3N-7 ITRON-MP におけるデッドロック回避の手法 *

辰巳 将司† 高田 広章‡ 石川 知雄† 宮内 新†
武蔵工業大学§ 豊橋技術科学大学¶

1 研究の背景と目的

高性能なリアルタイムシステムでは，多数の入出力装置がシステムに接続され，それらからの事象に対して決められた時間内にレスポンスを返す必要がある．また，システムに接続する入出力装置の増設などにより，プロセッサ数を増やす必要がある場合，システムの設計を最小限にとどめたいという要求がある．

本研究の対象である，μ ITRON3.0 仕様に準拠したリアルタイム OS である ItIs（ITRON Inprimentation by Sakamura Lab）のマルチプロセッサへの対応（ItIs/MP）も上記のような，リアルタイム性，拡張性に重点を置いて行われてきた [1]．もちろん拡張性，リアルタイム性の向上は大切だが，信頼性を保つことは必須である．最低でも，致命的な故障（デッドロック等）は絶対に避けなければならない．

ItIs/MP では，マルチプロセッサに対応させるにあたり，シングルプロセッサのアルゴリズムを継承している為にデッドロックが生じるケースが存在する．本研究では，上記のように，デッドロックが存在するケース着目し，デッドロックが生じる過程を明らかにし，その対策を検討し，評価することを目的とする．

2 デッドロックが生じる仕組み

デッドロックが生じるのは，ロックを 2 重にとる際に，いくつかの特殊な命令がその他の通常の命令とは異なる順序でロックをとる為であり，この 2 つの命令が同時に実行された場合にデッドロックが生じる．

通常の命令 (`sig_sem` 等) が

```
A(semephore)  ->  B(task)
```

の順でロックをとるのに対して，`rel_wai` は

```
B(task)  ->  A(semephore)
```

の順でとるので，この命令が同時に発行された場合にデッドロックが生じる可能性がある．

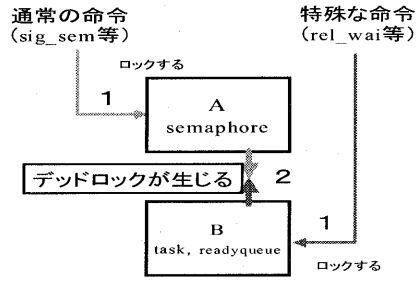

図 1: デッドロックが生じる場合

3 回避策

上記デッドロックの回避策としては，(B) をロックしたあとにタスク状態を見て記憶し，その後アンロック (B) してから (A) をロックする．そして (B) をロックする際，タスク状態が変わっていたらリトライし，変わっていなければそのままロックし，その後 (B)，(A) とアンロックする（図 2）．このように処理を行えばデッドロックは回避できる．

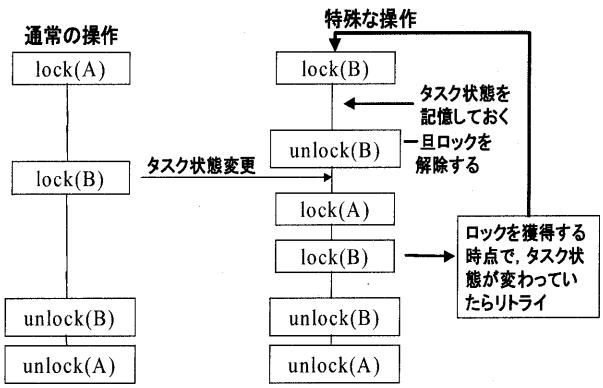

図 2: 回避策

4 効率を良くするために

上記のリトライをするという方法は，デッドロックは回避できるがリアルタイム性を損なわないとは言い切れない．そこで基本的な考えは変えずにリトライしない方法を考察してみる．

---

*The technique of the deadlock evasion at ITRON-MP
†Masashi Tatsumi,Hiroaki Takada,Tomo Ishikawa, Arata Miyauchi
§Musashi Institute of Technology
¶Toyohashi University of Technology

