Vol.2016-OS-138 No.13
2016/8/8

マルチコアプロセッサ環境における組込みシステム向けVMMの研究

前田　剛志[1]　小柴　篤志[1]　佐藤　未来子[1]　並木　美太郎[1]

概要：本論文では，マルチコアプロセッサ環境における組み込みシステム向け仮想マシンモニタ (VMM) について述べる．組込みシステム上で，IT 処理とリアルタイム処理を連携させることを考えたときに，VMM を用いて，汎用 OS とリアルタイム OS を共存させるという構成が考えられる．ただし，仮想化の実現により，仮想化のオーバヘッドが組込みシステムに必要とされる実時間性を保証できない可能性が存在する．そこで本論文では各ゲスト OS に割り当てる物理資源 (CPU，メモリ，I/O) に対して，レベルを設定することにより，実時間性を必要とするゲスト OS が優先的に物理資源を利用できるように VMM で資源を管理することにより，実時間性を保証できる組込みシステム向け VMM の実現を行った．評価として，ゲスト OS 側から MVM に移行する際のオーバーヘッドについて計測を行った．
今後の課題として，未実装である資源仮想化の実装とそのうえで，RTOS と汎用 OS を動作させ，評価を行うことがあげられる．

A Study of VMM-Based Embedded System on Multi-core Platforms

Maeda Takeshi[1]　Koshiba Atsushi[1]　Satou Mikiko[1]　Namiki Mitaro[1]

1. はじめに

近年，自動車やスマートフォンに代表されるように，広範な分野でマイクロプロッサが採用され，組込みシステムは広く普及している．さらに技術の進歩により，マルチコアプロセッサ，大容量 RAM 等の採用に見られるような，ハードウェアの高度化が進んでおり，IT システムとの連携を行うというような，組込みシステムへの要求は高度化している．組込みシステムでは，組込みシステムに求められるリアルタイム要求を満たすことが出来るよう，リアルタイム処理を行うリアルタイム OS(以下，RTOS) が採用されることが多い一方，前述のように組込みシステムに対する要求は高度化，複雑化しており，リアルタイム性を実現しつつ，汎用 OS を利用したいといった要求もある．
このような要求にこたえるために，RTOS と汎用 OS を共存させることにより，リアルタイム性の求められるタスクにはリアルタイム OS を，IT システムの処理には汎用 OS を利用するというような両者の特徴を生かすことのできる実行基盤が求められている．この要求を満たすために，特に，Gandalf[1] や DARMA[2] のような，仮想化技術を用いる実行基盤の研究が存在する．しかし，組込みシステムに仮想化技術を適用することによる，仮想化処理のオーバヘッドによるリアルタイム性の阻害，予測可能性の損失という課題があり，組込みシステムの要求と仮想化技術が共存するような，実行基盤が求められている．

これらの要求を満たすため，仮想マシンのスケジューリング手法の研究 [3],[4] や，メモリ管理の研究 [5] が存在する．本稿では,CPU，メモリ，I/O の仮想化を実現する，組込みシステム向け仮想マシンモニタ MVM の設計および一部機能の実装を行った．実装に用いたターゲットボードは Zynq-7000 SoC ZC702 であり，これは組込みシステムでよく利用される ARM プロセッサを内蔵している．また，仮想化を実現するうえで，ARM TrustZone 技術を利用する．

本稿では，まず，課題を 2 章に，3 章にて目標を述べる．4 章で本システムを実現するために利用する ARM TrustZone について述べる．5 章で本研究の先行研究であ

[1] 東京農工大学
Tokyo University of Agriculture and Technology

る仮想マシンモニタ OVM について述べ, 6 章では提案する組込みシステム向け仮想マシンモニタの設計について述べ，7 章で TrustZone を用いた実装について述べ，8 章で評価を行う．最後に 9 章で本稿のまとめを述べる．

