3J-9

基本ブロックを投機単位とする並列実行VLIWアーキテクチャの予備評価

田中耕司† 横田隆史† 大津金光† 大川猛†
†宇都宮大学大学院工学研究科情報システム科学専攻

1 はじめに

命令レベル並列性とスレッドレベル並列性，二種類の並列性を利用しプロセッサの性能向上を実現する手法として，我々はBBPC(Basic Block Parallel Computing)[1]を開発している．

BBPCは実行中の基本ブロック内におけるVLIW命令と，投機を行う基本ブロックのVLIW命令を合成し，一度に実行する手法である．しかしこの手法は現在，有効性の評価が不十分である．本研究では，VLIW命令合成を行うソフトウェアシミュレータを作成した．このシミュレータを用いてBBPCの性能に関する予備評価を行った．

2 並列実行手法BBPC

BBPCではVLIWをベースとしたアーキテクチャによって並列実行手法を実現している．VLIWアーキテクチャの利点として比較的低いハードウェアコストでプロセッサの高性能化が図れる点が挙げられるが，BBPCではVLIWプロセッサの利点を継承し，かつ単一のスレッドでは活用されることのないVLIW命令中の空きスロットを有効活用することにより性能向上を達成する．

また，図1はBBPCによって実現される投機的マルチスレッディングを図示したものである．どのようなプログラムであっても，条件分岐によって基本ブロックを単位とした実行経路を選択していると解釈できる．したがって，基本ブロックを単位として投機を行い，分岐先基本ブロックを投機スレッドとして並列実行することでスレッド数を増やすことができる．これにより並列性が向上し，処理時間の短縮が期待できる．このことから，BBPCでは図1のようにプログラム中の分岐前基本ブロックをメインスレッド，分岐先基本ブロックを投機スレッドとし，VLIWプロセッサ上で同時マルチスレッディングを実現する．分岐前基本ブロックのVLIW命令内で有効に使われていない命令スロットに，分岐先基本ブロック内のVLIW命令を合成することによって命令の空きスロット部分を削減でき，命令実行の並列度が向上する．合成したVLIW命令を実行することにより投機的マルチスレッディングを実現する．

3 BBPCの実現アーキテクチャ

図2はBBPCを実現するアーキテクチャ構成図である．BBPCではVLIWアーキテクチャに，BBPC独自のモジュールを追加することで実現している．図2における網掛け部分が追加されたモジュールである．BB-Cacheはループ判定，パスの分岐履歴保存，パス命令の保存を行う他，ループ内のメインおよび投機スレッドとなる基本ブロックの決定を行う．Instruction Generatorはメインおよび投機スレッドとなる基本ブロックの各VLIW命令を合成する機構であり，二つ目のレジスタファイルは各スレッド間におけるデータの同期と投機失敗時のデータ破棄を行うために追加された機構である．BBPCではプログラム中のループを判別し投機を開始するが，投機実行時は図2における網掛け部分のモジュールを使用し，投機実行時と非投機実行時では使用するパイプラインが一部異なる．

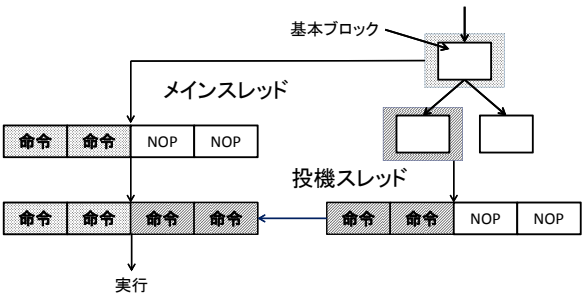

図1: BBPCの実行方式

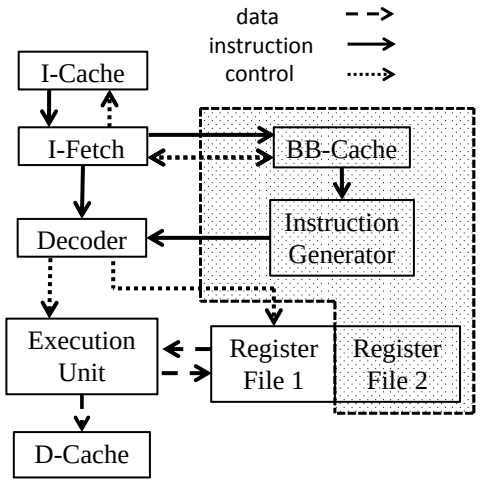

図2: BBPCを実現するアーキテクチャ

Preliminary Evaluation of Speculative Parallel Computing VLIW Architecture Based on Basic Block
†Koji Tanaka, Takashi Yokota, Kanemitsu Ootsu, Takeshi Ohkawa
Graduate School of Engineering, Utsunomiya University (†)

