A-14

動的変更機能を有する組込みシステム向けメモリ保護機構
Memory Protection Mechanism with Dynamic Changes for Embedded Systems

山田 晋平†　　中本 幸一†‡
Shimpei Yamada　Yukikazu Nakamoto

１．はじめに

近年，組込みシステムの大規模化に伴いプログラムが複雑になり，プログラムの想定外の動作が他のプログラム領域に影響を与え，信頼性が低下する問題が起きている．このような，想定外の動作によるアクセスを防ぐための手段として，メモリ保護機構の利用が挙げられる．

PC やサーバといった汎用システムは，プロセッサのメモリ管理ユニット(MMU：Memory Management Unit)によるメモリ保護の機構を備えている[1]．Linux における典型的なメモリ保護では，特権レベルと非特権レベルの 2 レベルを用いることで，カーネル領域を保護する．各プロセスの領域は複数の非特権レベルの多重仮想空間によって保護される．

組込みシステムでは，保護すべき資源が OS からミドルウェアやアプリケーションにまで分散する傾向にあるため，それぞれの領域に対する保護が必要である．

本研究では，組込みシステムにおいて，一つのモジュールの想定外の動作が，他のモジュールやシステム全体に与える影響を抑えることを目的とし，組込みシステムの制約を考慮したメモリ保護機構の設計・開発を行う．また，ARM9，SH3 の各プロセッサに対する実装・評価を行う．

２．関連研究

文献[2]では，プロセス内部に拡張コンポーネントのための粒度の細かいメモリ保護機構を用いることを提案している．これはプロセス内の拡張コンポーネントに異なる保護特性(読込み，書込み，実行)を設定できるものである．

Nook では，Linux のデバイスドライバのバグにより，他の領域へ書き込むことによる誤動作を防ぐため，デバイスドライバをカーネルアドレス空間内で，異なる保護機能を有する低い特権レベルで動作できるようにしている[3].

組込みシステムでは，メモリ保護機能を提供するμITRON4.0/PX 仕様[4] と，その実装である IIMP カーネル[5] がある．これは複数のタスクをまとめて保護リージョンに置き，保護リージョン間のタスクをアクセスをできないようにしている．保護リージョンは非特権レベル・特権レベルの両方に置くことができる．

３．組込みシステムで求められるメモリ保護

組込みシステムでメモリ保護機構を用いる場合に考慮すべき点を以下に述べる．

3.1　複数領域の保護

1 章で述べたとおり，組込みシステムでは，OS のみならず，アプリケーションまでもが，保護対象である資源を直接管理することが多い．そのため，それぞれの領域を保護できることが求められる．

3.2　特権レベル領域の保護

特権と非特権の 2 レベルを用いた保護では，非特権レベルのプログラムから特権レベルの OS を呼び出すために，時間のかかるシステムコールトラップを用いる必要がある．これは時間制約の厳しい組込みシステムでは大きなオーバヘッドとなる．そこで，特権レベル内にもう一つのレベルを設け，従来の特権レベルを L1，新たに設けたレベルを L2，非特権レベルを L3 とする．L1 と L2 は同じ特権レベルに存在するため，システムコールトラップと比べて高速な関数呼出しで互いを呼び出すことができる．

リアルタイム性の求められるプログラムは L2 に配置し，ダウンロードされるプログラムなど OS に影響を与える可能性があるものは L3 に配置するものとする．

L2 のプログラムは L1 に配置される OS に影響を与えないことを前提とするが，想定外の動作により影響を与えてしまうことが考えられる．そのため，L1 と L2 の間でもメモリ保護が必要となる．

４．組込みシステム向けメモリ保護機構

提案するメモリ保護機構に求められることは，プログラム実行時に必要最小限のメモリ領域をアクセス可能とする一方で，組込みシステム特有の分散した資源管理に対応したアクセス制御を行うことである．さらに，こうした制御を特権レベルでも利用可能とすることにより，非特権レベルの実行時に発生する，システムコール呼出しのオーバヘッドをなくすことである．

4.1　リージョン

提案するメモリ保護機構では，リージョンと呼ぶ単位を用いて保護を行う．リージョンとは，同じアクセス保護を有する複数のプログラムをまとめ，それらのプログラムをメモリ上に配置にするのに必要な情報と合わせたものである．

リージョンはその種類に関わらず複数定義できる．システム内のすべてのプログラムは，定義されたリージョンのどれか一つに所属させる．このようにすることで，保護をリージョン単位で行うことができる．

4.2　リージョン間の関係

提案するメモリ保護機構では，あるリージョンから別のリージョンに対して，読込み，書込み，実行のアクセスがそれぞれ可能であるかどうかを示す，リージョン間の関係を用いる．メモリ保護機構は，このリージョン間の関係に基づいて，各リージョンに含まれるプログラムの保護を行う．