2. 課題

第 1 章で挙げたように，組込みシステムではリアルタイム性が求められる．このようなシステムへの応答要求・計算機の処理・応答処理という一連のプロセスにおいて，計算機の処理は，利用者にとっての遅延時間と表現される．リアルタイムシステムではこのプロセスの遅延時間と応答処理終了までの時間の合計が，一定時間以内である必要があり，この時間をデッドラインと呼ぶ．また，この応答処理終了がデッドラインを超過してしまうことをデッドラインミスと呼ぶ．

近年の組込み向けハードウェアの性能の向上に伴い，リアルタイム処理とともに，IT システムを並列に動作させるといった要求が組込みシステムに求められている．そこで，豊富なソフトウェアが利用でき，周辺装置を扱うドライバなどの備わっている Linux のような汎用 OS を利用することが考えられる．しかし，汎用 OS はリアルタイムタスクを処理することを考慮していないという問題がある．そこで，リアルタイム処理は RTOS に，IT システムの利用は Linux を利用することが考えられる．そのようなシステムを実現するために，VMM を利用して，複数 OS を VM として並列動作させるという構成が考えられる．ただし，VMM を利用する方式の問題として，VM ごとにリアルタイム性が異なり，汎用 OS の処理が RTOS の処理を阻害する，といったことが生じる可能性がある．この場合，RTOS と汎用 OS を共存させるメリットが無くなり，RTOS のタスクに要求されるデッドラインを超過するなどといった致命的な事態が起こりうる．また，ゲストとして利用する OS はコードサイズが大きく，完全な検証が難しく，コードサイズが大きくなるほどその信頼性は不確かなものとなる．そのため，ゲスト OS のバグが VMM および他のゲスト OS に影響を及ぼさないことは必須の要件である．

本研究では，組込みシステムへの要求の高度化に伴う要求を満たすこと，特にリアルタイム処理と IT システムの連携という要求を満たすことを念頭に置いている．そこで，IT システムとリアルタイム処理を組み合わせるような複合型組み込みシステムの実行基盤を,VMM によって汎用 OS と RTOS を共存させる形で実現する．そこで，以下のような課題が挙げられる．

(1) **リアルタイム性の異なる VM に対するリアルタイム性の実現**

(2) **ゲスト OS の誤動作から VMM を保護する必要性**

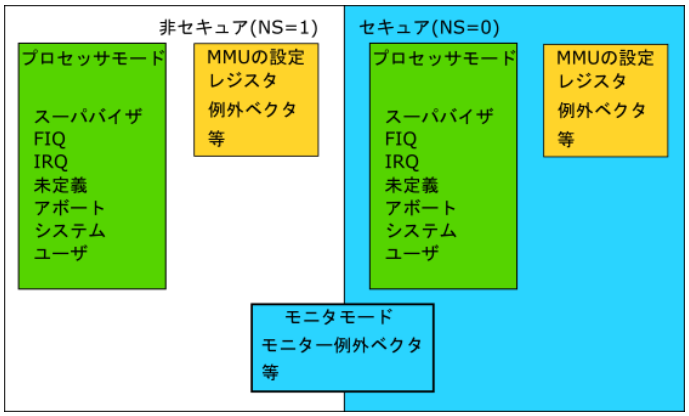

図 1 TrustZone の概要図

3. 目標

本研究では，汎用 OS と RTOS といった，リアルタイム性の異なるシステムの共存を実現するために，柔軟なハードウェア資源仮想化および管理手法を提供することで，リアルタイム性と予測可能性の損失の問題，およびシステム保護の問題を解決することを目標とする．

4. ARM の仮想化機能

4.1 アーキテクチャー概要

本研究では，評価ボードとして Zynq-7000 SoC ZC702 を採用し，システムの実装を行っている．Zynq はその特徴として ARM プロセッサとは別に FPGA を持っており，ARM プロセッサと FPGA を組み合わせて利用可能である．この評価ボードを選択した理由として，まず組込みシステムで広く採用されている ARM プロセッサである，Cortex-A9　MPCore を持っているという点である．この Zynq-7000 は Cortex-A9 を 2 つ持っており，近年の組込みシステムのマルチコアプロセッサの採用するというモデルとして適しており，また Cortex-A9 は TrustZone と呼ばれるセキュリティ拡張機能を持っており，これが組込みシステム向け仮想マシンモニタを実装する際に強力なハードウェア支援になると考えたためである．