4 BBPCのVLIW命令合成

VLIW命令合成の動作は図2に示した機構Instruction Generatorで行われる．動作の流れとしては，まずBB-Cacheでメインスレッドおよび投機スレッドとなる基本ブロックが決定される．その後，メインおよび投機スレッドとなった基本ブロックのVLIW命令がInstruction GeneratorへVLIW命令ずつ送信される．そして機構Instruction Generatorにおいて，メイン側のVLIW命令の空きスロットに投機側のVLIW命令を合成する．合成された命令は命令デコーダに送られる．図3がBBPCにおける命令合成の原理図である．BB-Cacheから送られてきた各VLIW命令はメイン側のVLIW命令の空きスロットを埋めるように合成される．

また，メイン側のVLIW命令の空きスロットを投機側のVLIW命令内の操作で埋めるが，空きスロットを全て埋めて余った分については次以降のメイン側VLIW命令の空きスロットを埋めるために使用される．合成された命令は定められたVLIW命令の規則を守っていなければならない．そのため，命令合成の際にVLIW命令の規則を違反しないかどうかを確認した後に合成を行う．規則違反が発生する場合にはメイン側のVLIW命令に合成を行わず，次以降のメイン側VLIW命令の空きスロットを埋めるために使用する．

メインおよび投機スレッドで実行する基本ブロック中の各VLIW命令を1命令ごとに合成するが，メインスレッド側のVLIW命令における空きスロットをどの程度投機スレッド側のVLIW命令で埋めることができるかによって，1つのVLIW命令中の並列度も変化する．1つのVLIW命令中の並列度はBBPCにおける性能向上率を測る指標となり，最終的な性能に大きく関わってくる．

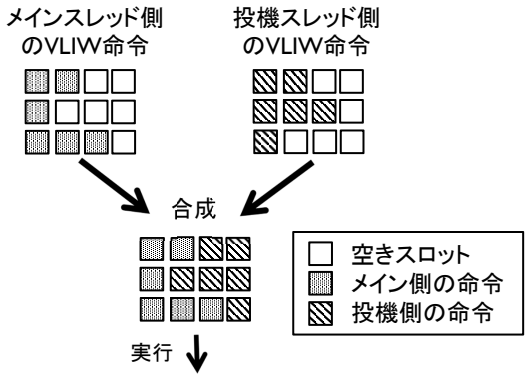

図3: BBPCの命令合成

5 シミュレータによる評価

命令合成による性能向上を調べるため，VLIW命令合成動作を模擬するソフトウェアシミュレータを作成した．VLIW命令合成のシミュレートの結果，出力されたVLIW命令列が元のVLIW命令列と比較して，どの程度メインスレッド側のVLIW命令における空きスロットを埋めることができるのかを調査した．

作成したソフトウェアシミュレータはVLIW命令列を入力データとする．出力データはBBPCのモデルに基づいてメインおよび投機スレッドの各VLIW命令を合成した結果となる．入力および出力データは16進数表現形式のVLIW機械語列である．シミュレータへの入力データにはベンチマークプログラムであるSPEC CINT2000の181.mcfと256.bzip2におけるループ部分のVLIW命令列を使用した．181.mcfは最も実行頻度が多かったループを，256.bzip2は二番目に実行頻度が多かったループを対象とした．対象としたループはBBPCが対象としている非数値処理系プログラム中で実行頻度が多く，かつループ中の基本ブロック数が少ないものと多いものを選択した．

表1がVLIW命令合成によって得られた結果である．どちらのプログラムにおいても，VLIW命令合成前と後を比べるとVLIW命令合成によって実行ステップ数が減少している．このことからループの実行速度が向上し，その分だけ性能向上が期待できる．また，表1の速度向上率はパス実行時における速度向上率である．結果から1.3倍から1.5倍程度の速度向上ができることがわかった．

表1: VLIW命令列合成結果

プログラム	合成前の実行ステップ数	合成後の実行ステップ数	速度向上率(倍)
181.mcf	12	9	1.33
256.bzip2	50	33	1.51

6 おわりに

本稿では，基本ブロックを投機単位とする並列実行手法BBPCの評価のためVLIW命令を合成する動作を再現するソフトウェアシミュレータを開発し，シミュレーション評価を行った．結果からBBPCにおけるVLIW命令合成によって速度向上ができることがわかった．今後はさらにシミュレーションを行う範囲を広げ，投機実行時の動作を模擬することによってBBPCの評価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補助金 (基盤研究 (C)24500055，同 (C)24500054，同 (C)25330055，若手研究 (B)25730026) の援助による．

参考文献

[1] 修 沢坤，横田 隆史，大津 金光，大川 猛，馬場 敬信: “基本ブロックを単位とする投機的マルチスレッドVLIWアーキテクチャ”，情報処理学会第74回全国大会講演論文集，pp.1-81〜1-83，2012.