Vol.2011-SE-172 No.2
Vol.2011-EMB-21 No.2
2011/5/17

ハードウェアの振舞いを考慮したスピンロックのモデル検査

加藤寿和[†1]　一場利幸[†1]
本田晋也[†1]　高田広章[†1]

本研究では，マルチプロセッサシステムで用いられるスピンロックアルゴリズムを複数種類のメモリコンシステンシモデルを含めてモデル化し，メモリバリア命令を加えた際の排他性に関して検証を行った．近年，組込みシステム向けのハードウェア（プロセッサ）においても，高速化のために，Total Store Ordering（TSO）や Partial Store Ordering（PSO）のようなメモリコンシステンシモデルを持つものがある．メモリアクセスの順序がソフトウェアの記述と入れ替わるため，メモリバリア命令を適切に使用する必要がある．モデルを用いて検証を行った結果，適切な位置にメモリバリア命令を挿入した場合に排他性が満たされることを確認できた．PSO のモデルでプロセッサ数が 2 個の条件での検証では，状態数は 1.8×10^9 であった．

Model Checking of the Spin-lock in Consideration of Hardware Behavior

Toshikazu Kato,[†1] Toshiyuki Ichiba,[†1]
Shinya Honda[†1] and Hiroaki Takada[†1]

We verified the exclusiveness of spin-lock algorithm with memory barrier instructions by modeling the algorithm with several kinds of memory consistency models. Recent hardware for embedded systems uses memory consistency models such as Total Store Ordering(TSO) and Partial Store Ordering(PSO). Since in these models the real order of memory access differs from the described order in software, it is necessary to insert memory barrier instruction to the software description appropriately. Our verification models confirm the satisfaction of exclusiveness when memory barrier instruction is inserted appropriately. The number of state was 1.8×10^9 on verifying PSO model with two processors.

1. はじめに

近年，大規模化・複雑化が進む組込みシステムにおいても，マルチプロセッサシステムの重要性が増している．その背景には，消費電力の増大を抑えつつ処理性能の向上を図るためには，クロック周波数を上げるよりも，プロセッサ数を増やしたほうが有利であるという状況がある．特に，複数のプロセッサを 1 つの LSI 上に集積したマルチプロセッサは，処理性能面からも消費電力面からも利点が大きく，広範な組込みシステムへの適用が期待される．

異なるプロセッサ上で動作するプログラム間の排他制御手法としてスピンロックがある．スピンロックを実装するにはメモリアクセスを不可分に行う命令（不可分命令）が必要である．しかし，高速化技術が組み込まれた近年のハードウェアにおいて，スピンロックアルゴリズムを単に不可分命令を用いて実装しただけでは排他制御が実現できない可能性がある．この問題は高速化技術として，メモリアクセスにリラックスコンシステンシモデルを採用した場合に，メモリアクセスの順序がプログラム記述と一致しないために生じる．組込みシステムで多く用いられている ARM プロセッサにおいても，ARMv6 からは，リラックスコンシステンシモデルを用いることが可能となっている．

このようなハードウェア上でスピンロックのプログラムを正しく動作させるためには，メモリバリア命令をプログラム中の適切な位置に挿入する必要がある．しかし，メモリバリア命令の挿入位置を決定することは容易でなく，必要な位置に挿入されないと正しい排他制御が行われない．逆に，不必要な位置に挿入されるとオーバヘッドが大きくなり，リアルタイム性が失われる可能性がある．この問題はキューイングスピンロック[1] のような複雑なスピンロックアルゴリズムになるとさらに難しくなる．検証手法として，実機におけるストレステストが挙げられるが，プロセッサの実行タイミングといったハードウェアの振舞いは無数にあるため，十分な検証を行うことは困難である．組込みシステムには高度な信頼性が求められることが多いため，メモリバリア命令の挿入位置が適切かどうかを確実に検証する必要がある．

ソフトウェアの検証を，形式検証であるモデル検査によって行う各種研究がなされている[2)–4)]．モデル検査を行うことで，システムの振舞いを網羅的に探索することができる．モデル検査ではソフトウェアやアルゴリズムのみをモデル化することが一般的であるが，アル

†1 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University

Vol.2011-SE-172 No.2
Vol.2011-EMB-21 No.2
2011/5/17

ゴリズムのみをモデル化するだけでは，上記のようなハードウェアに起因する問題点を検知することができない．

そこで本研究では，スピンロックアルゴリズムだけでなくハードウェアもモデル化の対象として，スピンロックの排他性に関するモデル検査を実施した．ARMv7 アーキテクチャ[5]のハードウェアを対象とし，モデル検査ツールには SPIN[6] を用いた．システムを全てモデル化すると探索の際に状態爆発が発生する恐れがあるため，スピンロックの動作に関連するハードウェアのみをモデル化した．具体的には，不可分命令，メモリコンシステンシモデル，メモリバリア命令に関連するハードウェアをモデル化の対象とした．ハードウェアモジュール単位でモデルを作成し，それらを組み合わせる方法を取ることでモジュール単位でのモデルの変更を容易に行えるようにした．また，検証するプロセッサ数の変更が容易に行えるようなモデル設計を行った．

