェッチされた命令の組みと、同時に実行される命令の組みとが異なることが許され、基本ブロック間でのオーバラップが可能となる。その結果、NOP命令を減らしコードの命令密度を高くすることができる。

2.2 適応型再構成

適応型再構成とは、命令実行時に種々の要因により発生する命令実行時間の変動のために生じる不要な待ちを、ハードウェアで極力排除し実行時間の増大をおさえることを目的としたものである。これを実現するために、高速な同期機構(重複可能バリア同期機構)を用いてユニット間の実行時間の差を吸収することにより、無駄な待ちを解消する。また、動的に命令の実行順序を保証するので、従来のVLIWのようにタイミングを保証するためだけのNOP命令は必要ない。

2.3 V++ の構成

V++の基本構成を図1に示す。V++には従来のVLIWプロセッサに、再構成を行なうための遅延レジスタ(Delayer)、適応型再構成を実現するための同期ユニット(Synchronizer)を付け加えている。また、ユニットごとのオペレーションの相対的な実行進度を表すDelay-Counter(DC)が付加されている。本稿では、同期ユニットとして重複可能バリア同期機構を用いることを仮定している。実行ユニットは4つとし、転送ユニット(TU)・演算ユニット(AU)×2・分岐ユニット(BU)の構成とする。

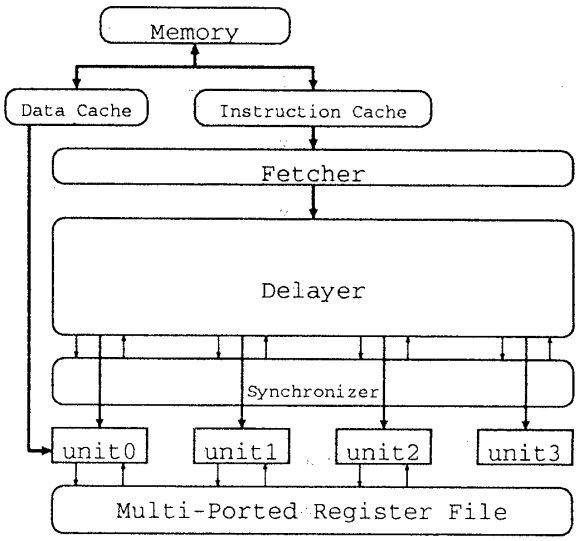

図 1: V++の基本構成

3. 適応型再構成戦略

3.1 適応型再構成によるソフトウェアパイプライニング

VLIWの効果的実行を実現する手法としてソフトウェアパイプライニングがある。ソフトウェアパイプライニングは、ループをオーバラップさせることにより十分な並列性を抽出することで、実行の高速化を達成するものである。ソフトウェアパイプライニングにおける定常状態では、異なるイタレーションに属するオペレーションの組みが同時に実行されなければならないため、定常状態に至るまでのプロローグ、定常状態からループを抜けるまでのエピローグが必要となる。これに対してV++では、適応型再構成によって、フェッチされた1ワード上のオペレーションの組みと同時に実行されるオペレーションの組みとが異なることが許されるため、単純なループならばプロローグ・エピローグを用いることなくループ本体のコードだけでソフトウェアパイプラインを実行できる。

*Strategies for Adaptive Restructuring in the Instruction-Restructurable Processor Architecture V++
Hiroki KANEOKA　Hiromitsu TAKAGI
Takaya ARITA　Kimio KAWAGUCHI
Nagoya Institute of Technology

3.2 同期タグの規定型再構成

コンパイラは、適応型再構成に必要な同期操作を実現するために、プログラムコード中に同期が必要となる位置(バリア領域の入口・出口)の情報を埋め込む。この情報を同期タグと呼ぶ。入口を示す同期タグの付加された命令は、その実行直後にバリアの入口が位置することを意味し、出口を示す同期タグの付加された命令は、その実行直前にバリアの出口が位置することを意味するものとする。しかし、このような同期タグ構成では、基本ブロックの境界においてバリアの整合性を保つために、バリアの存在を示すだけの命令が必要になり、場合によってNOP命令を埋める必要がある。その場合結果的に、ソフトウェアパイプライニングにおけるプロローグ・エピローグと同様のコードが必要となってしまう。そこで、同期タグの規定型再構成を行なうことによって、フェッチされた同期タグを、それと同時にフェッチされたオペレーションに対して指定された相対位置のオペレーションに付加されているものとして扱うことを可能にする。これにより同期処理のためだけに必要な命令なしに整合性を保つことができる。

