Vol.2009-SE-164 No.30
Vol.2009-EMB-13 No.30
Vol.2009-CSEC-45 No.30
2009/5/29

動的変更機能を有するメモリ保護機能の評価

山田晋平[†1]　中本幸一[†1,†2]

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．想定外の動作による不正なアクセスを防ぐための一つの手段として，メモリを保護する機構がある．本稿では，複数のプログラム領域間におけるメモリの読込み・書込み，機能の実行について，許可されたアクセスのみを可能とするメモリ保護機構を提案する．これにより，システムに応じた細かな保護を実現し，組込みシステムのさまざまな保護のパターンに対応する．また，ARM9 における実装を示し，評価を行う．

Evaluation of Memory Protection Mechanism with Dynamic Changes

Shimpei Yamada[†1] and Yukikazu Nakamoto [†1,†2]

Recently, in embedded systems, programs become complex with increasing size of system. As a result, unexpected behavior of a program has an impact on the other program areas, and the reliability of the system decrease. To prevent illegal access to other program areas, memory protection mechanism is necessary for embedded systems. In addition, resources tend to be dispersed into OS, middlewares and applications in embedded systems. Middlewares, and applications manage resources directly, so that it is necessary to protect multiple areas. This paper proposes a memory protection mechanism that enable allowed accesses between multiple program areas. Additionally, we evaluate the overhead when proposed memory protection mechanism is used on ARM9 processor.

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．また，プロセッサの低電力化，劣悪な稼動環境での利用などによりメモリの故障が発生し，プログラムの動作に悪影響を及ぼしている（例えば，文献 1)）．このような，想定外の動作による不正なアクセスを防ぐための一つの手段として，メモリを保護する機構がある．

PC やサーバといった汎用システムは，プロセッサの MMU(Memory Management Unit) によるメモリ保護の機構を備えている[2)]．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．この資源には，メモリ領域や管理データにマップされたデバイス制御レジスタが含まれる．また，組込みシステムでは，ミドルウェアやアプリケーションが資源を直接管理することが多いため，複数領域に対するメモリ保護が必要となる．加えて，組込みシステムは，各分野に特化した専用システムとして作られるため，プログラム領域を保護するパターンはさまざまであり，システムごとにメモリ保護機構を用意しなければならない．

本稿では，組込みシステムにおける，さまざまな保護のパターンに対応可能な保護機構を提案・評価する．提案するメモリ保護機構を用いることで，各プログラム領域同士の間における互いのメモリの読書き，機能の実行を個別に許可または禁止できる．アクセスの許可・禁止の設定に応じて，プログラム実行時にメモリ保護を変更することによって，必要な領域のみへの限定したアクセスを実現し，不正アクセスやバグによる誤動作の影響を防ぐ．

筆者らはすでに組み込みシステム向けメモリ保護機構の用件を発表している[3)]．本稿ではこれに基づき，ARM9 における実装と評価を述べる．

本稿の構成は以下のとおりである．2 章で関連研究について述べる．3 章では組込みシス

†1 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
†2 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

Vol.2009-SE-164 No.30
Vol.2009-EMB-13 No.30
Vol.2009-CSEC-45 No.30
2009/5/29

テムで求められるメモリ保護について，4 章では提案するメモリ保護機構の仕様について述べる．5 章では ARM9 における実装を示し．6 章では実装に対する評価を行う．最後に 7 章で結論を示す．

2. 関 連 研 究

Nook では，Linux のデバイスドライバのバグにより，他の領域への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[4]．また，動的にメモリ保護情報を変更するものとして，文献 5) がある．ここでは，ユーザプロセス空間に書き込まれた悪意のあるプログラムを Linux カーネルから実行できないようにするために，システムコール実行時に仮想マシンモニタでユーザプロセスの保護情報を変更している．

組込みシステムにおいては，メモリ保護機能を提供する ITRON4.0/PX 仕様[6] と，その実装である IIMP カーネル[7] 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空間，特権レベル空間の両方に置くことができる．