4.2 ARM TrustZone[8]

本研究では，仮想マシンモニタを実現するために，ARM のセキュリティ拡張機能 TrusZone を利用する．TrustZone はメモリ空間をセキュアワールドメモリ空間とノーマルワールドメモリ空間に分離する．セキュアワールドにはプロセッサが Secure な時にアクセス可能で，プロセッサが NonSecure な状態の時にはアクセスができない．同様に，ハードウェア資源に対して Secure なハードウェア資源に対してはプロセッサが Secure な状態であるときにのみアクセスが可能である．また，プロセッサの Secure，NonSecure

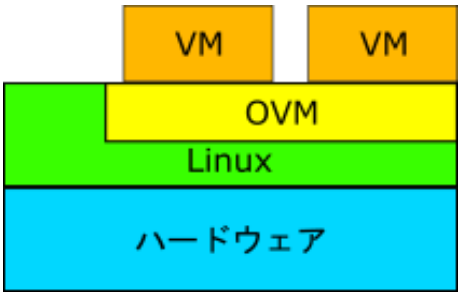

図 2　OVM の概要図

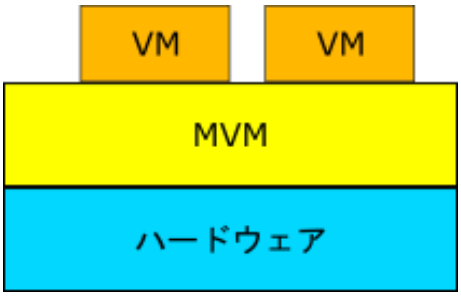

図 3　MVM の概要図

の状態の切り替えを実現するために，monitor プロセッサモードが用意されており，特定の例外やセキュアモニターコール (SMC) 命令が発行された場合にプロセッサは monitor モードとなり，ハンドラを実行し，状態の遷移を実現する．TrustZone の概要図を図 1 に示す．

TrustZone の機能では，セキュアワールドにてセキュリティが求められるアプリケーションを，ノーマルワールドにて汎用 OS のような信頼性に欠ける OS を動かすといった用途が想定されている．このため，各ワールドに独立した例外ハンドラやシステム制御レジスタ，アドレス変換テーブル等を持たせることが可能である．

このような機能を持っているため，SafeG[6] のように，TrustZone を用いて RTOS と汎用 OS をハイブリッドで動作せるというような研究もなされている．

5. 仮想マシンモニタ OVM

先行研究として，仮想マシンモニタ OVM の紹介をする．仮想マシンモニタ OVM(OptimusVirtualMachine) は準仮想化方式の組込み向けホストハイパバイザ (Type2) であり，マルチコアプロセッサ，メモリおよび I/O を仮想化する．ホスト OS として Linux を動作させ，VMM である OVM を Linux のカーネルドライバとして組み込むことにより，システムが構成される．つまり，OVM は図 2 のような構成である．

5.1 全体概要

OVM は高度な組込みシステムの実行基盤を提供するために，次の二点を満たすことを目標としている．

(1) **システムの必要とするリアルタイム性を保証すること**
組込みシステムにおけるリアルタイム処理では高いリアルタイム性が必要とされ，IT 処理ではリアルタイム性ではなく効率的なハードウェア資源の利用が必要になる．そこで，OVM ではシステム管理者やゲスト OS の判断で，静的または動的にリアルタイム性を保証する手段を変更できることを目標とする．

(2) **複数のパラダイムのシステムが並行・並列に動作できること**
想定する「高度なシステム」として，IT システムに用いられる汎用 OS と，複雑なリアルタイム処理を行うことに用いられる RTOS と，高いリアルタイム性を要求するハードウェア上で直接動くプログラム，といった複数のパラダイムのシステムから構成されるシステムを考えている．これらが並列，つまり同時に動作することや，並行，つまり短時間で切り替わりながら動作することがある．そこで，OVM ではマルチコアプロセッサに対応し，複数のパラダイムの OS を並行・並列に動作させ，システムの機能ごとに適した OS で処理を実行することを目標とする．