2. 対象ハードウェア

本研究で対象とするハードウェアは，ARM Cortex-A9 MPCore[7] による共有メモリ型マルチプロセッサである．ARM Cortex-A9 MPCore は，ARMv7 アーキテクチャである Cortex-A9 を 1 から 4 個まで搭載できるマルチプロセッサである．命令の実行順序を入れ替えるアウト・オブ・オーダ実行とメモリへの書き込み時にライトバッファを利用することができる．これらの利用により，実行効率を上げられるが，メモリアクセスの順序がプログラム記述と一致しない可能性がある．そのため，マルチプロセッサ環境では意図しない結果を得ることがある（シングルプロセッサではこの問題は発生しない）．この問題を解決するために，ARMv7 には，同期のためのメモリバリア命令がある．また，ARMv7 にはメモリへの不可分なリード・モディファイ・ライト操作を行う不可分命令として LL/SC 命令がある．

2.1 メモリコンシステンシモデル

メモリコンシステンシモデルとは，ハードウェアによる実際のメモリアクセス順序を定めたものである．メモリコンシステンシモデルを考える上で関連する内容として，アウト・オブ・オーダ実行とライトバッファがある．

メモリアクセスには，リード（R）とライト（W）の 2 種類があるため，メモリアクセスの順序付けには，R → R，R → W，W → R，W → W の 4 種類が考えられる．ここで X → Y は,「操作 Y が実行される前に，操作 X は完了しなければならない」という順序依存関係を表す．これらの順序付け（R → R，R → W，W → R，W → W）を緩和（relaxed）したメモリコンシステンシモデルをリラックスコンシステンシモデルという．メモリコンシステンシモデルには，その緩和の程度により，**表 1** のように 4 つのモデルが存在する．例えば，TSO は W → R の順序付けが緩和されたモデルであり，W が完了する前に R が完了する可能性がある．メモリコンシステンシモデルによっては，意図した動作を行うためにメモリバリア命令が必要となることがある．

表 1 メモリコンシステンシモデル
Table 1 memory consistency models.

メモリコンシステンシモデル	緩和する順序制約	メモリバリア命令
SC(Sequential Consistency)	なし	必要なし
TSO(Total Store Ordering)	W → R	必要な場合あり
PSO(Partial Store Ordering)	W → R，W → W	必要な場合あり
WO(Weak Ordering)	すべて	必要な場合あり

2.2 メモリバリア命令

メモリバリア命令はメモリコンシステンシモデルに関わらず，メモリアクセスの順序を制限する命令である．ARMv7 には，DSB（Data Synchronization Barrier）命令と DMB（Data Memory Barrier）命令の 2 種類のメモリバリア命令がある．DSB 命令はメモリバリア命令の後続処理を停止して，先行するメモリアクセスを完了させる．後続処理を停止するためプロセッサ内の実行効率は下がるが，メモリ更新が直ちに他プロセッサから観測される状態になるため，システム全体の実行効率は上がる．DMB 命令はメモリバリア命令の次のメモリアクセスまでに，先行するメモリアクセスを完了させる．メモリアクセス以外の後続処理を実行できるためプロセッサ内での実行効率は上がるが，メモリ更新が他プロセッサから観測されるタイミングが遅れるためシステム全体の実行効率は低下する．

2.3 LL/SC 命令

LL（Load-Linked）命令と SC（Store-Conditional）命令は，組み合わせて使用され，不可分にリード・モディファイ・ライト操作を行う命令である．ARMv7 では，LL/SC 命令はそれぞれ ldrex/strex 命令として実装されており，ハードウェア機構であるモニタを用いて実現されている．モニタはメモリアクセスを監視し，メモリ領域の排他状態を保持する．そのためにモニタは OPEN 状態と EXCLUSIVE 状態を持つステートマシンを 1 つのプロセッサに対し 1 つ持ち，排他状態としてタグ付けしたアドレスを保持する．

3. リラックスコンシステンシモデル下でのプログラミング

リラックスコンシステンシモデルではメモリアクセスの順序がプログラムの記述と異なる

Vol.2011-SE-172 No.2
Vol.2011-EMB-21 No.2
2011/5/17

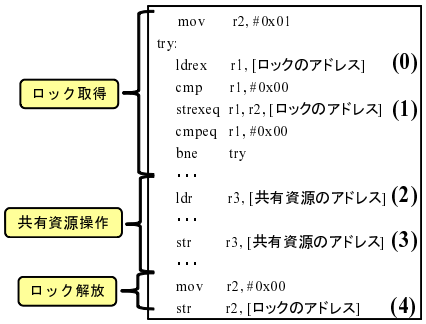


図 1 スピンロックの対象ハードウェア上での実装
Fig. 1 Implementation of the spin-lock on the target hardware.