4.1 タイムアウト処理の場合

はじめにタスクをロックし，そのタスクがセマフォキューにつながっていた場合には一旦タスクのロックを解除する．そしてセマフォをロックし，タスクをロックする．ここで他の処理が割り込んでいたかを調べるためセマフォキューにまだつながっているかをチェックする．まだつながっているのならばそのままタイムアウトの処理（待ち解除）してロックを解除する．もしすでにセマフォキューにはつながっていなかった場合は，その命令はすでに他の何らかの処理によって待ち状態を解除されているということなので，何もせずにそのままロックを解除して処理を終了する．このように処理することによりリトライする必要が無くなる．

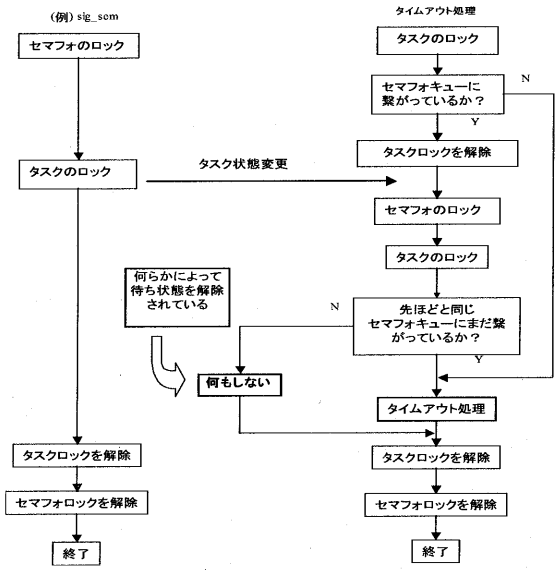

図 3: タイムアウト処理

4.2 rel_wai (待ち状態の強制解除命令) の場合

これもタイムアウト処理の場合とほとんど変わりない．

もし，何らかの処理に割り込まれ，キューから外れた場合は何もせずに処理を終了させればよい．

4.3 chg_pri (優先度変更命令) の場合

この命令の場合は，リトライを回避するには他の部分を変更する必要がある．

- TCB（タスクコントロールブロック，タスク情報が示されている）に ’優先度を xx に変更したい’ というフラグ（優先度変更保留フラグ）を追加する．
- 待ち状態を解除するときにも優先度変更の処理をするようにする．

この 2 つを追加することにより，次のように処理をすることが出来る．

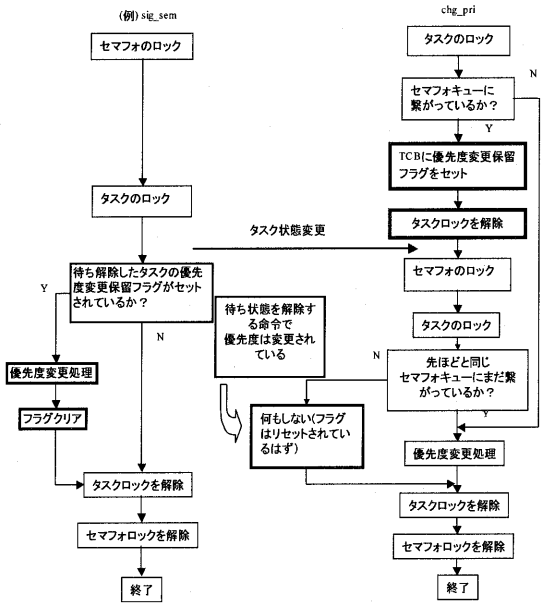

図 4: 優先度変更処理

図のようにはじめにタスクのロックを解除するときに chg_pri フラグをセットしておき，その後セマフォキューにまだ繋がっているかをチェックしたときに繋がっているのならば，フラグを戻して優先度変更の処理を進めてロックを解除して終了する．もし繋がっていなかったのならば，そのときは他の，待ち状態を解除する命令の方で優先度が変更されているはずなので，こちら側では何も処理をしないでロックを解除する．

5 おわりに

本発表では，従来のデッドロックの生じる過程を明らかにし，それに対する回避策を示してきた．リトライしない方法を用いることにより，スループットの低下を抑えつつ，デッドロックの回避を行うことができる．今後の方針としては，このアルゴリズムの実装，リトライする方式としない方式での性能比較，等が挙げられる．

参考文献

[1] 大畠　祐一: “マルチプロセッサ環境におけるマイグレート可能タスクの導入”，武蔵工業大学修士論文,2000