† 兵庫県立大学大学院 応用情報科学研究科
Graduate School of Applied Informatics, University of Hyogo
‡ 名古屋大学大学院 情報科学研究科 組込み研究センター
Center for Embedded Computing Systems, Graduate School of Information Science, Nagoya University

4.3 メモリ保護機構の動作

提案するメモリ保護機構では，現在のリージョンから，別のリージョンのプログラムを呼び出すとき，保護の変更を行う．保護の変更には changeProt と呼ぶ保護変更ルーチンを用い，リージョンと，リージョン間の関係に基づいて作られるメモリ保護情報を参照して，遷移先リージョンに対応した保護に変更する．

5. 実装

提案するメモリ保護機構ではメモリ保護を実現するために，プロセッサの持つ MMU を用いる．以下に MMU と，MMU を用いた保護の実装方式について述べる．

5.1 MMU

MMU とは，仮想アドレスから物理アドレスのアドレス変換情報，アクセス保護情報に基づくメモリの保護を行うものである．これらの情報はページテーブルと呼ぶテーブルにて管理される．また，MMU を持つプロセッサには，アドレス変換のバッファである TLB(Translation Lookaside Buffer)を持つものがある．この TLB の内容をソフトウェアで操作できるプロセッサは，メモリの管理を柔軟に行える．TLB の入替えをハードウェアで実現するプロセッサでは高速に入替え処理を行えるが，ソフトウェアによる TLB の操作は特定の TLB エントリの無効化などに限られる．

5.1.1 ARM9 の MMU

ARM9 プロセッサでは、TLB の入替えはハードウェアが行う．また，ARM9 の MMU にはドメインという機能がある．ドメインは 0 から 15 の 16 個が存在し，ページテーブル内の各ページをいずれか一つのドメインに所属させることができる．各ドメインに対して，アクセス保護情報を参照した保護を行うか，参照せずにアクセスを許可または禁止させるかを指定することができる．

5.1.2 SH3 の MMU

SH3 プロセッサでは TLB はソフトウェアで管理し，専用のレジスタを介してアクセスできる．また，アドレス空間識別子(ASID：Address Space IDentifier)により複数のプロセスを区別し，排他的に使用される仮想アドレス空間をお互いに保護することができる．

5.2 実装方式

提案するメモリ保護機構では，組込みシステムで用いられている多くのプロセッサに対応するため，複数の実装方式を示し，実装対象のプロセッサの MMU が持つ機能に応じて，実装方式を選べるようにしている．本稿で提案している実装方式を以下に示す．

5.2.1 ページテーブル切替え方式（実装方式 1）

ページテーブル切替え方式の概要を図 1 に示す．この方式では各リージョンに対してページテーブルを一つずつ用意し，リージョンを遷移するごとに，参照するページテーブルを切り替える，あるいは TLB のエントリを遷移先に応じて書き換える．

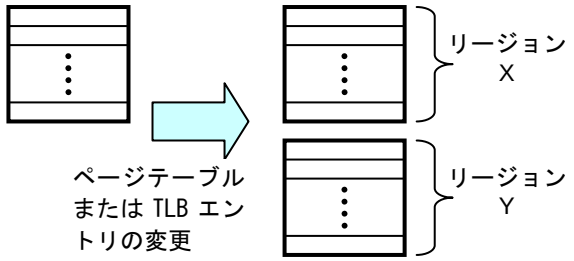

図 1　ページテーブル切替え方式の概要

5.2.2 仮想アドレス空間経由方式（実装方式 2）

仮想アドレス空間経由方式では，一つのページテーブルに全リージョンのアドレス変換情報とアクセス保護情報を収める．これにより，ページテーブルの切替えの必要がなくなる．ただし，実行中のリージョンから他のリージョンに直接アクセスできてしまうため，実行中のリージョンからアクセス不可能なリージョンへのアクセスを制限する仕組みが必要となる．

図 2 では，各リージョンが用いるメモリ空間を ARM9 のドメインに対応付け，リージョン遷移の際にアクセス可能なドメインをメモリ保護機構が変更することで，必要なリージョンにのみアクセスできるようにしている．

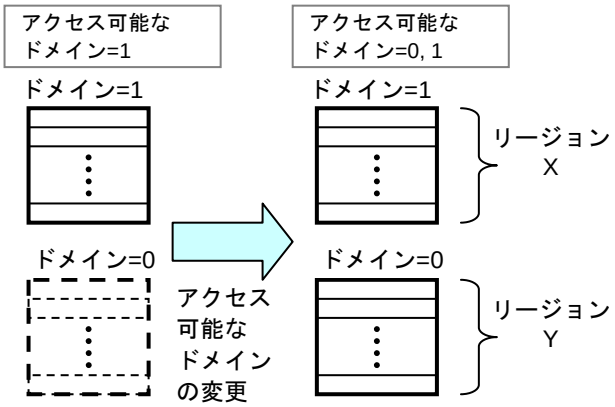

図 2　仮想アドレス空間経由方式の概要

5.2.3 アドレス空間指定方式（実装方式 3）

アドレス空間経由方式の概要を図 3 に示す。この方式では、各リージョンに一意な ASID を割り当て，一つのページテーブルに全リージョンのアドレス変換情報とアクセス保護情報を収める．メモリ保護機構はリージョン遷移時に，ASID を遷移先リージョンに応じたものに変更する