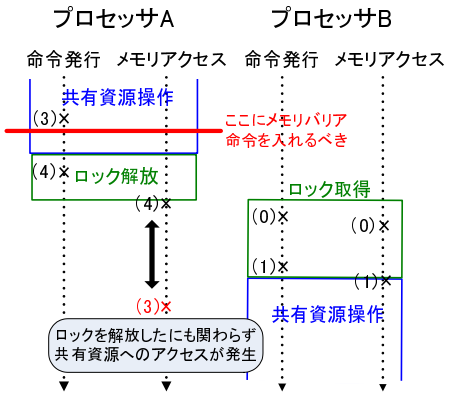


図 2 正しい排他制御が行われない例
Fig. 2 An example that exclusiveness is not satisfied.

ためメモリバリア命令が必要となる場合がある．スピンロックプログラムを用いてこの問題を説明する．以下では，ロックの取得から解放までを排他区間と呼ぶ．

スピンロックはロックを取得できるまで，ロックを繰り返しチェック（ビジーウェイト）するロック方式である．この方式は，排他区間が短い場合に有効であり，リアルタイム OS のカーネル内における排他制御で多く用いられている．スピンロックの実装には，あるプロセッサがロック変数のリード・モディファイ・ライト操作をしている間，他プロセッサからそのロック変数への同時アクセスを排他する必要があるため不可分命令が必要となる．基本的なスピンロックアルゴリズムを，ldrex/strex 命令を用いて実装すると**図 1** のようになる．

図 1 のプログラムを 2 章で述べたハードウェア（アウト・オブ・オーダ実行やライトバッファが実装されたハードウェア）上で実行する場合，正しい排他制御が行われない可能性がある． あるプロセッサ A によるロック取得の strex 命令（図 1（1））と共有資源操作の ldr 命令（図 1（2））が完了する順序が入れ替わり，他プロセッサ B が排他区間にあるにも関わらず，プロセッサ A が共有資源を操作する状況が発生する．また，プロセッサ A が排他区間の処理を実行している場合に，プロセッサ A による共有資源操作の str 命令（図 1（3））とロック解放の str 命令（図 1（4））が完了する順序が入れ替わり，プロセッサ A が排他区間から完全に出ていないにも関わらず，プロセッサ B がロックを取得し，共有資源を操作してしまう状況が発生する．

TSO モデルでは，W → R が緩和されるため，前者が起きる可能性がある．PSO モデル，WO モデルでは，W → R に加えて W → W も緩和されるため，両者とも起きる可能性がある．**図 2** に，後者が起こる場合の処理の流れを示す．

以上の問題を考慮すると，メモリバリア命令が必要と予想される位置は，図 1（1）の命令と図 1（2）の命令の間，図 1（3）の命令と図 1（4）の命令の間と考えられる．この位置にメモリバリア命令を挿入した際に，正しい排他制御が行われるかどうかを 6 章で検証する．

4. モデル化の方針

本研究では，モデルの記述には仕様記述言語である Promela を用いた．モデル化の流れとしては，プロセッサ上で動作するスピンロックアルゴリズムのモデルを作成し，その後，そのモデルに不可分命令，リラックスコンシステンシモデルの要素を順に追加した．ハードウェアモジュール単位でモデルを作成し，それらを組み合わせる方法を取ることで，モジュール単位でのモデルの変更や，プロセッサ上で実行させるプログラムの変更を容易に行えるようにした．また，検証するプロセッサ数を容易に変更できるようにモデル設計を行った．

4.1 スピンロックアルゴリズムのモデル化

スピンロックアルゴリズムについては，アルゴリズムを Promela のプロセス（以降，単にプロセスと呼ぶ）として記述する．本研究では，このプロセスをプロセッサプロセスと呼び，Pi（i=1,2,...）と表す．1 つのプロセッサに対し，1 つのプロセッサプロセスを用意する．また，メモリは全てのプロセッサプロセスからアクセスできるようにする必要があるため，大域変数として表現する．

プロセッサプロセスがメモリアクセスを行う方法として，メモリアクセスを処理する別のプロセスを用意し，そのプロセスに，プロセス間通信を行うチャネル（以降，単にチャネルと呼ぶ）を用いて，プロセッサプロセスが処理を依頼する方法と，プロセッサプロセス内で，直接メモリアクセスの処理を行う方法が考えられる．本研究で対象としたハードウェアでは，プロセッサ外部に，LL/SC 命令を監視するモニタがあるため，前者の方法を主としてモデル化した．一方，後者の方法を用いた場合，プロセス数が減り，状態数を低減できる．そのため，後者についても比較のためにモデルを作成し検証を行った．これらの詳細は次節で述べる．