しかし，これらの関連研究では，必要最低限のメモリのみをアクセス可能とする機能，さらにこれを特権レベルでも実現したものはない．

3. 組込みシステムで求められるメモリ保護

組込みシステムにおいて求められるメモリ保護について説明する．

3.1 複数領域のメモリ保護

1 章で述べたとおり，組込みシステムでは，ミドルウェアやアプリケーションが資源を直接管理することが多いため，複数領域に対するメモリ保護が必要となる．ここで，複数領域のメモリ保護の例を説明する．**図 1** は通信ミドルウェアと，通信ミドルウェアに関係しない他のプログラムの関係を示している．この通信ミドルウェアは，通信のセッションを制御する通信制御プログラムと，実際にデータを転送するデータ転送プログラムに分割されており，互いの機能の呼び出しながら協調動作する．メモリのアクセスについては，データ転送プログラムが，通信制御プログラムの持つ通信設定情報を読み込み，それ以外の読書きは禁止する必要がある．

通信ミドルウェアと直接関係しない他のプログラムとの間では，互いのメモリの読書きや機能の呼出しを禁止する必要がある．一方，通信ミドルウェアのプログラム同士が協調動作

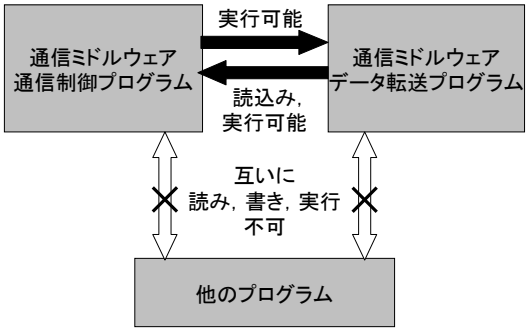

図 1 複数領域のメモリ保護

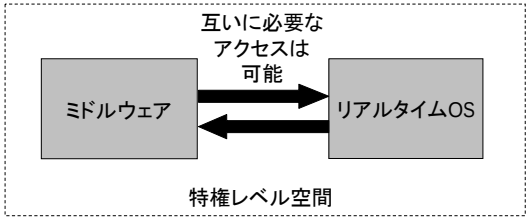

図 2 特権レベル空間における保護

するために，図 1 の場合では，互いに機能の呼出しが可能，データ転送プログラムから通信制御プログラムのメモリの読込みが可能であるとし，それ以外はアクセス不可とするといったメモリ保護が考えられる．このようにシステム内で必要となるアクセスのみを可能とすることで，プログラムの想定外の動作が起こった場合に，他のプログラム領域にその影響を及ばないようにできる．

3.2 特権レベル空間における保護

図 2 のように，ミドルウェアとリアルタイム OS が同じ特権レベル空間にある場合，ミドルウェアがリアルタイム OS を破壊してしまう場合がある．このような特権レベル空間の場合にも，互いに必要なアクセスのみを可能とすることで，他のプログラム領域への影響を防ぐ．また，保護機構を特権レベル空間に配置することで，ソフトウェアトラップよりも処理の軽い関数呼出しにより保護を変更できる．

Vol.2009-SE-164 No.30
Vol.2009-EMB-13 No.30
Vol.2009-CSEC-45 No.30
2009/5/29

表 1 リージョンの構成要素

構成要素名	説明
リージョン名	リージョンを識別するための名前の指定
プログラム	リージョンに含まれるプログラムの指定
リージョンの種類	リージョンが特権リージョンか非特権リージョンかの指定
メモリ配置	プログラムの配置が RAM か ROM かの指定， アドレスの指定
ページサイズ	プログラムを格納するメモリのページサイズの指定
外部関数	リージョン外から呼ばれる関数の引数，型の並び

4. メモリ保護機構の仕様

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ることである．なお，本機能の仕様策定にあたっては，μITRON[8] や OSEK/VDX[9] のようなリアルタイム OS に適用できるように，OS に依存しないものとした．

4.1 リージョン

提案するメモリ保護機構では，メモリ保護の単位をリージョンと呼ぶ．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のメモリ上における配置に必要な情報を合わせたものである．リージョンには特権リージョンと非特権リージョンが存在する．特権リージョンはプロセッサの特権モード・非特権モードに関わらず実行でき，非特権リージョンは非特権モードでのみ実行できるリージョンである．リージョンの構成要素を**表 1** に示す． 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．

4.2 リージョン間の関係

提案するメモリ保護機構では，リージョン間でどのようなアクセスが可能であるかを示す accept という関係を定義する．accept 関係は，あるリージョンから別のリージョンへ，読込み，書込み，実行のそれぞれについてアクセス可能であることを示す．読込み，書込み，実行のすべてのアクセスが禁止される場合，accept 関係は存在しない．

