2J-4

投機的なメモリアクセスを支援するための キャッシュコヒーレンスプロトコルの検証

関口 祐司 † 十鳥 弘泰 †† 大川 猛 † 大津 金光 † 横田 隆史 † 馬場 敬信 †
† 宇都宮大学大学院工学研究科情報システム科学専攻 †† 宇都宮大学大学院工学研究科システム創成工学専攻

1 はじめに

計算機の速度向上方法の 1 つに, 並列計算機によるプログラムの並列実行が存在する. しかし, プログラムの並列化は, データ依存などにより並列化は困難である. そこで改善策として, プログラムのループ部分の命令列からスレッドを作成し, 複数のスレッドを投機的に実行する投機的マルチスレッド実行がある. 投機的マルチスレッド実行では, マルチスレッド実行中のデータの読み書きはスレッドの実行が確定するまで実行結果に反映することができない. そこで投機的メモリアクセスを効率的に実現する SVC などのメモリアクセス機構の研究が行われている [1].

投機的メモリアクセス機構でキャッシュコヒーレンスプロトコルを使用する場合, 汎用的なマルチコアプロセッサのキャッシュコヒーレンスプロトコルよりも状態数が多く複雑である. そのため投機的メモリアクセス機構を実現するキャッシュコヒーレンスプロトコルを設計する場合, 人手での網羅的な動作検証は膨大な時間が必要になる.

そこで本稿では, モデル検証ツール SPIN を使用してキャッシュコヒーレンスプロトコルの検証を行う. 投機的メモリアクセス機構を実現するキャッシュコヒーレンスプロトコルの検証を目指して, 汎用的なマルチコアプロセッサで使用される MESI プロトコルの検証を行い, SPIN による検証方法を明らかにする.

2 キャッシュコヒーレンスプロトコルの検証

モデル検証ツール SPIN は, Gerard J. Holzman により提案されたモデル検査のためのツールである [2].

SPIN は, 通信プロトコルの設計仕様, 特に振る舞い仕様を検証することを目的にしている. 振る舞い仕様は Promela というモデル記述言語で記述する. SPIN では, 並行プロセスの非決定的な振る舞いを Promela で記述するため, マルチコアプロセッサの設計の検証に使用できる.

マルチコアプロセッサでは, 複数のプロセッサからの同時アクセスにより競合が発生した場合, データの整合性を保つために排他制御を実現させる必要がある. 排他制御を実現させるための留意点としてデッドロックやライブロックが挙げられる. SPIN では, デッドロックやライブロックなどを検出することができる.

マルチコアプロセッサでは, 2 つ以上のスレッドやプロセスなどの実行単位が実行終了を待ち, 次の実行に進めない状況が起きてしまうデッドロックを避けなければならない.

SPIN では Promela で実行可能な文が存在しなくなった場合に, デッドロックとして検出する. しかし, SPIN でデッドロックが検出されなかったとしても, 正しいシステムが設計されたとは限らない.

デッドロックでは, 複数のスレッドが資源獲得のための処理が進行しない. 一方, あるスレッドにおいて資源獲得の処理が進行しているのに関わらず, 他のスレッドが資源の獲得ができない状態が発生する. この状況をライブロックと呼ぶ. ライブロックが発生している場合, あるスレッドが同じ動作を永久に行っている状況が起きている.

SPIN では, 進行性検査という検証を使用することでライブロックの検出を行うことができる. 進行性とは, 無限の実行列において無限回特定の場所に訪れるという性質である. この性質を利用して SPIN では, 特定の実行ポイントを指定し, 無限の実行列を繰り返す処理が, 指定した実行ポイントを頻繁に訪れることを証明する.

本稿で述べる SPIN によるキャッシュコヒーレンスプロトコルの検証は, デッドロックやライブロックが発生しないことを確認することである.

3 MESI プロトコルの検証

SPIN によるキャッシュコヒーレンスプロトコルの検証の例として, MESI プロトコルの検証を行う.

前の節で進行性検査を使用することでライブロックを検出できることを述べた. そこで進行性検査を行うために, 図 1 に示すように無限に状態が遷移する Promela コードを作成した.

```
Ca_mod:
   if
//from bus
   ::from_bus ? invalid -> ...
   ::from_bus ? load -> ...
//from processor
   ::timeout -> ...
   fi
```

図 1: Promela コードによる記述例

Promela ではプロセス単位で SPIN で実行される. そこで記述した Promela コードは, プロセスをキャッ

Verification of MESI Cache Coherence Protocol
†Yuji Sekiguchi, Takeshi Ohkawa, Kanemitsu Ootsu, Takashi Yokota, Takanobu Baba
††Hiroyoshi Jutori
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (†)
Department of Innovation Systems Engineering, Graduate School of Engineering, Utsunomiya University (††)

Copyright ©2012 Information Processing Society of Japan.
All Rights Reserved.

