2パス限定投機システムの提案 – スレッド実行機構 –

米田 淳一† 福田 明宏†† 十鳥 弘泰†† 佐藤 和史† 大津 金光† 横田 隆史† 馬場 敬信†
†宇都宮大学工学研究科情報工学専攻 ††宇都宮大学工学部情報工学科

1 はじめに

近年の半導体集積技術の向上により，1つのチップ上で豊富なハードウェア資源を使用できるようになった．また，性能向上の要求に答えるために，細粒度の命令レベル並列性だけでなく，より大きな粒度のスレッドレベルにおいても並列性を抽出することが求められている．このような背景から，我々は2パス限定投機方式を提案した[1]．これまでに，トレースベースのシミュレーションを用いて2パス限定投機方式の性能評価を行い，2パス限定投機方式の有効性を示してきた．

2パス限定投機方式を実現するにあたって，アーキテクチャの設計や最適化手法の検討を行う必要がある．そのために，2パス限定投機方式を実現するシステムの詳細なシミュレーション環境の構築が必要である．

本稿では，2パス限定投機方式の実行アーキテクチャにおけるスレッド実行機構の設計を行う．

2 2パス限定投機方式

2パス限定投機方式について，具体例を挙げて説明する．なお，以下においてパスとは，プログラム中のどの命令列が実行されるかを示したものをいう．

図1のようなループ構造を持つプログラムがあるとする．この図において丸印は基本ブロック，実線の矢印は制御の流れを示す．ここで，A→B→Fのパスが最も多く，A→C→E→Fのパスが二番目に多く実行されるものとする．これらのパスをそれぞれ#1パス，#2パスと呼ぶ．2パス限定投機方式では，この#1パス，#2パスそれぞれに対して，最適化したコード(以下，投機スレッドコード)をあらかじめ生成しておく．プログラムの実行時，次に実行されるパスの予測が行われ，どちらかのパスの投機スレッドコードが選択され，選択されたコードに対して一つのスレッドが生成，実行される(図2)．2パス限定投機方式では，これを複数のスレッドに対して並列に行うことで，マルチスレッド実行を行い，プログラムを高速化する．

2パス限定投機方式では次に実行される可能性の高いパスを予測し，投機的にスレッドを生成，実行する．そのため，その予測が誤っている場合，プログラムの正しい実行結果を保つために，誤って投機実行されたスレッドを破棄しなければならない．パスが正しく予測されたかどうかは，assert命令により判定する．assert命令は条件分岐命令と同様のテストを行い，条件が不成立である場合(以下，assertの不成立)はスレッドの実行結果を破棄し，回復処理を行った後，適切なパスのコードを実行し直すことで，正しい実行結果を保証する．

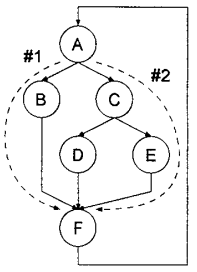

図1: ループ内パス

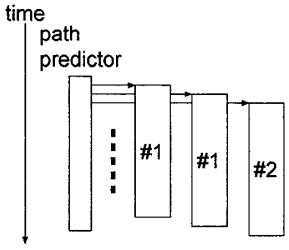

図2: マルチスレッド実行の例

3 2パス限定投機システム PALS

プログラムを高速に実行する2パス限定投機方式を実現するためには，2パス限定投機方式を実現するのにふさわしいアーキテクチャの検討が必要である．本節では，2パス限定投機方式を実現するにあたっての課題と，2パス限定投機方式を実現するアーキテクチャである2パス限定投機システム PALS(PAth Limited Speculation)について述べる．図3にPALSアーキテクチャ全体の概要を示す．

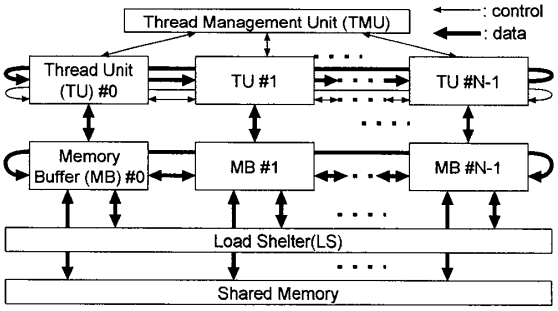

図3: PALSの構成

3.1 パスの予測

2パス限定投機方式はスレッド毎にパスの予測を行い，投機スレッドコードを選択して実行する．パスの予測を行う方式として，先行研究において2レベルパス予測手法および2レベルパス予測器を提案した．この予測方式を実現する手段として，パス予測の集中管理を行う専用のハードウェア TMU(Thread Management Unit)を用いることとした．

3.2 投機的メモリアクセス

投機スレッドが破棄された場合，当該スレッドが破棄されるまでに行ったストアの内容も破棄されなければならず，さらにストアされる前のデータを復元する必要がある．投機スレッドが行うストアを共有メモリに直接書き込んでしまっては，スレッドが破棄された時に，ストアされる前の値を復元することができない．そのため，投機スレッドが行うストアは，スレッドローカルなバッファに書き込みを行い，スレッドの実行が確定

Proposal of Two-Path Limited Speculation System
– Thread Processing Unit –

†Junichi Yoneda, Kazuhumi Satou, Kanemitsu Ootsu, Takashi Yokota and Takanobu Baba.
††Akihiro Fukuda, Jutori Hiroyoshi.
Department of Information Science, Graduate School of Engineering, Utsunomiya University (†)
Department of Information Science, Faculty of Engineering, Utsunomiya University (††)