3 章で述べた複数領域のメモリ保護は accept 関係を用いて表現できる．図 1 の複数領域のメモリ保護の例は，**図 3** のような accept 関係になる．図 3 中の accept に続く 3 文字は，

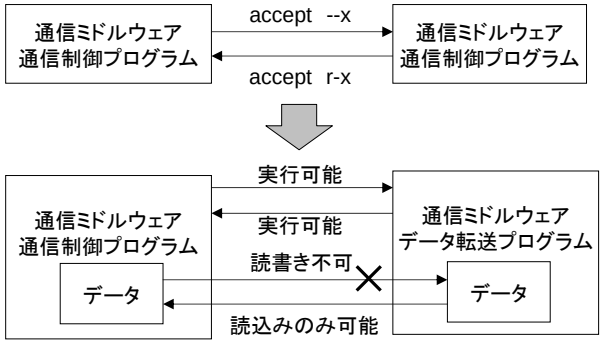

図 3 accept 関係の例

1 文字目から順に，読込み，書込み，実行のアクセスが可能であるかどうかを示す．r，w，x はアクセス可能を，-（ハイフン）はアクセス禁止を表す．通信ミドルウェアの各プログラムとそのほかのプログラムは，互いにアクセスが禁止されるため，accept 関係は存在しない．通信ミドルウェアの各プログラム同士は，図 3 の上側に示す accept 関係が存在し，メモリの読書きと機能の実行の保護で表すと図 3 の下側のようになる．

4.3 メモリ保護機能の設計

提案するメモリ保護機構は accept 関係を実現するため，保護変更の API である `changeProt` を提供する．`changeProt` は現在のリージョンが他のリージョンの関数を呼び出し，リージョン遷移する際に呼ばれる．`changeProt` の仕様を**図 4** に示す．

accept 関係は `changeProt` を利用して以下のように実現される．リージョン A から B へ実行可能な accept 関係がある場合，**図 5** に示す呼出し関係となる．保護機構が存在しない場合は，A の `func1` が B の `func2` を直接呼び出す．保護機構が存在する場合は，リージョン遷移の際に `changeProt` を呼び出し，保護を変更する必要がある．`func1` から `changeProt` を直接呼び出すためには，`func1` 中の `func2` を呼び出す部分を変更しなければならない．`func1` を変更せずに `changeProt` を呼び出せるようにするために，`func2` と同じ名前，同じ引数を持つラッパ関数を用いる．ラッパ関数は A から呼ばれ，`func2` のアドレスとリージョン B のリージョン ID，A から渡された引数を用いて `changeProt` を呼び出す．ラッパ関数は，`func2` の名前と引数を用いて自動生成される．

ラッパ関数より呼び出された `changeProt` は，保護の変更，目的の関数の呼出しを行う．

Vol.2009-SE-164 No.30
Vol.2009-EMB-13 No.30
Vol.2009-CSEC-45 No.30
2009/5/29

```
void changeProt
(region_t reg_id, (void *fn)(), char *arg_p, int arg_size);
```

・引数

reg_id	:	遷移先リージョンを示すリージョン ID（各リージョンを表す一意な値）
fn	:	目的の関数へのポインタ
arg_p	:	目的の関数に渡す引数へのポインタ
arg_size	:	目的の関数に渡す引数のサイズ（バイト数）