3.3 基本ブロックの境界における投機的実行の実現

分岐命令がフェッチされた時、次にフェッチされる基本ブロックは、分岐予測によって決められる。分岐命令の実行が終了し分岐先が確定した時に分岐予測が失敗しているならば、確定した分岐先の基本ブロックの命令により、既にフェッチされていた命令は置き換えられる。また、分岐先が確定するまでの間、副作用のない範囲で分岐予測された基本ブロックのオペレーションが実行される。これを投機的実行と呼び、その副作用のない範囲を越えて実行しないことを、同期処理によって保証する。

分岐命令で分岐予測が失敗したとき、投機的実行がなされているユニットがある場合、同期機構の状態に矛盾が生じる可能性があるため、同期機構の状態をclearする必要がある。この時、BU以外のユニットが基本ブロックの境界に達しているものと達していないものとがある場合は、境界に達していないユニット全てが境界に達した後、同期機構の状態をclearして次の基本ブロックを実行する。一方、BU以外の全てのユニットが基本ブロックの境界に達していなかった場合は、同期機構の状態に矛盾が生じないのでclearする必要がない。このような動作により同期機構の状態は常に正しい状態に保たれる。

分岐予測が失敗した場合、基本ブロックの境界の検知が必要がある。それは、DCの値を調べることによって可能である。BUが持つDCに対するそれ以外のユニットのDCの相対値が正ならば、そのユニットは基本ブロックの境界に達していないと判断でき、0以下ならば、そのユニットは基本ブロックの境界に達していると判断できる。

動的変動率 (%)	0	5	10	15	20
V++ (cycles)	47	50.5	52.3	56.6	58.9
規定型のみ V++ (cycles)	47	50.6	54.4	58.3	62.5
実行時間の比 (%)	100	99.8	96.2	97.2	94.2

表 1: 実行時間の比較

4. シミュレーション結果

V++アーキテクチャのシミュレータを作成し、実行時間を測定した。サンプルプログラムとして10の階乗を計算するプログラムを用いて、規定型再構成のみを用いたV++と適応型再構成も用いたV++の実行時間を比較した。オペレーションには動的変動があるものとし、TU,AUの実行されているNOP以外のオペレーションに対し、0,5,10,15,20%の確率でそのクロック中に実行終了できないとした。結果を異なる変動で20回実行し、その平均で評価した。

その結果を表1に示す。このサンプルプログラムでは、動的変動が大きくなるにつれて約0 ～ 6%実行時間が短縮されている。今回用いたサンプルプログラムの基本ブロックは、大きくても2命令であるので、動的変動をその基本ブロック内で吸収するのは、ほとんど不可能であると考えられる。また、投機的実行を効果的にするようなレジスタ割当ての最適化を行なわなかったために十分な実行時間の短縮が行なわれなかったと考えられる。実行時間が6%程度短縮されたのは、ほとんどの場合、分岐予測が当たった際に、実質的な基本ブロックサイズが十分大きくなって、動的変動が吸収されたためと考えられる。

5. おわりに

今回の結果から、並列度の低いサンプルプログラムでも約0 ～ 6%実行時間が短縮されることを確認できた。今後、投機的実行を効果的にするようなレジスタ割当ての最適化を行なうことにより実行時間のさらなる短縮が可能であると予想できる。

参考文献

1. 有田隆也, 曽和将容, "動的再構成型 VLIW プロセッサアーキテクチャ V++", 並列処理シンポジウム JSPP '92 論文集,pp.265-272(1992).
2. 有田隆也, 曽和将容, "命令語再構成型プロセッサアーキテクチャ", 電子情報通信学会論文誌, Vol. J76-D-I, No. 4, pp. 184-186 (1993).
3. 高木浩光, 有田隆也, 曽和将容, "細粒度並列実行を支援する種々の静的順序制御方式の定量的評価", 並列処理シンポジウム JSPP '91 論文集,pp.269-276(1991).