4.2 不可分命令のモデル化

本研究では，不可分命令のモデルとして，モニタモデルと呼ぶモデル（**図 3**）と，LL/SC

Vol.2011-SE-172 No.2
Vol.2011-EMB-21 No.2
2011/5/17

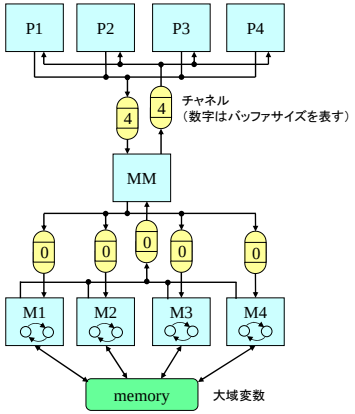
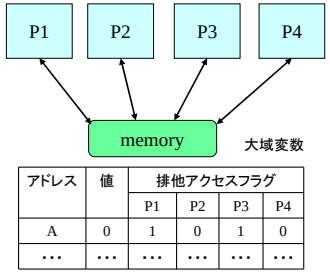


図 3 不可分命令のモデル図（モニタモデル）
Fig. 3 Model chart of atomic instruction (monitor model).



アドレス	値	排他アクセスフラグ			
		P1	P2	P3	P4
A	0	1	0	1	0
…	…	…	…	…	…

図 4 不可分命令のモデル図（LL/SC 命令モデル）
Fig. 4 Model chart of atomic instruction (LL/SC instruction model).

命令モデルと呼ぶモデル（**図 4**）の 2 つのパターンのモデルを作成した．図中の矢印はデータの流れる方向を表す．また，Promela ではチャネルのバッファサイズが 0 の場合に通信方法が同期通信となる．

はじめ，ARMv7 では LL/SC 命令の実現のためにモニタが用いられているため，モニタをモデル化することにより，LL/SC 命令をモデルで扱うモニタモデルを考えた．しかし，このモデルにリラックスコンシステンシモデルの要素を追加したところ，状態数が増え，プロセッサ数を増やした場合の検証時間が膨大となった．そこで，状態数低減のために，モニタをモデル化せずに，LL/SC 命令の機能だけに着目した LL/SC 命令モデルを作成した．

まず，モニタモデルについて述べる．モニタをモデル化する方法としては，各プロセッサのステートマシン全てを 1 つのプロセス内で保持する方法と，各プロセッサのステートマシンをそれぞれ 1 つのプロセス内で保持する方法がある．前者ではプロセッサ数の条件を変更した場合に対応が難しいため，後者を選択した．本研究では，プロセッサのステートマシンを保持するプロセスをモニタプロセスと呼び，プロセッサプロセス Pi に対応するモニタプロセスを Mi（i=1,2,...）と表す．

あるプロセッサのステートマシンの動作には，他プロセッサのメモリアクセス（特に自身が保持するタグ付けされたアドレスへの更新）も関係するため，各モニタプロセス間での通信が必要になる．各モニタプロセスから全てのモニタプロセスへのチャネルを用意する場合，チャネル数が大きくなるとともに，プロセッサ数の変更に柔軟に対応できない．そこで，各モニタプロセスとは別のプロセスを 1 つ用意して，各モニタプロセスから他モニタプロセスへの通信を管理させることとした．本研究では，このプロセスをモニタマネージャプロセスと呼び，MM と表す．モニタマネージャプロセスは，ハードウェアにおけるバスの役割も持っており，各プロセッサプロセスから発行される要求をシーケンシャルに処理する．

次に LL/SC 命令モデルの概要を述べる．LL/SC 命令モデルでは，アドレスごとにフラグを用意し，各プロセッサプロセスの ldrex 命令によるメモリアクセスをフラグで保持する．このフラグは大域変数として宣言する．strex 命令を実行する場合は，フラグを確認し自プロセッサプロセスのフラグが立っている場合のみメモリを更新し，更新したアドレスに関するフラグを全てクリアする．str 命令は無条件でメモリ更新を行い，更新したアドレスに関する全てのフラグをクリアする．LL/SC 命令モデルでは，プロセッサプロセス内で上記の処理を行う．

5. メモリコンシステンシモデルのモデル化

メモリコンシステンシモデルに関しては SC，TSO，PSO の順でモデル化を行った．WO に関しては，他のメモリコンシステンシモデルと同様の方法でモデル化できず，モデル化が困難であると考えられたため，本研究では，WO についてのモデル化は行わなかった．この理由は，6.3 節で述べる．

5.1 Sequential Consistency のモデル化

SC ではメモリアクセスの実行順序がプログラムの記述どおりになればよい．SC のモデル化は，4 章で述べたプロセッサプロセスにおいて，メモリアクセスを発行後にその結果を受信するまで待機することで実現できる．プロセッサプロセスは，結果を受信すると後続の処理に進むことができる．

5.2 Total Store Ordering のモデル化

5.2.1 モデルの構成

TSO では，W → R が緩和される必要がある．**図 5** に TSO を考慮したモデル図を示す．ロード要求がストア要求を追い越す状況を発生させるため，SC モデルにライトバッファの要素を加える．そのために，プロセッサプロセスから発行されるメモリアクセスの要求を受け取り，ストア要求とロード要求の順序を入れ替えるプロセスをプロセッサプロセスとモニタマネージャプロセスの間に，各プロセッサプロセスに対して 1 つずつ設ける．本研究で

Vol.2011-SE-172 No.2
Vol.2011-EMB-21 No.2
2011/5/17

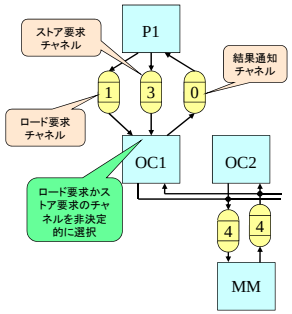


図 5 TSO のモデル図
Fig. 5 Model chart of TSO.