・機能

保護を reg_id で指定された遷移先リージョンに応じたものに変更し，
fn が指す関数に，arg_p，arg_size で表される引数を渡して呼び出す．

図 4 changeProt の仕様

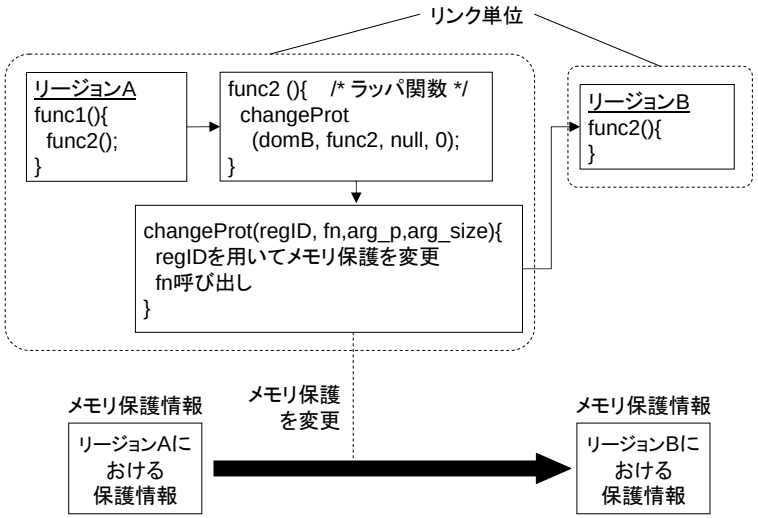

図 5 changeProt を用いたリージョン遷移

呼び出した関数が終了すると処理が changeProt に戻るので，変更した保護を元に戻してから呼び元の関数に戻る．

以下のように，前述したリージョンは開発ツールを使って構成される．リージョンは，外部からアクセス可能な外部関数をリージョン毎に宣言したリージョンファイルと，保護情報をエディタで作成し，リージョン情報を定義したファイルを作成する．作成したファイルを基に，changProt を介して外部関数を呼び出すラッパ関数を生成する．リージョン毎に当該リージョン，他リージョンのデータとラッパ関数をリンクし，仮想アドレス空間に順次配置する．この時点ではラッパ関数における外部関数のアドレスは未定義である．外部関数のアドレスを抽出して，ラッパ関数に埋め込み，リージョン情報中の保護情報を抽出しページテーブルを生成する．最後に，生成したページテーブルをメモリ保護機構に渡す．

5. 保護の変更の実装方式と ARM9 における実装

本章では，提案するメモリ保護機構の実装に用いる MMU，保護の変更を実装する方式，ARM9 における実装について述べる．

5.1 MMU

提案メモリ保護機構では，メモリの読書き・機能の実行の保護を，プロセッサが持つ MMU を用いて実現する．

5.1.1 MMU の種類

MMU を持つプロセッサには，アドレス変換のためのバッファである 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．

5.1.2 ARM9 の MMU

ARM9 の MMU について説明する．ARM9 では TLB の変更をハードウェアが行う．また，ARM9 の MMU では複数のアドレス変換方式が用意されており，使用する構成をユーザが設定することができる．本節では実装に用いた構成を**図 6** に示す．変換には 1 レベル目の変換テーブルから 2 レベル目のページテーブルを経由して物理アドレス空間にアクセスする，2 レベルの変換方式を用いた．2 レベル目には，コアースページテーブルと呼ばれる 256 エントリのページテーブルを用いた．アドレス変換の単位には 16KB の大ページを用いた．

仮想アドレスは 32 ビットで表される．1 レベル目の変換テーブルの先頭アドレスは変換テーブルベースレジスタ（Translation Table Base Register, TTBR）で表され，仮想アドレスのビット 20 から 31 をオフセットとして参照される．参照されるエントリには，2 レベル目のコアースページテーブルのベースアドレスと，ドメインが含まれる．ドメインに

Vol.2009-SE-164 No.30
Vol.2009-EMB-13 No.30
Vol.2009-CSEC-45 No.30
2009/5/29

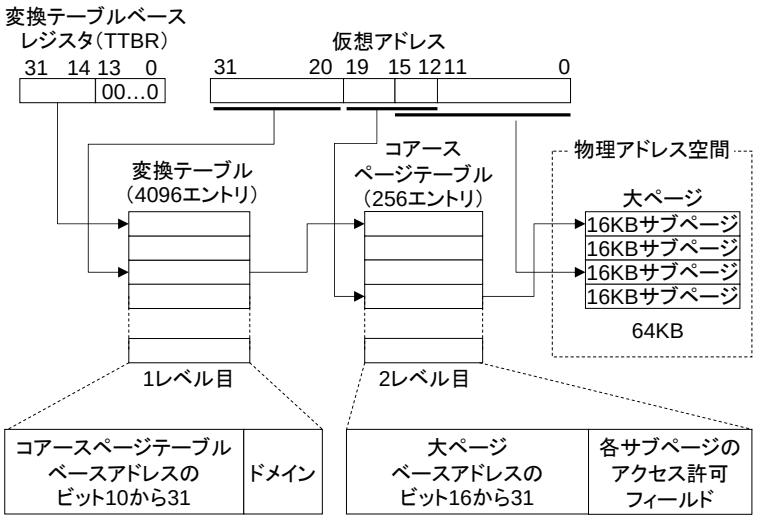

図 6 ARM9 の MMU

ついては後で説明する．コアースページテーブルは仮想アドレスのビット 12 から 19 をオフセットとして参照される．参照されるエントリには，主に大ページのベースアドレスと大ページに含まれる 4 つの各サブページに対するアクセス許可フィールドが含まれる．大ページは仮想アドレスのビット 0 から 15 をオフセットとして参照される．