OVM は，ゲスト OS のリアルタイム性に応じた各ハードウェア資源の仮想化方式を提供するために，ゲスト OS に対し，二段階のリアルタイムレベルを提供する．ゲスト OS のリアルタイムレベルに応じた仮想化方式で OVM が資源を割り当てることで，異なるゲスト OS がそれぞれ要求するリアルタイム性を保証する．

5.2 リアルタイムレベル

OVM は以下に示す二段階のリアルタイムレベルをゲスト OS に設定可能にする．リアルタイムレベルによって異なる仮想化方式をゲスト OS に提供することができ，リアルタイム要求の厳しいゲストには資源の占有や優先利用，そうでないゲストには資源の効率的な利用をさせる．これにより，リアルタイム性の異なる OS の共存を実現する．

(1) HRT(ハードリアルタイムゲスト向け)
他のシステムから独立してリアルタイム性を保証する手法を用いる．このレベルは計測と制御を司る RTOS に用いることを想定している．

(2) SRT(ソフトリアルタイムゲスト向け)
優先的に物理資源を獲得するが，余剰資源は共有する．このレベルはメディア関係の処理を司る汎用 OS に用いることを想定している．

6. 仮想マシンモニタ MVM の設計

この章では，OVM の設計思想を引き継いだ，Type1 仮想マシンモニタ MVM の設計について述べる．

6.1 概要

MVM は Type1 仮想マシンモニタであり，Linux のカーネルモジュールとして組み込まれていた Type2 仮想マシンモニタである OVM とは異なる．このように設計を変更した理由として，ARM TrustZone を利用して，仮想化およびハイパバイザーの保護を実現することを考えたとき，OVM の場合，VMM の機能をセキュアワールドに置くために Linux もセキュアワールドに配置され，VMM 保護の観点から見ると疑問であった点があげられる．MVM の場合，セキュアワールドで動作するアプリケーションは VMM である MVM のみという構成になるため，この問題を解消できると考えられる．

MVM は資源の仮想化機構に加え，ゲスト OS から MVM へ遷移し，MVM の機能を利用するハイパバイザーコール機能を持っている．ゲスト OS はコンフィギュレーションファイルにより設定され，設定項目としてリアルタイムレベル，メモリ使用量を記述する．

6.2 CPU 仮想化手法

6.2.1 コアの占有機能

組込みシステムにおいて，ハードリアルタイム性を要求する分野では，時間制約が厳しく，デッドラインを超えると，システムに致命的な事態を起こす．CPU を仮想化し，ゲスト OS をコンテキストスイッチさせ，並行に動作する場合，VM が得られる実行時間は短くなり，ハードリアルタイム性を要求するタスクに対して，リアルタイム性を保証できない．そこで，OVM ではハードリアルタイム性を要求するゲスト OS に対応するため，ゲスト OS がコアを占有することで，コンテキストスイッチによるオーバヘッドを無くし，リアルタイム性を保証する．この機能は HRT レベルのゲスト OS に提供される．

6.2.2 仮想 CPU のスケジューリング機能

コア占有機能とは異なり，ハードウェアリソースを共有し，複数のゲストを平行に動作させ，リアルタイム性を必要とするゲストをに対応するためにスケジューリング機能を提供する．MVM ではゲスト OS の必要とするリアルタイム性を考慮する，リアルタイムスケジューリングを行う．

