2A-05

組込み機器向け複数 OS 同時実行におけるメモリ保護の一検討

髙橋　由梨香†　東山　知彦†　出原　章雄†　落合　真一†

三菱電機株式会社　情報技術総合研究所†

1. はじめに

近年，組込み H/W の高性能化に伴い単一 H/W プラットフォームで，汎用 OS とリアルタイム OS 等，複数 OS を同時実行可能となりつつある．複数 OS 同時実行の場合，複数 OS を管理する S/W 機構が必要となる．本稿ではこれをハイパーバイザと呼ぶ．この場合，一般的に H/W のサポートによる仮想化支援機能(Hardware Assisted Virtualization，以下 HAV)を用いる．一方，組込み向け機器は HAV を持たない H/W 使用も多く，我々はこうしたターゲットに向けた組込み向けハイパーバイザを提案している[1].

本提案では，各 OS が H/W 構成情報を保持し，OS 間のメモリ保護を行う．この場合 H/W 構成変更により，都度 OS 修正が必要となる．そこで HAV を持たない H/W 上で，各 OS が H/W 構成情報を保持することなく，複数 OS 間のメモリ保護を実現する方法の検討を行った．

2. 課題

我々の組込みハイパーバイザは，マルチコアを前提とし，各コアに対し異なる OS を割り当て，OS が保持する H/W 構成情報に基づきコアごとにメモリ管理ユニットを設定する．これにより，OS のメモリ領域を分離実行する．

一方で，製品仕様変更等により H/W 構成が変更されることがある．我々の従来提案手法では，H/W 構成変更への対応の為に都度 OS の修正が必要となり，工数の増大が発生する．H/W 構成変更のイメージを図 1に示す．本図変更前では OS0 に割り当てられた I/O を I/O-A，I/O-B，OS1 に割り当てられた I/O を I/O-C，I/O-D としている．この例では製品仕様変更等により H/W 構成変更が発生した場合，I/O-B は OS1 に割り当てられ，I/O-D はなくなったことを表している．

このように H/W 構成の変更が発生した際にも，毎回追加で OS の修正を行うことなく，容易に対応可能とすることが課題となる．本論文では，この課題を解決に向けたメモリ保護方式を検討する．

A study on memory protection method for multiple OS platform in embedded systems

Yurika Takahashi, Tomohiko Higashiyama, Akio Idehara, Shinichi Ochiai

† Information Technology R&D Center, Mitsubishi Electric Corporation

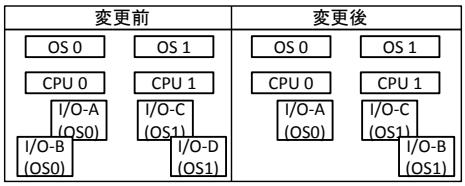

図1　H/W 構成変更のイメージ

3. 設計

H/W 構成の変更による割り当てメモリ領域の変化のイメージを図 2に示す．

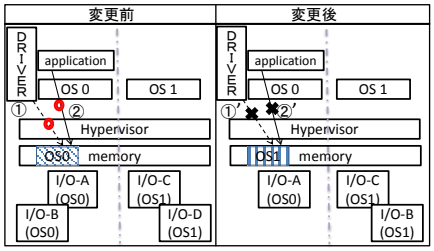

図2　割り当てメモリ領域の変化

H/W 構成は図 1と同様の変更内容とする．図 2では，H/W 構成の変更により，メモリ領域の割り当てが変化していることを意味する．図 2のメモリ中の領域は I/O-B に割り当てられている領域とする．図 2の①，①'はドライバから，②，②'はアプリケーション(以降アプリ)からのアクセスを意味する．変更前は I/O-B は OS0 に割り当てられていたため①，②はメモリアクセス可能であったが，構成変更後は OS1 に割り当てられているため，ハイパーバイザにより①'，②'はメモリアクセス不可とする必要がある．そこで今回は，ドライバおよびアプリからメモリアクセスに対し，割り当てられているメモリ領域内のアクセスのみを許可するための設計を行うこととした．以降，これらに対する設計内容を示す．

(a) ハイパーバイザが H/W 構成情報のテーブルを生成

(b) 割り当てられた領域に対してのみメモリアクセス処理を行うように OS の処理の修正

3.1. ハイパーバイザによるテーブルの作成

H/W 構成変更に対する柔軟な対応の実現の為に OS 自体に H/W 構成情報を所有させず，ハイパーバイザが H/W 構成情報に関するテーブルを作成し，OS がそのテーブルを参照することで H/W 構成データを抽象化する．