サブページに対するアクセス許可フィールド（AP）は，システムの持つシステム保護ビット（S）と ROM 保護ビット（R）との組合わせにより**表 2** に示す 6 種類の状態を表すことができる．

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 個存在し，各コアースページテーブル内のページは，どれか一つのドメインに所属させることができる．各ドメインは以下の保護情報を持つ．

- アクセス不可
  ページへの全てのアクセスを禁止する
- アクセスチェック
  ページテーブルエントリ内のアクセス許可情報を用いてページへのアクセスをチェックする

表 2 アクセス許可の状態

AP	S	R	特権モード	非特権モード
00	0	0	アクセス不可	アクセス不可
00	1	0	読込み可	アクセス不可
01	0	1	読込み可	読込み可
10	x	x	読書き可	アクセス不可
11	x	x	読書き可	読込み可
xx	1	1	読書き可	読書き可

x は don't care

- アクセス可
  ページへのアクセス時にアクセス許可情報を用いたチェックを行わない

ドメインの保護情報は，ドメイン制御アクセスレジスタに格納されている．ページへのアクセス時には，そのページの所属するドメインの保護情報がチェックされ，ページのアクセス許可情報に基づいたメモリの保護が行われるかどうかが決まる．

5.2 保護の変更の実装方式

提案するメモリ保護機構における，保護の変更を実装する方式として，以下の 3 種類が考えられる．

5.2.1 ページテーブル切替え方式

ページテーブル切替え方式の概要を**図 7** 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替える，あるいはページテーブルのエントリを変更する．このときアドレス変換情報とアクセス保護情報が変更され，保護を変化前の TLB の内容に意味がなくなるため，TLB をフラッシュする必要がある．

5.2.2 仮想アドレス空間経由方式

仮想アドレス空間経由方式はの概要を図 7 に示す． この方式では，全リージョンに対応するアドレス変換情報とアクセス保護情報を，一つのページテーブルに収める．これにより，一つの物理アドレスに複数の仮想アドレスが対応することになる．各リージョンは，自身に対応する範囲の仮想アドレスにアクセスする．提案する保護機構では，ラッパ関数により保護を変更し，リージョンの遷移を行うが，この方式ではラッパ関数を介さずに任意のリージョンにアクセスすることができるため，実行中のリージョン以外の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．