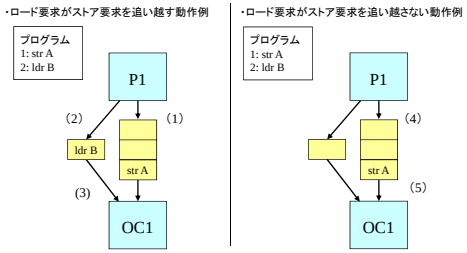


図 6 TSO 上でのメモリアクセスの動作例
Fig. 6 An example of memory access sequence on TSO.

は，このプロセスをオーダコントローラプロセスと呼び，Pi に対応するオーダコントローラプロセスを OCi（i=1,2,...）と表す．また，プロセッサプロセスとオーダコントローラプロセスの通信のために 3 つのチャネルを設ける．プロセッサプロセスからロード要求を伝えるロード要求チャネル，ストア要求とメモリバリア要求を伝えるストア要求チャネル，オーダコントローラプロセスからプロセッサプロセスにメモリアクセスの結果を伝える結果通知チャネルである．ストア要求チャネルがライトバッファをモデル化したものとなる．オーダコントローラプロセスでは，ロード要求チャネルとストア要求チャネルから非決定的に要求を取り出して処理することで，ロード要求がストア要求を追い越す状況を発生させる．チャネルでは，FIFO 型の通信となるためストア要求同士の順序付けは保証される．オーダコントローラプロセスは，モニタマネージャプロセスに受信した要求の処理を依頼する．

5.2.2 提案モデルにおけるメモリアクセスの流れ

ストア命令実行時の処理の流れについて述べる．strex 命令はその結果が必要になるため，プロセッサプロセスはストア要求チャネルに要求を送信後，結果通知チャネルから結果を受信するまで待機する．str 命令はその結果が不要であるため，プロセッサプロセスはストア要求チャネルに要求を送信後，後続の処理を続ける．オーダコントローラプロセスは任意のタイミングで受信した要求を処理する．

今回のモデルでは，strex 命令は実行結果が直ちに必要となるため，strex 命令後のロード命令という順序付けが緩和できなかった．3 章ではスピンロックが排他性を満たさない原因として，strex 命令に続く ldr 命令が，strex 命令に先行して完了してしまうことを述べたが，提案モデルによる検証では，この問題を検知できない．strex 命令発行後に後続処理に進むことができるモデルにするためには，アウト・オブ・オーダ実行の要素をモデル化することが考えられるが，このモデル化を，状態爆発が発生しないように行うことはできなかった．そのため，本研究ではスピンロックが排他性を満たさない原因として，別に考えられるストア要求とストア要求の順序の入れ替えに焦点を当てることとし，strex 命令に関する順序付けの緩和は今後の課題とした．

次に，ロード命令実行時の処理の流れについて述べる．ldr，ldrex 命令にはその結果が必要になるため，プロセッサプロセスはロード要求チャネルに要求を送信後，結果通知チャネルから結果を受信するまで待機する．オーダコントローラプロセスは，ロード要求で指定されたアドレスへのストア要求が，既にストア要求チャネルにあるかどうかを確認する．該当要求がある場合には，そのストア要求の値をロード要求の返り値とする．その際，ロード要求が ldrex 命令ならばモニタマネージャプロセスを通して，モニタプロセスにアドレスのタグ付けを依頼する．該当要求がない場合には，モニタマネージャプロセスにロード要求の処理を依頼する．

以上で述べたメモリアクセスの処理の流れを踏まえ，**図 6** にロード要求がストア要求を追い越す場合と追い越さない場合の動作例を示す．ロード要求がストア要求を追い越す場合の動作例では，はじめに P1 がストア要求を送信する（図 6（1））．OC1 がその要求を処理する前のタイミングで，P1 がロード要求を送信する（図 6（2））．そして，OC1 がロード要求を選択し処理する（図 6（3））と，ロード要求がストア要求を追い越す．次に，ロード要求がストア要求を追い越さない場合の動作例では，はじめに P1 がストア要求を送信する（図 6（4））．P1 がロード要求を送信する前に，OC1 がストア要求を処理する（図 6（5））と，ロード要求はストア要求を追い越さない．