6.2.3 割り込みスケジューリング機能

複数のゲスト OS が並行に動作している環境で，ゲスト OS が CPU の実行権を持っていない場合にいつ割り込みを処理するか，という問題が生じる．割り込みにはソフトウェア割り込みとハードウェア割り込みが存在する．CPU の処理が原因となるソフトウェア割り込みは，現在物理 CPU の実行件を持っている仮想 CPU により発生していることが確定しているので，即時割り込みを行う．ハードウェアの処理が要因となるハードウェア割り込みは，割り込み先の仮想 CPU が物理 CPU の実行権を持っていない場合があるので，割り込み先の仮想 CPU が実行権を得てから割り込み処理を行う，遅延割り込み機能を持つ．

6.3 メモリ仮想化手法

6.3.1 HRT レベルのメモリ管理

HRT レベルのゲスト OS にはシャドウページングのようなメモリ仮想化手法を適用した場合，ページングのオーバヘッドやページテーブルの用意や操作がタスクのリアルタイム性を損ねてしまい，システムに致命的な事態を招きかねない．そこで，HRT レベルでは，ゲスト OS 起動時にページテーブルの用意や実行中の操作が必要でないよう，物理アドレス=仮想アドレスの対応の記述されたページテーブルを OVM 起動時に用意し，これを利用することで，リアルタイム性を実現することを考えている．また，扱うページサイズの粒度を荒くすることにより，TLB ミスの起こる頻度を小さくすることを実現する．

6.3.2 SRT レベルのメモリ管理

SRT レベルのゲスト OS はプロセスモデルの汎用 OS を想定している．プロセスモデルのシステムは，プロセスごとに仮想アドレス空間を持っていて，それらを切り替えることで，プロセスの切り替えを実現している．このレベルを MVM 上で動作させる場合，プロセスそれぞれはゲスト仮想アドレス空間上で動作し，ゲスト OS は，ゲスト仮想アドレスとゲスト物理アドレスの対応が記述されたページテーブルを切り替えることでプロセスの並行動作を実現しようとする．これに対し，MVM 上でゲスト仮想アドレスと実アドレスの対応を記述したシャドウページテーブルを用意し，こちらを利用することで，メモリの仮想化を実現する．そのため，プロセスの数だけシャドウページテーブルを MVM 上に用意し，プロセスのコンテキストスイッチに応じて，シャドウページテーブルを切り替える．

6.4 I/O 仮想化手法

リアルタイム性を実現する必要があるため，I/O をスケジューリングするといった仮想化処理は，オーバーヘッドがシステムに致命的な事態を引き起こす可能性がある．そのため，MVM ではゲスト OS の I/O 要求をハードウェアにパススルーすることにする．そのため，使用する I/O デバイスは重複を許さない．割り込みに関しては，6.2.3 節で述べたように，対象の VM が CPU 実行権を得ていない場合，CPU 実行権を得た時点で遅延割り込みを行う．

6.5 MVM 領域の隔離

想定するシステムでは，MVM が各ゲスト OS を管理する形であり，ゲスト OS の利用するハードウェア資源を仮想化し，ゲスト OS に与える役割がある．仮にゲスト OS の誤作動が MVM を破壊してしまうと，システムは致命的な状況に落ちてしまう．そのため，ゲスト OS の資源管理を行う MVM は，ゲスト OS よりも高い特権レベルを持

ち，ゲスト OS により，その領域が侵されないよう，ゲスト OS から隔離される必要がある．そのため，ゲスト OS から MVM に処理を移す場合は，例外の発生やハイパバイザーコールなど，特定のパスでのみ可能であるようにする．

6.6 メモリ保護機能

複数のゲスト OS が並列に動作するときに，仮想アドレスの衝突から，あるゲスト OS が別のゲスト OS のメモリ領域を侵害を引き起こす可能性がある．そこで，メモリ保護機能を導入する．ゲスト OS:A の実行中にはゲスト OS:A の物理メモリ領域のみにアクセス可能にし，ゲスト OS:B の物理メモリ領域にはアクセスできないように制御することで，メモリ保護機能を実現する．