Vol.2009-SE-164 No.30
Vol.2009-EMB-13 No.30
Vol.2009-CSEC-45 No.30
2009/5/29

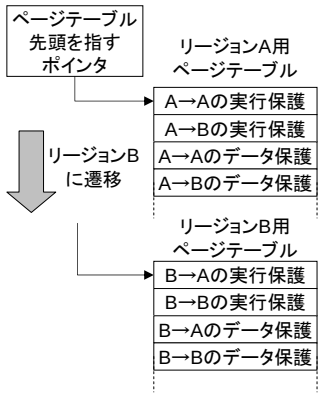

図 7 ページテーブル切替え方式

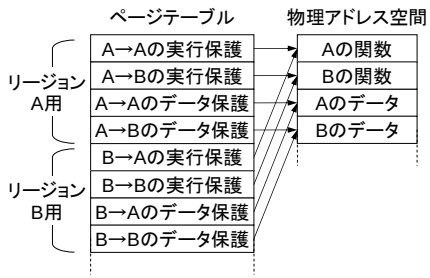

図 8 仮想アドレス空間経由方式

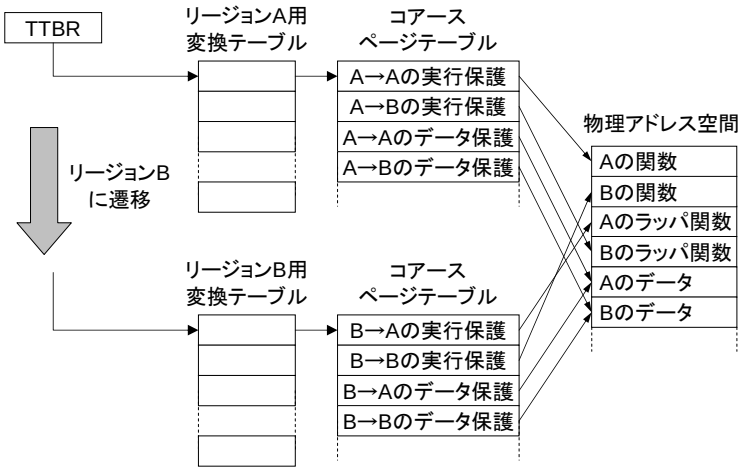

図 9 ARM9 におけるページテーブル切替え方式

5.2.3 アドレス空間指定方式

アドレス空間指定方式では，まず TLB に全リージョンのアドレス変換情報と保護情報を書き込んでおく．そして多重仮想空間における複数プロセスを区別するためのアドレス空間識別子を，リージョンに応じて切替えることで，参照される TLB エントリを変更し，保護の変更を行う．この方式はアドレス空間識別子を持ち，TLB をソフトウェアで変更できるプロセッサで使用可能な方式である．

5.3 ARM9 における実装

提案メモリ保護機構ではリージョンを遷移させるために，`changeProt` を呼び出し，遷移先リージョンに応じて保護を変更しなければならない．ARM9 にはアドレス空間識別子が存在しないため，保護の変更の実装はページテーブル切替え方式と仮想アドレス空間経由方式について考える．

5.3.1 ARM9 におけるページテーブル切替え方式

ARM9 におけるページテーブル切替え方式の実装の概要を**図 9** に示す．この例ではリージョン A と B が存在し，リージョン A から B に遷移する場合について示している．ARM9 におけるページテーブル切替え方式では，各リージョンに対して変換テーブルを一つ用意し，コアースページテーブルには各物理ページに対するアクセス許可情報を格納しておく．

リージョン A を実行している場合，TTBR はリージョン A 用の変換テーブルを指している．現在のリージョン A からリージョン B の関数を呼び出す場合，まず，リージョン B に対応するラッパ関数を呼び出す．ラッパ関数は `changeProt` を呼び出し，保護の変更を行わせる．`changeProt` は TTBR に，遷移先のリージョン B 用の変換テーブルを指させる．TTBR の変更によりページテーブルが切り替わり，現在の TLB の内容が意味を持たなくなるため，TLB をフラッシュする．これらの処理により，MMU がリージョン B 用の変換テーブルを参照するようになるので，目的のリージョン B の関数を呼び出す．目的の関数の処理が終了すると，呼出しとは逆の手順で処理を行う．`changeProt` は TTBR に，遷移前のリージョン A 用の変換テーブルを指させ，保護を元に戻す．

5.3.2 ARM9 における仮想アドレス空間経由方式

ARM9 における仮想アドレス空間経由方式の実装では，5.1 節で示したドメインを利用する．まずリージョンとドメインを一対一で対応付ける．そして，ドメインアクセス制御レジスタは，現在のリージョンに対応するドメインのみアクセスチェックにし，その他のドメインはアクセス不可にする．このようにすることで，現在のリージョンのみアクセス許可情報に基づいたアクセスを行うことができる．

図 10 に，ドメイン 0 に属するリージョン A とドメイン 1 に属するリージョン B が存在し，リージョン A からリージョン B に遷移する場合を示す．アクセス不可のリージョンは変換テーブルにおいて灰色で表現されている．