シュプロセスとし検証するプロセッサの数だけ用意する．このキャッシュプロセスでは，MESI プロトコルの各状態を C 言語と同様に動作するラベルと goto 文を使用して状態遷移を行う．

このキャッシュプロセスは，if 文 (選択構文) を使用しており，図のように::の後ろの文が実行可能な文をランダムに選択して実行する．timeout を使用することで，実行可能な文が存在しなかった場合，timeout の後ろの文が実行可能になる．このコードは他のキャッシュプロセス (プロセッサ) からの要求がある場合，図の if 文の下の行の文が実行される．そして実行可能な文がない場合，timeout の後ろの文が実行される．この図では，timeout の後ろの文に自分自身のプロセッサからの動作を記述した．MESI プロトコルにおいて状態遷移する命令は，ストア命令とロード命令のみである．

SPIN を使用した検証は，Promela の記述が正しいことを証明しなければならない．そこで Promela を実行させて仕様通りの通信ができているかを，各キャッシュプロセス間の通信を出力させて動作検証を行った．本稿では，3 コアプロセッサの検証を行った．

4　実行例

ストア命令は，各プロセッサのストア命令実行時にキャッシュにキャッシュラインを割り当ててデータを書き込む．

この動作を行った実行例を図 2 に示す．この図では，キャッシュプロセス間の通信を示している．図中の cache2 プロセスは，bus プロセスに無効化信号 (invalid) を送る．そして，この bus プロセスが他のキャッシュプロセスを無効化信号を送信する．

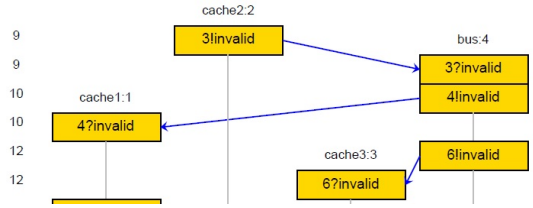


図 2: ストア命令の実行例

ロード命令は，ロードミス時に他のキャッシュに要求キャッシュラインが存在するか確認する．確認した結果に対して，状態が遷移する．また他のキャッシュのキャッシュラインの状態を遷移させる．

この動作を行った実行例を図 3 に示す．図 3 では，cache2 のロードミス時に，bus プロセスにロード要求 (request) を送信する．bus プロセスは，この信号を受信したら各キャッシュプロセスに信号を送信して，キャッシュラインが存在するかを確認する．そしてキャッシュラインが存在したので，存在したことを cache2 に送信して状態を遷移させる．

他のキャッシュプロセスも同様な動作検証を行った結果，仕様通り動作することを確認した．

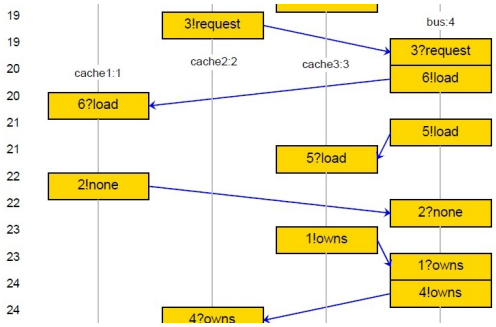


図 3: ロード命令の実行例

5　デッドロック・ライブロックの検出

MESI プロトコルをもとにしたコードを SPIN で検証を行った結果，キャッシュプロセスの実行が停止することはなかった．SPIN では，Promela コードをもとに検証器を作成することができる．この検証器を実行することでデッドロックの検出をすることができる．その検証器の結果からデッドロックは検出されなかった．

また進行性検査を行った結果，ライブロックは検出はされなかった．

以上のように検証を行った結果，MESI プロトコルは仕様通り動作していることを確認できた．

6　おわりに

本稿では，MESI プロトコルの検証をモデル検証ツール SPIN を使用して，仕様通りにプロトコルが動作をしていることの検証を行った．検証の結果，MESI プロトコルではデッドロック・ライブロックが起きないことを証明できた．キャッシュコヒーレンスプロトコルの設計の検証を SPIN で行う場合，進行性検査が有効的であることが明らかになった．

本稿で述べた検証は進行性検査を使用して検証を行った．しかし SPIN では他にもいくつか検証方法が存在する．キャッシュコヒーレンスプロトコルを設計する場合，指定した条件が成立することを検査する検証方法を使用することで，無駄な状態数を削減できる可能性がある．

今後は，投機的メモリアクセス機構を実現するためキャッシュコヒーレンスプロトコルの設計を行い，SPIN で検証を行う．

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)21500050，同 (C)21500049) の援助による．

参考文献

[1] S. Gopal, T. N. Vijaykunmar, J. E. Smith, and G. S. Sohi: “Speculative Versioning Cache”, the Fourth International Symposium on High-Performance Computer Architecture (HPCA’98), pp.195-205, 1998.

[2] Gerard j. Holzmann: “The Model Checker Spin”, IEEE Trans. on Software Engineering, Vol.23, No.5, pp.279-295, May 1997.

Copyright ©2012 Information Processing Society of Japan.
All Rights Reserved.