6.7 ハイパバイザーコール

ハイパバイザーコールはゲスト OS から MVM を呼び出し，MVM の機能を利用するためのソフトウェア割り込み機能である．

ハイパバイザーコールとして，明示的にハイパバイザーコールを発行し，他の VM に処理を移すディレイ機能を持たせる．これは，OS がアイドルタスクを実行するときのように，実行するべきタスクが無く，タスク起床のイベントを待つような場合に，このディレイ機能を使用する．これにより，他の VM に処理を移して，タスク起床のイベントが来るまで，VM を実行待ち状態にする．

また，割り込みを関連付けられていないイベントを待ちたい場合のディレイ機能もハイパバイザーコールにより実現する．このようなイベントは，割り込みによりイベントが通知されるわけではないので，一定時間ごとに，イベントを確認する必要がある．そのため，この機能には引数として，確認すべきフラグと，確認を実行する時間を渡すことで実現する．

ハイパバイザーコールの機能は仕様変更や機能の追加に応じて随時増えることを考えている．

7. TrustZone による実装方式

本章では，MVM のメモリ管理の設計に基づき，ARM アーキテクチャをターゲットとし，TrustZone を利用した場合の実装方式について述べる．

7.1 スケジューリングに用いる割り込み

ARM プロセッサには通常割り込み (IRQ) と高速割り込み (FIQ) の 2 つの割り込み入力がある．ゲスト OS は自分のプロセスの管理などには IRQ 割り込みを利用し，MVM の VM スケジューリングには FIQ 割り込みを利用する．そのため，FIQ 割り込みが生じた場合，セキュアワールド側の MVM にフック出来るようにコプロセッサのセキュア構成レジスタの設定を行う．2 種類の割り込みはそれぞれ独立したハンドラ，スタックを利用するため，このような構成が可能である．

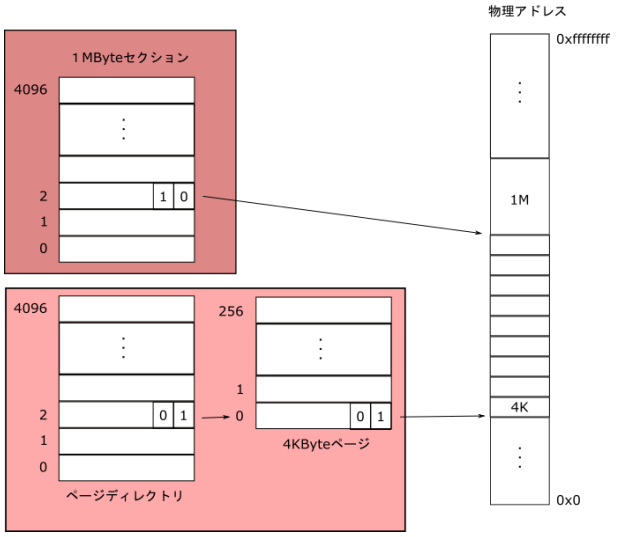

図 4 ページテーブルを利用した変換

7.2 MMU

想定する ARM プロセッサには MMU が実装されており．ページテーブルを利用することで，仮想メモリを物理メモリへ変換やアクセス制御を行う．MMU を利用してアドレス変換を行う場合，ページテーブルを利用する．ページテーブルにより，4GByte の仮想アドレス空間を表現し，それぞれのエントリにより，物理アドレスへの変換を実現する．ページテーブルには二つのレベルがあり，L1 マスタ・ページテーブルと L2 ページテーブルである．L1 マスタ・ページテーブルは 4GByte のアドレス空間を 1MByte のセクションに分割し，4096 のエントリからなる．L1 マスタ・ページテーブルは，そのまま粒度 1MByte の仮想ページを変換するセクション機能と，L2 ページテーブルのページディレクトリとしての機能のどちらかを持つことができる．L2 ページテーブルは 4KByte など，セグメントよりも粒度の細かいサイズのページを扱う．これらの機能は図 4 のように，L1 マスタ・ページテーブルエントリの下位 2bit で識別するため，仮想アドレス空間内で，混在させることが可能である．この異なる粒度のページの混在を許すアーキテクチャが，6.3 章の設計を実現することを可能とする．