Vol.2009-SE-164 No.30
Vol.2009-EMB-13 No.30
Vol.2009-CSEC-45 No.30
2009/5/29

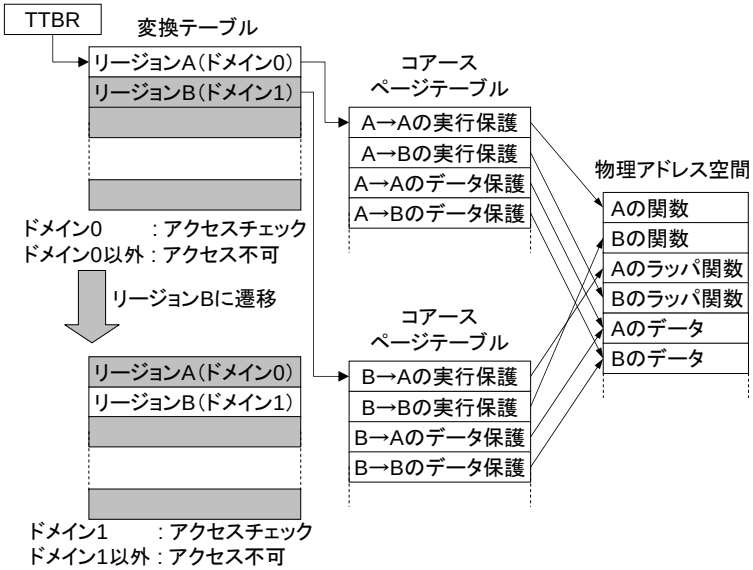

図 10　ARM9 における仮想アドレス空間経由方式

リージョン A を実行する場合，対応するドメイン 0 のみアクセスチェックさせ，それ以外のドメインはアクセス不可とする．現在のリージョン A からリージョン B の関数を呼び出す場合，まず，対応するラッパ関数を呼び出す．ラッパ関数は changeProt を呼び出し，保護の変更を行わせる．changeProt はドメインの保護情報を変更する．遷移先のリージョン B に対応するドメイン 1 のみアクセスチェックにし，それ以外のドメインはアクセス不可とする．この処理により，MMU がリージョン B 用の変換テーブルを参照するようになるので，目的のリージョン B の関数を呼び出す．目的の関数の処理が終了すると，呼出しとは逆の手順で処理を行う．changeProt は，遷移前のリージョン A に対応したドメイン 0 のみアクセスチェックにし，それ以外のドメインをアクセス不可にすることで，保護を元に戻す．

6. 評　　価

提案メモリ保護機構では，リージョンを遷移する度に保護が変更される．提案メモリ保護機構を適用した場合の，保護の変更にかかる実行命令数を調べるため，**表 3** に示す環境で動作する，提案メモリ保護機構の評価用プログラムを作成した． 評価用プログラムではリージョン A と B が存在し，リージョン A から B の関数は実行可能，それ以外のアクセスは禁止とした．リージョン A には関数 func1 を持つプログラムが，リージョン B には関数 func2 を持つプログラムが属している．func1 は func2 を呼び出す関数，func2 は引数を持たず何もしない関数である．メモリ保護機構を用いた呼出しの関係は図 5 と同じになる．これら関数はそれぞれ 1 ページに収まるサイズである．

表 3　実験環境

プロセッサ	ARM9(ARM922T[10])
ボード	KZ-ARMEXPCI(M4)
動作クロック	132MHz
コンパイラ	gcc(3.3)
コンパイルオプション	-O2

表 4　実行命令数の比較

保護の変更方式	呼出し時	戻り時
ページテーブル切替え方式	26	9
仮想アドレス空間経由方式	25	8

6.1 実行命令数の比較

評価用プログラムにおける，func1 から func2 の呼出しにかかる命令数と，func2 から func1 に戻るのにかかる命令数を，それぞれのメモリ保護変更方式に対して調べた．その結果を**表 4** に示す． 保護の変更方式は 5.3 節で述べた 2 種類である．ページテーブル切替え方式における保護の変更は，リージョンに対応する変換テーブルの先頭アドレスを TTBR に書込み，TLB をフラッシュすることで行う．仮想アドレス空間経由方式ではドメインアクセス制御レジスタに，必要なリージョン以外をアクセス不可とする値を書込む．TTBR への書込みもドメインアクセス制御レジスタへの書込みも同じ命令数で行えるため，TLB フラッシュをするための命令数のみが両者の差となって現れている．そのため，両者の間に命令数の差があまり見られない結果となった．呼出し時と戻り時のそれぞれで差が現れているのは，目的の関数の呼出しの前後で保護の変更を行う必要があるためである．

6.2 制約条件の比較

5.3 節の各変更方式をシステムに適用するために，考慮すべき条件が存在する．

ページテーブル切替え方式では，一つのリージョンに一つの変換テーブルを対応付ける

Vol.2009-SE-164 No.30
Vol.2009-EMB-13 No.30
Vol.2009-CSEC-45 No.30
2009/5/29

ため，リージョンの数だけ変換テーブルが必要となる．ARM9 では変換テーブル一つが 32 ビットのエントリを 4096 個持つため，リージョン数を R としたとき，変換テーブルには $4 \times 4096 \times R$ バイトのメモリが必要になる．