ハイパーバイザの作成するテーブルの内容は，ターゲットの仕様に基づき，割り当てられた I/O

領域の開始のアドレス，領域のサイズ，アクセス権とした．ここでいうアクセス権とは，メモリアクセスが発生した場合，割り当てられたメモリ空間に対し，ユーザ空間からのアクセスを認めるか，カーネル空間からのアクセスのみを認めるか，といったマップ要求元の所有する権利の有無を示す．このデータはマップ要求元により，処理を変更する必要がある際に用いる．

3.2. OS 内のメモリアクセス処理修正への変更

各 OS に修正を加え，OS 起動時に，ハイパーバイザが作成したテーブルを参照し，自身に割り当てられたメモリ領域のみをアクセス可能とするように変更を行う．また，OS 起動時以外に，ドライバやアプリからメモリマップ登録要求が発行される可能性がある．そこで，OS 起動後のドライバおよびアプリからのメモリ空間へのアクセスに対しアクセス制限を行う．

4. 実装

今回使用した開発環境を表 1に示す.

表 1　開発環境

Target	Cyclone V 評価ボード(Altera 社)
CPU	ARM Cortex-A9(2Core,800MHz)
Memory	DDR SDRAM(133 MHz),1GB
S/W version	Kernel: Linux 3.9.0 ベース
	glibc: GNU C Library version 2.15
Cross Toolchain	toolchain: linaro-arm.2012.11, (gcc:version 4.7.3)

4.1. ハイパーバイザによるテーブルの作成

ブートローダの起動時にハイパーバイザをロードする．ハイパーバイザがハイパーバイザ管理領域を用意し各 OS の H/W 構成情報のテーブルを作成する．

4.2. OS 内のメモリアクセス処理修正への変更,

OS 起動時にテーブルのデータを参照し，割り当てられた領域内のみアクセスするようにした．また OS 起動後のメモリマップ登録要求に対しては，作成したテーブルのデータに基づくマップ制限を行うように OS を修正した．修正後は，メモリマップ登録要求が発行された場合以下の処理を行う．修正後の動作を図 3に示す.

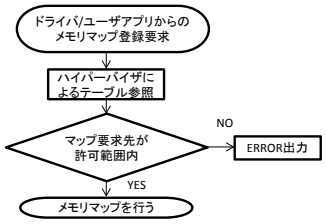

図 3　メモリアクセス制御イメージ

(1) ドライバまたはアプリからのメモリマップ登録要求が発行されたらテーブルを参照
(2) メモリマップ要求先アドレスの範囲とアクセス権は条件を満たしているかを評価
(3) 評価結果により異なる処理を行う
　(a) 評価結果が真の場合：要求されたアドレス空間にマップ登録
　(b) 評価結果が偽の場合：エラーを返し，要求されたマップ登録は行わない

本書では Linux の修正例を説明する．以下の 2 点に対して，アクセスを許可されていない範囲へアクセスできないように修正を加え実装する．

(a) ドライバからのメモリマップ登録要求 (ioremap 関数を用いたメモリマップ要求)
(b) アプリからのメモリマップ登録要求 (/dev/mem を介したメモリマップ要求)

4.2.1. ドライバからのメモリマップ登録要求

ドライバが I/O を使用する際，該当する I/O 領域のマップ登録を要求する．マップ要求が起きた場合に，**図 3**の処理に基づき，マップ登録の可否の評価を行うように修正を行うこととした．

4.2.2. アプリからのメモリマップ登録要求

アプリからのメモリマップ要求では，read, write, mmap システムコールが使用される．read, write 実行時は，Linux が管理する RAM 領域に限定したアクセスとなり，マップ制限は不要である．一方，mmap を行う際には，マップ先のアドレスの確認は行われない為，mmap に対し修正を加え，マップ登録可否判定を行う．

5. 評価と考察

今回機能評価として，ハイパーバイザによる H/W 構成情報を抽象化したテーブルの作成と，テーブルの内容に基づくマップ制限の動作確認を行った．確認の方法は，試験用のアプリおよびドライバを用いた．評価の結果，いずれの場合も割り当てられていない領域へのメモリアクセスが生じた場合は，メモリマップを許可せず，あらかじめ定義したエラーを返すことを確認した．本評価により，H/W 構成変更を行った際にも，OS に追加の修正を行わずにメモリ保護を行うことが確認でき，本手法の有効性が示された．

6. まとめ

本検討では H/W 構成に依存しない形でメモリの保護方式の検討を行った．ハイパーバイザが H/W 構成情報に関するテーブルを作成し，H/W 構成情報を抽象化した．また，ターゲットボードに対し実装し，評価を行った．今後は，ハイパーバイザが作成したテーブルの内容に応じて必要なドライバをロードする仕組みを実装し H/W 構成を完全な抽象化を目指す。

参考文献

[1] 出原, 東山, 落合「組込み制約下での複数 OS 実行環境の設計」 情報処理学会 組込みシステムシンポジウム 2014 (ESS2014)