7.3 TrustZone を用いた MVM 領域の隔離

VMM を実現するためには，ゲスト OS とハードウェアの間に存在する，ゲスト OS よりも特権でありハードウェアを仮想化し，ゲスト OS に提供できるような環境に VMM が存在する必要がある．そこで，MVM を ARM 上に実装するにあたって，MVM とホスト OS を TrustZone 技術によって，ゲスト OS から不可視で，全てのハードウェアを

利用できるセキュアワールドにて動作させる．ゲスト OS は，限られたハードウェア資源に対してのみアクセスが可能な，ノーマルワールドにて動作させることによって，セキュアワールドにある MVM はゲスト OS の誤動作からも保護することが出来る．

プロセッサの Secure/NonSecure 状態は，コプロセッサのセキュア構成レジスタ (SCR) の NS ビットの状態によって決定する．また，メモリ領域の Secure/NonSecure はページエントリの NS ビットにより判断され，プロセッサが非セキュア状態の時に，セキュアなページにアクセスすることはできず，ゲスト OS からみて，MVM は不可視なものとなる．

7.4 ハイパバイザーコール

ゲスト OS が MVM へ明示的に遷移する方法として，ハイパバイザーコールを用意する．ハイパバイザーコールは，セキュアモニタ例外を発生する SMC 命令を発行することにより，プロセッサのモードをモニタモードへ変更し，セキュアワールドに遷移することにより実現する．さらに SMC 命令の引数として r0 レジスタにハンドラの番号を指定することによって，特定のセキュアモニタコールハンドラを利用でき，ハイパバイザーコールを実現する．プロセッサのモニタモードは Secure/NonSecure を遷移するときに使用するモードであり，SMC 命令を発行することにより，プロセッサのモードをモニタモードへ変更する．SCR の NS ビットを変更することにより，Secure/NonSecure を移動するが，このビットのセットはモニタモードでのみ行われる．

7.5 メモリ保護機構

メモリ保護機構は，コアを共有してゲスト OS に実行時間を割り当てる場合に，TLB 内部で仮想メモリの衝突が起こりえる問題に対応すべく用意する機能である．このメモリ保護機構はドメインによる制御によって実現する．ドメインによる制御は，L1 マスタ・ページテーブルの 4bit のドメインビットによって，メモリ領域に 0 から 15 のドメインを割り当て，図 5 のドメインアクセス制御レジスタ (DACR) のビットによって指定されたドメインの割り当てられたメモリ領域ににのみアクセス可能に設定するアクセス制御である．これを実現するために，ゲスト OS が利用するドメインを一様に決定する．

また，ドメイン情報は TLB にも反映されるため，TLB に異なるゲスト OS の仮想アドレスが存在しても，ドメインビットにより実行中のゲスト OS の TLB エントリが選択されることとなり，TLB 内での仮想アドレスの衝突を防ぐことができる．また，NS ビットの情報も TLB に反映されるため，MVM とゲスト OS のアドレスが TLB 内で衝突することも避けることができる．