仮想アドレス空間経由方式では，一つのリージョンに変換テーブルのエントリを対応付けるため，変換テーブル自体は一つあれば良い．そのため ARM9 では，変換テーブルに必要なメモリは 4×4096 バイトでよい．ただし，任意のリージョンへ直接アクセスすることを防ぐため，一つのリージョンに一つのドメインを対応づける必要がある．よって，リージョンの数がドメインの最大数以下となる必要がある．ARM9 ではドメインの最大数が 16 なので，リージョンの数は 16 以下に限定される．

必要なメモリ量は仮想アドレス空間経由方式の方が少ないが，ドメインの最大数の制約を受けるため，十分なメモリがある場合にはページテーブル切替え方式も有効である．

6.3 適用事例

特権/特権間の保護機能は一方の特権リージョンにリアルタイムタスク，もう一方の特権リージョンにカーネルを置き，リアルタイムタスクのカーネル呼出しのオーバヘッドを小さくする一方，リアルタイムタスクのバグやメモリの一時故障による異常動作からカーネルを保護することができる．

非特権/特権間保護機能は，非特権保護リージョンにダウンロードプログラムような信頼性の低いプログラムを置き，特権空間の保護リージョンにシステム全体に関与する資源をおくことで，非特権保護リージョンのプログラムから特権保護リージョンの資源を保護する．この場合も，特権リージョンのプログラムから非特権リージョンのプログラムを保護することが可能である．

7. おわりに

本論文では，組込みシステムにおいて，プログラムの想定外の動作により他のプログラム領域に影響が及ぶことを防ぐため，組込みシステムにおけるさまざまな保護のパターンに対応可能なメモリ保護機構を提案した．提案メモリ保護機構は各プログラム領域の間におけるメモリの読書き，機能の実行について，許可されたアクセスのみを可能とする．この機構を用いて，システム内で必要となるアクセスのみを可能とすることで，プログラムの想定外の動作が起こった場合に，他のプログラム領域にその影響が及ばないようにした．

また，一つの関数が別の関数を呼び出す単純なプログラムに対して提案メモリ保護機構を適用し，関数呼出しにおいて保護が変更されるときの実行命令数を調べた．評価では，ARM9 における 2 種類の保護の変更方式の実装について差を比較した．その結果，実行命令数の面では，ページテーブル切替え方式で必要な TLB フラッシュ用の命令数が差として現れた．

今後は，リアルタイム OS を用いたシステムに対して，提案メモリ保護機構を適用した場合における評価を行う．

参考文献

1) Chandra, V. and Aitken, R.: Impact of Technology and Voltage Scaling on the Soft Error Susceptibility in Nanoscale CMOS, *Proc. the 2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, pp.114–122 (2008).
2) Silbeschatz, A., Galvin, P. and Gagne, G.: *Operating System Concepts*, John Wiley & Sons, Inc., 6th edition (2002).
3) Yamada, S., Nakamoto, Y., Azumi, T., Oyama, H. and Takada, H.: Generic Memory Protection Mechanism for Embedded System and Its Application to Embedded Component Systems, *Proc. IEEE 8th International Conference on Computer and Information Technology*, pp.557–562 (2008).
4) Swift, M.M., Bershad, B.N. and Levy, H.M.: Improving the reliability of commodity operating systems, *Proc. the 19th ACM Symposium on Operating Systems Principles*, pp.207–222 (2003).
5) Seshadri, A., Luk, M., Qu, N. and Perrig, A.: SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes, *Proc. 21st ACM SIGOPS symposium on Operating systems principles*, pp.335–350 (2007).
6) TRON 協会：μITRON4.0 仕様 保護機能拡張 Ver.1.00.00 (2002).
7) TRON 協会：IIMP プロジェクト．〈http://www.assoc.tron.org/iimp/〉（参照 2009-04-21）．
8) Takada, H. and Sakamura, K.: *μ ITRON for small-scale embedded systems*, Vol.15, No.6 (1995).
9) OSEK/VDX: *Operating System Version 2.3.3* (2005).
10) ARM: *ARM922T (Rev 0) Technical Reference Manual* (2001).