5.2.3 メモリバリア命令

DSB 命令はその命令の後続処理を停止して，先行するメモリアクセスを完了させる命令である．プロセッサプロセスは，ストア要求チャネルへ DSB 要求を送信後，その完了通知が返るまで待機する．オーダコントローラプロセスはストア要求チャネル内の要求を FIFO 順に処理していき，DSB 要求を受信後，結果通知チャネルを通して完了を通知する．DSB 要求を受け取るときにはチャネル内のストア要求は全て処理されている．よって，プロセッサプロセスが完了通知を受信した時には先行するメモリアクセスは完了している．プロセッサプロセスは完了通知を受信後に後続処理を再開する．

DMB 命令は，その命令の次に行うメモリアクセスまでに，先行するメモリアクセスを完

Vol.2011-SE-172 No.2
Vol.2011-EMB-21 No.2
2011/5/17

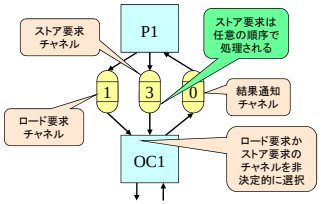


図 7 PSO のモデル図
Fig. 7 Model chart of PSO.

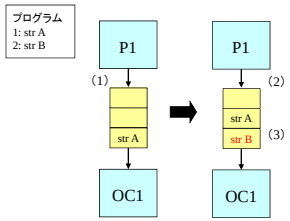


図 8 PSO 上でのメモリアクセスの動作例
Fig. 8 An example of memory access sequence on PSO.

了させる命令である．DSB 命令と異なり，プロセッサプロセスはチャネルへ DMB 要求を送信後に後続の処理を続けることができる．この際，プロセッサプロセス側で DMB 命令発行を表すフラグを立てる．プロセッサプロセスはメモリアクセス命令がある度にそのフラグを確認する．もし，メモリアクセス処理時にフラグが立っていることを確認した場合，メモリアクセスを行わずに DMB 命令の完了通知を受信するまで待機する．オーダコントローラプロセス側の動作は DSB 命令の時と同じである．DMB 要求を受け取るときには，チャネル内のストア要求は全て処理されているので，DMB 要求受信後，結果通知チャネルを通して完了を通知する．完了通知を受け取ったプロセッサプロセスはフラグをクリアし，停止していた後続のメモリアクセス処理を行う．

5.3 Partial Store Ordering のモデル化

PSO では，W → R の緩和に加え，W → W を緩和する必要がある．そこで，前節で述べた TSO のモデルを拡張し，ストア要求がストア要求を追い越す状況を発生させる．そのために，ストア要求チャネルのメッセージ処理を FIFO 順ではなく任意の順序で処理させる．Promela のチャネルへの送信メッセージは，チャネルにおけるバッファの最後に追加される．これを任意の順序で処理されるようにするには，チャネルへの順序付けメッセージ送信機能を利用する．ストア要求を送信する際に，任意の数字を Promela の非決定性処理を用いて選択しメッセージに追加する．順序付けメッセージ送信を行うとメッセージに追加した数字についての辞書的順序付けが行われる．これにより非決定的にメッセージ順序の入れ替えが起こる．メモリバリア要求を送信する際には，チャネルにおけるバッファ内のストア要求を追い越すとメモリバリア命令の仕様と異なるため，バッファの最後に追加する．

図 7 に PSO を考慮したモデル図を示す．TSO のモデルと基本的に同じだが，プロセッサプロセスがストア要求をチャネルに送信する際の動作が，前節で述べた動作と異なる．また，PSO のモデルでも TSO のモデルと同様の理由で，strex 命令後のロード/ストア命令という順序付けを緩和できなかった．

ストア要求を任意の順序でストア要求チャネルに送信する際に，同じアドレスへのストア要求がバッファ内にある場合，その要求を上書きする．この処理はストア要求をストア要求チャネルに送信する際にプロセッサプロセスが行う．

図 8 に，ストア要求がストア要求を追い越す動作例を示す．はじめに，P1 がアドレス A へのストア要求を送信する（図 8（1））．その要求が処理される前に，P1 がアドレス B へのストア要求を送信する（図 8（2））．その際，順序付けメッセージ送信を行うことによりチャネルのバッファ内でアドレス A へのストア要求の追い越しが起こる（図 8（3））．

表 2 検証を実施したモデル
Table 2 Models that we verified.

モデル名	不可分命令のモデル	メモリコンシステンシモデル
MON-SC	モニタモデル	SC
MON-TSO	モニタモデル	TSO
MON-PSO	モニタモデル	PSO
LLSC-SC	LL/SC 命令モデル	SC
LLSC-TSO	LL/SC 命令モデル	TSO
LLSC-PSO	LL/SC 命令モデル	PSO

6. SPIN による検証

6.1 概　　要

設計したモデルを用いて，スピンロックの排他性に関する検証を実施した．検証にはモデル検査ツール SPIN を用いた．検証を実施したモデルは**表 2** のとおりである．

不可分命令のモデルとして，状態数低減のために LL/SC 命令モデルを設計したが，このモデルでは，メモリバリア命令のうち，DMB 命令の要素を取り去った．DSB 命令と DMB 命令はソフトウェアから見ると実行効率が違うだけであり，メモリアクセスの順序を制御する点では同じためである．

