5K-3

スヌープキャッシュをベースにした投機的メモリアクセス機構の検証

関口 祐司 † 十鳥 弘泰 † 大津 金光 † 大川 猛 † 横田 隆史 † 馬場 敬信 †
† 宇都宮大学大学院工学研究科

1 はじめに

投機的マルチスレッド実行をハードウェアで実現するためには, 投機的なメモリアクセスをサポートするメモリアクセス機構が必要になる. 我々は, 投機的なメモリアクセスを実現するスヌープキャッシュをベースにした投機的メモリアクセス機構を提案している [1].

本稿では, 提案するメモリアクセス機構の動作検証を行うため, トレースログを用いたシミュレータによる検証方法について述べる.

2 投機的メモリアクセス機構 PALSVC

本研究で開発しているメモリアクセス機構 PALSVC (PAth Limited Speculation Versioning Cache) は , 2 パス限定投機システム PALS での投機的メモリアクセスの性能を改善するための機構である. PALS とは, プログラムループ中に存在する条件分岐により生じる複数の実行経路 (パス) のなかで, 実行頻度の高い上位 2 本のパスに限定して投機的マルチスレッド実行を行うマルチコアプロセッサシステムである [2].

2.1 PALSVC を用いた PALS のハードウェア構成

PALSVC[1] を用いた場合の PALS のハードウェア構成を図 1 に示す.

PALS のハードウェア構成

図中の TMU (Thread Management Unit) は, 次のパスの予測と, その結果に基づいたスレッドの割り当てを行う. TU (Thread Unit) は汎用プロセッサコアに相当し, プログラムコードを実行する. 全ての TU はリングバスで接続されている.

PALSVC のハードウェア構成

図中の L1 キャッシュには投機実行中のデータ (投機データ) と, スレッドの投機実行後に投機が確定したデータ (確定データ) を格納する. 投機が失敗した場合は投機データのみ無効化し, 投機が成功した場合に投機データを確定データにする. 投機的メモリアクセスでは, 各スレッドが同一アドレスへのストアを実行する毎に新しいバージョンのデータ値が生じ, プログラムオーダ上でのスレッドの位置によって参照すべきバージョンが異なる. そのため, 図中の VCCD (Version Control Cache Directory) を用いてスレッドの親子関係に基づくバージョン管理を行う. VCCD にはアドレス毎にバージョン管理情報を用意し, バージョン情報とデータの依存関係を記憶する. また, VCCD ではデータの依存関係を記憶しているため, メモリ依存違反の検出を行うことができる. これらのモジュールは, 共有バスに接続されている. なお, 共有バスはスプリットトランザクションを前提とする.

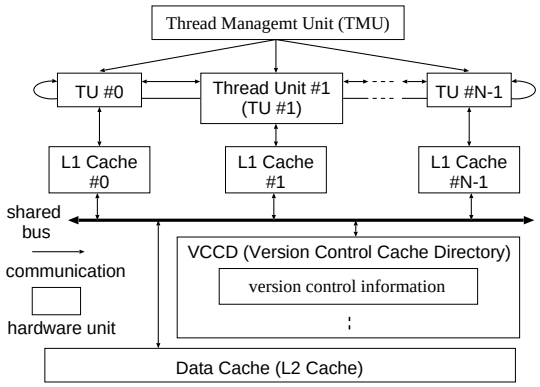
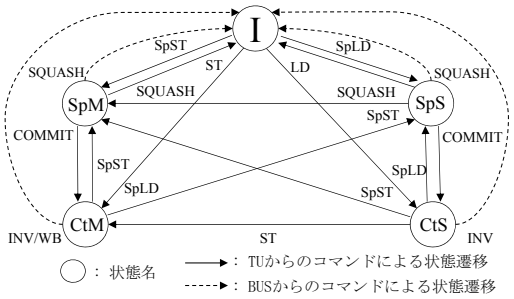


図 1: PALSVC を用いた PALS のハードウェア構成

2.2 PALSVC のキャッシュコヒーレンスプロトコル

PALSVC のキャッシュコヒーレンスプロトコルの状態遷移図を図 2 に示す.

確定データ (CtM 状態, CtS 状態) のラインにキャッシュヒットした場合, ラインのデータをライトバックするか VCCD が判断しキャッシュミスと同様に処理を行う.



状態名	説明
I	キャッシュラインが無効である
SpM	投機的に変更されたデータである
SpS	投機的にデータが読み込まれた
CtM	確定データである
CtS	確定データのコピーである

コマンド名	説明
SpST	投機実行中のストア(投機ストア)
SpLD	投機実行中のロード(投機ロード)
ST	非投機実行中のストア(非投機ストア)
LD	非投機実行中のロード(非投機ロード)
COMMIT	スレッドの投機成功を通知
SQUASH	スレッドの投機失敗を通知
INV	キャッシュラインを無効化
WB	データをライトバック

図 2: キャッシュコヒーレンスプロトコルの状態遷移図

Verification of Speculative Memory Access Mechanism Based on Snoop Cache
†Yuji Sekiguchi, Hiroyoshi Jutori, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota, and Takanobu Baba
Graduate School of Engineering, Utsunomiya University (†)

3 PALSVCの検証

我々はPALSVCが仕様通り動作すること詳細に検証するためにISIS-SimpleScalar [3] を利用して検証用シミュレータを実装する. ISIS-SimpleScalerを用いたシミュレータの通信例を図3に示す. 図中の四角はクラスを意味しており, ハードウェアユニットクラスではキャッシュやプロセッサなどの動作を定義する. ポートクラスではユニット間の通信を定義し, パケットクラスでポート上で通信するパケットを定義する. ハードウェアクラスにPALSVCのキャッシュ・VCCDの動作を定義し, ポートクラス・パケットクラスよりハードウェア間通信を実現する.