した段階で共有メモリに書き込む必要がある．このスレッドローカルなバッファを PALS では MB(Memory Buffer) と呼称する．

4 スレッド実行機構

本節では，複数のスレッドを並列に実行するために複数個必要となる，計算を行う主体たるスレッド実行機構について焦点をあてて述べる．以下ではスレッド実行機構を単に TU(Thread Unit) と表記する．1 つの TU に割り当てられるスレッドの個数は 1 つである．

2 パス限定投機方式において，スレッドは個々のループイテレーションを単位とした投機スレッドコードを実行する．スレッド間通信のオーバヘッドを減らすことが，プログラムの高速化につながる．そのためには，2 パス限定投機方式に適したスレッド間通信を実現する必要がある．

4.1 静的データ依存と動的データ依存

ループの各イテレーションをスレッドで並列実行するにあたり，投機スレッドコード内に含まれる命令間の依存について整理する．

静的データ依存とは，静的にすなわちプログラムを実行することなく依存関係を知ることができる依存である．具体的には，命令間のレジスタやアドレスを命令内の即値で指定したメモリアクセスなどがある．

動的データ依存とは，プログラムの実行時に初めてデータの依存関係が発生する依存である．具体的には，レジスタでアドレスを指定したメモリアクセス命令によって生じる．動的データ依存が生じる命令を動的アクセス命令と呼ぶこととする．動的アクセス命令が複数個存在する場合，これらすべての動的アクセス命令間に動的データ依存が存在する．

ループのイテレーションを各スレッドで実行することにより生じる問題点として，上述したスレッド間のデータ依存がある．スレッド間におけるデータ依存を解決するためには 2 通りの方法がある．1 つはデータ依存違反が発生しないようにする方法 (以下，違反回避方式) であり，もう 1 つはデータ依存違反が発生した時にそれまでの実行を破棄する方法 (以下，回復処理方式) である．

これらの方式をどのように採用するか，2 パス限定投機方式の特徴を考慮しながら検討する．

4.2 スレッド間通信の検討

前節で示したデータ依存を考慮したスレッド間通信を実現するために，TU には，スレッド間通信にレジスタ間通信とメモリ間通信の 2 種類を設けることとした．以下に，それぞれのスレッド間通信の手法について検討理由を述べる．

レジスタ間通信

レジスタ間通信とは各 TU 間でレジスタの値を直接送受信する方法であり，違反回避方式を採る通信方式である．

なぜレジスタ間通信を用いたかについて述べる．レジスタに対する演算は静的データ依存のみで構成される．違反回避方式は値の書き込み側と読み込み側が同期を取ることで，データ依存違反を回避する方法である．すなわち，スレッドのアクセス対象に静的データ依存がある場合，同期処理を行うことで先行スレッドにより値が書き込まれてから，後続スレッドが値を読む．この場合，アクセス対象の書き込みを行った直後と読み込みを行う直前で同期待ちを行えばよいため，同期待ちのオーバヘッドは最小限に抑えられる．

一方で，回復処理方式を用いた場合，レジスタはデータ依存違反が頻繁に起こると考えられ，実行効率が悪くなることが予想される．

メモリ間通信

メモリ間通信は TU が持つ MB 同士で通信を行う方式であり，回復処理方式を採る通信方式である．

回復処理方式では，同期待ちを行わずに読み込みを行う代わりに，データ依存違反を検知した場合はスレッドの内容を破棄し，スレッドの開始時の状態まで回復した後にスレッドを再実行する．この方式を実現するためには，ストア命令によって書き込まれる前の情報を保持する機構が必要である．具体例として Superthreaded Architecture[2] の Memory Buffer などがある．

回復処理方式では，ひとたびデータ依存違反が発生した場合は，スレッド開始時の状態まで回復するための処理により，大きなオーバヘッドとなる反面，同期待ちがないため，データ依存違反が発生しない場合は低オーバヘッドでのアクセスが可能である．

ここでレジスタ間通信の特徴を考慮に入れる．レジスタ間通信でやりとりされるレジスタ番号はすべて静的に解析ができたものである．すなわち，静的に解析できるデータはレジスタ間通信を，動的依存のあるデータはメモリ間通信をそれぞれ用いることで，効率的なスレッド間通信が実現できる．

4.3 レジスタ間通信の実装方法

メモリ間通信を実行するにあたって，TU は MB に対してロード/ストア命令を行えばよく，メモリ依存違反の検出などは MB の動作である．

本節ではレジスタ間通信の実装方法について述べる．レジスタ間通信によって値を送信するレジスタ番号をなんらかの形で示す必要がある．投機スレッドコードを生成する際に，wait bit mask という送信するレジスタ番号を示した情報を同時に生成する．

TU はループを実行する際に，この wait bit mask を読み込むことで次のスレッドに送信するレジスタ番号を取得する．

5 おわりに

本稿では，2 パス限定投機方式を実現するアーキテクチャである 2 パス限定投機システムを提案し，その中のスレッド実行機構について示した．今後の課題として，提案した機構によりプログラムの高速化を行えるかの検証がある．

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 (B)18300014，同 (C)19500037，同 (C)20500047）および宇都宮大学重点推進研究プロジェクトの援助による．

参考文献

[1] 横田隆史, 斎藤盛幸, 大津金光, 古川文人, 馬場 敬信: “2 パス限定投機方式の提案”, 情報処理学会論文誌: コンピューティングシステム, 46, SIG 16(ACS-12), 1–13, 2005.

[2] Jenn-Yuan Tsai,Pen-Chung Yew: “The Superthreaded Architecture: Thread Pipelining with Run-time Data Dependence Checking and Control Speculation”, PACT’96, pp.35–46, 1996.