実施した検証について説明する．表 2 の MON-TSO，MON-PSO，LLSC-TSO，LLSC-PSO モデルのそれぞれについて，メモリバリア命令を挿入した場合と挿入しなかった場合とで，スピンロックの排他性が満たされるかどうかを検証した．モデルの各プロセッサプロセスでは，スピンロックプログラムをインタリーブさせながら非決定的なタイミングで実行させる．そのプログラム中にメモリバリア命令を挿入する．

Vol.2011-SE-172 No.2
Vol.2011-EMB-21 No.2
2011/5/17

メモリバリア命令を挿入する位置について説明する．3 章で，リラックスコンシステンシモデルを考慮した場合に，予想されるメモリバリア命令の挿入位置を示した．TSO モデルでは図 1（1）と図 1（2）の間に，PSO モデルではそれに加えて図 1（3）と図 1（4）の間にメモリバリア命令が必要と予想された．しかし，設計したモデルでは strex 命令の順序付けが緩和されないため，strex 命令が用いられる図 1（1）と図 1（2）に関するメモリアクセスの順序は入れ替わらない．そのため，メモリアクセスの順序が入れ替わることにより問題が発生する位置は，図 1（3）と図 1（4）の間であると予測される．よって，今回の検証ではメモリバリア命令を図 1（3）と図 1（4）の間に挿入する場合としない場合を検証する．

また，表 2 の各モデルについて，プロセッサ数と検証時の状態数の関係を調査した．なお，6.2 節で述べるが，PSO モデルにおいてはメモリバリア命令を挿入しなければ排他性が満たされない．排他性が満たされない場合，SPIN は状態探索を終了するため，PSO モデルではメモリバリア命令を挿入した場合で状態数を調査した．

検証には，3.33GHz の 6 コアプロセッサ（Intel Xeon X5680），および，64GB のメモリを搭載したマシンを用い，検証ツールには，Spin Version 6.0.1 を用いた．なお，全ての検証において，SPIN による半順序簡約技法[8] を用いた．半順序簡約技法は状態探索時のメモリ使用量を削減するための技法である．

6.2 結　　果

排他性に関する検証結果を**表 3** に示す．状態数はモデルの振舞いの状態数を表し，使用メモリ量と検証時間は状態空間を全数探索する際の使用メモリ量と経過した時間を表す．ここで，使用メモリ量と検証時間は探索時のオプションにより変動するものであり参考程度に記載した．SPIN は偽と判定した時点で探索を終了するため，結果が偽のときの状態数，使用メモリ量，検証時間は記載していない．排他性の結果が記入されていない検証条件では 24 時間以内に検証が終了しなかったため，状態爆発とみなし探索を行ったところまでの状態数を示した．

各モデルについて，プロセッサ数と検証時の状態数の関係を**図 9** に示す．状態数の軸には対数目盛を用いている．状態爆発を起こし検証が終了できなかったものは示していない．状態数の値は検証する内容や挿入するメモリバリア命令の数により異なった．そのため，今回の検証条件では終了したものでも，検証内容や挿入するメモリバリア命令の数によっては状態爆発を起こす可能性がある．

6.3 考　　察

表 3 より，MON-TSO，LLSC-TSO モデルではメモリバリア命令を挿入しなくても，スピンロックによる排他性を満たすことが確認できた．MON-PSO，LLSC-PSO モデルでは MON-PSO で DMB 命令を挿入した場合を除いて，メモリバリア命令を挿入しない場合にスピンロックによる排他性を満たさないこと，メモリバリア命令を挿入した場合に排他性を満たすことを確認できた．SPIN は偽と判定した場合，反例となる実行経路を出力する．ス

表 3　排他性に関する検証結果
Table 3　Result of verification about the exclusivity.

モデル名	プロセッサ数	メモリバリア命令	排他性	状態数	使用メモリ [MB]	検証時間 [s]
MON-TSO	2	なし	真	3414249	94.22	10.6
		DSB 命令	真	2038046	55.73	6.54
		DMB 命令	真	2200633	60.52	6.94
MON-TSO	3	なし	-	*2176010400	-	-
		DSB 命令	真	2205580300	56348.83	31500
		DMB 命令	-	*2251678400	-	-
MON-PSO	2	なし	偽	-	-	-
		DSB 命令	真	1879319700	56041.93	104000
		DMB 命令	-	*1868281600	-	-
LLSC-TSO	4	なし	真	85200568	1445.82	257
		DSB 命令	真	69110997	1166.48	214
LLSC-PSO	3	なし	偽	-	-	-
		DSB 命令	真	29604016	504.65	77

*は状態爆発により検証を取りやめた時点の状態数

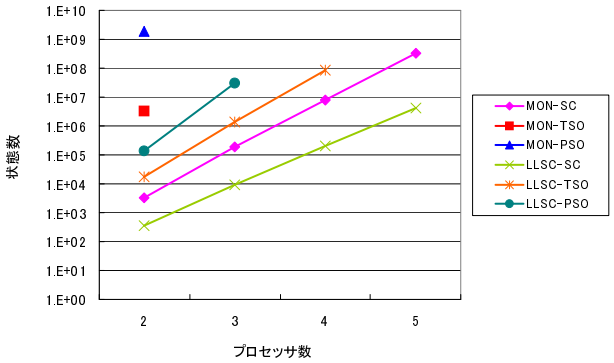


図 9　各モデルのプロセッサ数と状態数の関係
Fig. 9　Relations between the number of processors and the number of states for respective models.