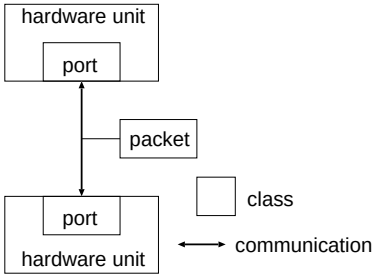


図 3: ISIS-SimpleScalerの通信例

3.1 検証用シミュレータ

検証用シミュレータの構成を図4に示す. 図の四角はクラスを意味しており, L1キャッシュクラス, VCCDクラス, L2キャッシュクラスはPALSVCの各ハードウェアユニットの動作を模擬する. L1キャッシュクラスは検証を行うTU台数分用意する. 共有バスクラスは, 各ハードウェア間との通信を行うポートクラスを定義する. ハードウェアユニット間通信の内容はパケットクラスで定義する. 各ハードウェアユニットは共有バスのポートクラスをスヌープすることで, ハードウェア間通信を実現する.

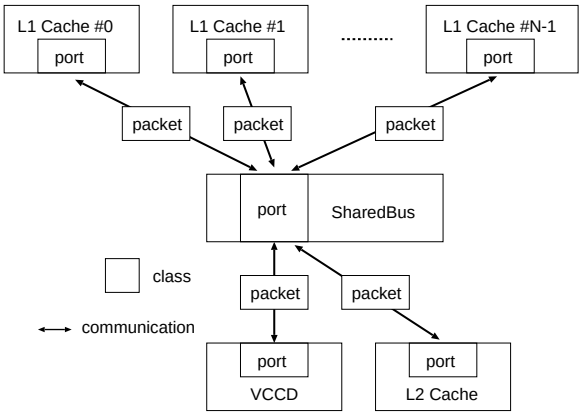


図 4: 検証用シミュレータの構成

検証用シミュレータは, PALSから取得した命令の情報やメモリのデータなどのトレースログをもとにシミュレーションする. トレースログを用いることによりPALSの動作を模擬することができ, PALSVCを用いたPALSの動作検証が可能になる. このシミュレータを使用して, アサーションベース検証とキャッシュのシミュレーションによる検証を行う.

3.2 アサーションベース検証

PALSVCの動作検証の1つとして, アサーションベース検証を行う. アサーションベース検証とは機能検証の1つであり, アサーションとは設計したシステムが満たすべき条件である. 検証用シミュレータに設計したシステムが仕様通りの動作を確認するためのアサーションを挿入し, シミュレーションを行うことで設計したシステムが要件を満たしているか検証を行う.

3.3 シミュレーションによる検証

PALSVCのシステムが正しい動作結果となっているか検証する必要がある. そのためには実際に検証用シミュレータを並列動作し, PALSで実行したプログラムの実行結果と同じ結果になることを確かめなければならない. 並列動作を実現するためには, ISIS-SimpleScalerのパケット通信機能を使用し, アービタレーションを行えるようにする必要がある. PALSVCでの共有バスのアービタレーションはVCCDが行う.

検証用シミュレータではPALSの投機的なメモリアクセスを実現し, 実際のメモリアクセスのデータを扱えるようにする. 投機ストア時に正しい値をキャッシュに書き込むことができることを確認し, 投機ロード時には正しい値を読み込むことができることを確認する.

検証用シミュレータでの実行結果とPALSの実行結果を比較し, 同じ実行結果であれば正しい実行結果であると言える.

アサーションベース検証とシミュレータによる検証を行い, PALSVCがシステムがとして正しいことを証明する.

4 おわりに

本稿では提案した投機的メモリアクセス機構PALSVCを検証するためのシミュレータを使った検証方法について述べた. PALSVCの動作検証は, アサーションベース検証とシミュレーションによる検証を行う.

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) の援助による.

参考文献

[1] 関口祐司ほか, “スヌープキャッシュをベースにした投機的メモリアクセス機構の提案,” 信学技報, Vol.112, No.322, pp.1-6, 2012.

[2] 十鳥弘泰ほか, “2パス限定投機方式を実現するマルチコアプロセッサPALSの提案,” 信学技報, Vol.109, No.319, pp.19-24, 2009.

[3] 藥袋俊也ほか, “ISIS-SimpleScalarの実装,” 情報処理学会研究報告, Vol.112, No.322, pp.29-34, 2004.