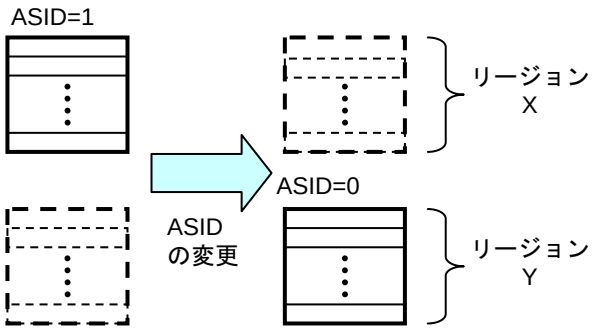

図 3　アドレス空間指定方式の概要

ことで保護を切替える．この実装方式の利用は ASID の機能を持ったプロセッサに限られる．SH3 は ASID を持つためこの実装方式を利用できる．

6．評価

本稿では，提案するメモリ保護機構を ARM9 と SH3 の各プロセッサに対して実装した．使用した環境を表 1 に示す．

表 1　実験環境

プロセッサ	ARM9(ARM922T[6])	SH3(SH7727[7])
ボード	KZ-ARMEXPCI(M4)	MS7727CP02
動作クロック	132MHz	96MHz
コンパイラ	gcc(3.3)	gcc(3.3)
コンパイル オプション	-O2	-O2

測定に用いたプログラムでは，L2 のリージョン A から L1 のリージョン B を呼び出して戻ってくるまでの時間を，メモリ保護機構によるオーバヘッドとして測定した．SH3 の実装は実装方式 1 と 3，ARM9 の実装は実装方式 1 と 2 を用い 1,000,000 回実行した平均値を求めた．また，リージョン A が L3 にあり，システムコールトラップで L1 のリージョン B を呼び出す場合のオーバヘッドも測定した．ARM9 と SH3 の各実装に対する測定結果を表 2 に示す．

表 2　測定結果

	ARM9	SH3
実装方式 1	8.46 μ s	21.52 μ s
実装方式 2	0.73 μ s	－
実装方式 3	－	2.48 μ s
(L3→L1)	1.42 μ s	7.86 μ s

ARM9，SH3 ともに，実装方式 1 が最もオーバヘッドが大きく，実装方式 2，3 が最もオーバヘッドが小さい結果となった．実装方式 1 は多くのプロセッサで用いることができる反面，リージョン遷移時に TLB をフラッシュする必要があるため，時間がかかっている． 実装方式 2，3 は各プロセッサが持つ MMU の機能を用いて高速に動作し，L3 のリージョンからシステムコールトラップを用いて L1 のリージョンを呼び出すよりも短い時間で処理している．

以上より，プロセッサに応じた実装方式を用いることで，従来の 2 レベルを用いた保護よりも高速に特権レベルのプログラムを呼び出せる L2 のレベルを設けることができた．

7. おわりに

本稿では，組込みシステムの特徴を考慮したメモリ保護機構を提案し，複数の実装方式を示した．また，ARM9 と SH3 に対して各実装方式を用いた実装を行い，実行時間のオーバヘッドを評価した．その結果，従来の 2 レベルを用いた保護よりも高速に特権レベルのプログラムを呼び出せるレベルを設けることができた．今後はリアルタイム OS にメモリ保護機構を組み込み，その有用性を検証する．

参考文献

[1] A. Silbeschatz, P. Galvin, and G. Gagne. Operation System Concepts, John Wiley & Sons, Inc., 6th edition, 2002.

[2] 品川高廣，河野健二，高橋雅彦，益田隆司．拡張コンポーネントのためのカーネルによる細粒度軽量保護ドメインの実現，情報処理学会，第 40 巻，第 6 号，pp.2596-2606, Jun 1999.

[3] M. M. Swift, B. N. Bershad, and H. M. Levy. Improving the Reliability of Commodity Operating Systems, Proc. 19th ACM Symposium on Operating Systems Principles, pp.207-222, Oct. 2003.

[4] TRON 協会．μITRON4.0 仕様 保護機能拡張 Ver.1.00.00, 2002.

[5] TRON 協会．IIMP プロジェクト, http://www.assoc.tron.org/iimp/.

[6] ARM. ARM922T (Rev.0) Technical Reference Manual, 2001.

[7] ルネサステクノロジ．DSP SH7727 ハードウェアマニュアル Rev.5.00， 2005.