Vol.2011-SE-172 No.2
Vol.2011-EMB-21 No.2
2011/5/17

ピンロックが排他性を満たさない場合の実行経路を分析したところ，3 章で予想したとおりメモリアクセスの順序が入れ替わることで，あるプロセッサが排他区間から出ていないにも関わらず，別のプロセッサが排他区間内の処理を開始する状況が起きていた．

また，表 3 からメモリバリア命令を挿入すると状態数が減少することが分かる．これはメモリバリア命令の挿入により，メモリアクセスの順序の入れ替えが発生しなくなり，モデルの振舞いにおける非決定的な要素が減少したためと考えられる．

今回，LLSC-PSO モデルで検証が終了したプロセッサ数は 3 つまででであった．状態数を減らすためには，システムの本質を失わない範囲でのさらなる抽象化を行うか，別の検証ツールを用いることが考えられる．SPIN 以外の検証ツールとして SMV[9] がある．

スピンロックによる排他性に関する検証結果は予想どおりの結果となったため，今回設計したモデルは妥当であると考える．しかし，設計したモデルにおける strex 命令の順序付けの緩和や，WO を扱うためのモデルの拡張は困難であると考えられる．設計したモデルでは ldrex 命令，ldr 命令，strex 命令の実行直後にその結果が必要となるため，プロセッサプロセスは後続の処理を停止する必要があった．このため，ldrex 命令，ldr 命令，strex 命令に関して順序付けの緩和ができなかった．そこで，strex 命令の順序付けの緩和や WO を実現する方法として，アウト・オブ・オーダ実行の要素をモデル化することが挙げられる．しかし，Promela でこれをモデル化した場合，命令の入れ替えのために命令セットの情報を保持する必要があり，探索中に状態爆発が発生すると考えられる．状態爆発に対しては，検証を行う前に命令の実行順序を変換する外部ツールの開発を検討している．外部ツールでは検証したいプログラムのアセンブリ記述をアウト・オブ・オーダ実行を考慮した実行順序に変換する．そのプログラムをプロセッサプロセス上で実行させ検証する．

本研究では，ハードウェアとして ARM Cortex-A9 MPCore を対象としたが，ハードウェアをモデル化する際に，一般性を失わないようにその動作に着目してモデル化した．よって，今回のモデル化手法は他のハードウェアにも適応できると考えられる．

7. おわりに

本研究ではスピンロックアルゴリズムだけでなくハードウェアもモデル化の対象として，スピンロックの排他性に関するモデル検査を行った．検証の結果，メモリバリア命令が必要と予想される位置に命令を挿入しなかった場合に排他性が満たされないこと，挿入した場合は排他性が満たされることを確認できた．MON-PSO モデルでは，プロセッサ数が 2 個の条件で検証を終了することができ，状態数は 1.8×10^9 であった．

今回設計した MON-PSO モデルでは，プロセッサ数を 3 個とすると状態爆発が発生し，検証を終了することができなかった．そのため，モニタを簡略化したモデルを作成し検証したが，大きな改善は得られなかった．今後は対象システムの本質を失わない範囲において，さらなる抽象化が必要である．また今後の予定として，WO のモデル化，strex 命令に関する順序付けの緩和，他のスピンロックアルゴリズムにおける検証がある．

参 考 文 献

1) 高田広章，坂村　健：中断可能なキューイングスピンロックアルゴリズム，電子情報通信学会論文誌. D-I, 情報・システム, I-コンピュータ， Vol.78, No.8, pp.661–669 (1995).
2) Cattel, T.: Modelization and verification of a multiprocessor realtime OS kernel, *Proceedings of the 7th IFIP WG6.1 International Conference on Formal Description Techniques VII*, Chapman & Hall, Ltd., pp.55–70 (1995).
3) 青木利晃，山崎真吾：モデル検査によるリアルタイムオペレーティングシステムの設計検証，組込みシステムシンポジウム，pp.159–166 (2008).
4) Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H. and Winwood, S.: seL4: formal verification of an OS kernel, *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, ACM, pp. 207–220 (2009).
5) ARMLtd: ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition, DDI0406B.
6) Spin - Formal Verification: "http://spinroot.com/spin/whatispin.html".
7) ARMLtd: Cortex-A9 MPCore Technical Reference Manual Revision: r2p2, DDI0407F.
8) Holzmann, G. J.: *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley (2004).
9) The SMV Systemy: "http://www-2.cs.cmu.edu/modelcheck/smv.html".