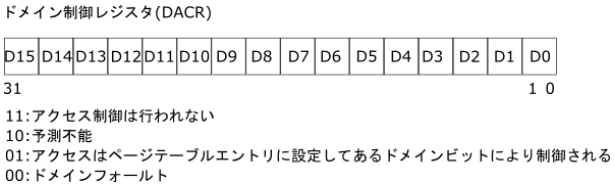

図 5 ドメインアクセス制御 (DACR) レジスタ

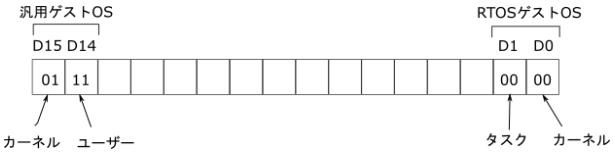

図 6 汎用 OS が実行中の DACR レジスタ

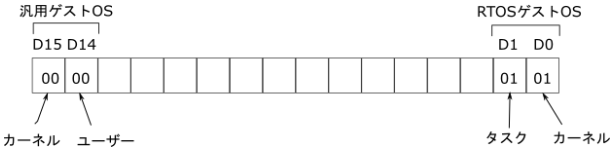

図 7 RTOS が実行中の DACR レジスタ

8. 現状と評価

現状として，MVM において，ノーマルワールド側でのゲスト OS の実行およびハイパバイザーコールを含む例外ハンドラの実装が完了しており，RTOS として，TOPPERS を動作させるテストを行っている．設計に関しては，リアルタイムレベルによって機能を切り分けることに関しての設計については記述した通りであるが，その適応する機能の具体的な部分，CPU であればスケジューリングのアルゴリズム，I/O であれば，I/O を共有しなくてはならない状況になった場合の優先度の設定やセマフォの部分といった部分がまだ出来ていない．

今回評価として，実現している，MVM とゲスト OS の TrustZone により隔離された環境において，ハイパバイザーコールを生成した際のオーバーヘッドを計測した．ハイパバイザーコールを発行すると，例外ハンドラへジャンプするが，今回の計測ではジャンプ先では処理を行わないことで純粋なハイパバイザーコールのオーバヘッドを計測した．結果は平均 0.195µsec のオーバヘッドを生じていることが分かった．このため，ハイパバイザーコールの機能実装の際は，このオーバヘッドを考慮して実装するべきである．

9. おわりに

本研究では，マルチコアプロセッサ環境における組込み向け VMM「MVM」によって，組込みシステムに求められるリアルタイム性，予測可能性の確保とメモリ仮想化処理の問題を解決する手法について述べた．

設定可能なリアルタイムレベルを導入することにより，

二レベルの資源管理手法を提案し，その仮想化方式を提案した．また，ARM TrustZone 技術を用いた MVM 実現手法を提案し，一部について実装を行った．

今後の課題は，仮想化機能部分に適用する手法の具体的な設計，および仮想化手法を実装したうえで，MVM 上で RTOS と汎用 OS を同時に動作させ，本手法の有用性について評価する必要がある．また未設計であるが，必要な機能である VM 間通信方式を設計，実装し，その評価を行いたい．

参考文献

[1] ”Making a Virtual Machine Monitor Interruptible”, Ito Megumi, Oikawa Shuich, Information and Media Technologies 6(4), pp.1139-1148, 2011
[2] ”汎用 OS と専用 OS を高効率に相互補完するナノカーネルの提案と実現”, 新井利明，関口知己，佐藤雅英，木村信二，大島訓，吉澤康文 情報処理学会論文誌 Vol.46 No.10 pp.2492-2504, 2005-10-15
[3] ”Real-Time Multi-Core Virtual Machine Scheduling in Xen”, Sisu Xi, Meng Xu, Chenyang Lu, Linh T.X. Phan, Christopher Gill, Oleg Sokolsky and Insup Lee EMSOFT, ’14 Proceedings of the 14th International Conference on Embedded Software Article, No.27, pp.1-10, 2014.
[4] ”Designing VM Schedulers for Embedded Real-Time Applications”,Alejandro Masur, Thomas Pfeuffer, Martin Geier, Sebastian Drossler and Samarjit Chakraborty, CODES+ISSS ’11: Proceedings of the seventh IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis, pp.29-38, 2011.
[5] ”gandalf vmm における shadow paging の実装と評価”, 伊藤恵, 追川修一, 情報処理学会論文誌:コンピューティングシステム（acs）vol.49 no.sig 2(acs 21) pp.98-112, 2008-03-15
[6] ”組込みマルチコア向け仮想化環境における性能低下抑止手法”，太田貴也，Daniel Sangorrin，本田晋也，高田広章，情報処理学会 第 123 回 OS・第 27 回 EMB 合同研究発表会，東京，Dec 2012.
[7] ”ARM Architecture Reference Manual ARM v7-A and ARMv7-R edition.” ARM Ltd. 2010.
[8] ”Programming ARM TrustZone Architecture on the Xilinx Zynq-7000 All Programmable SoC User Guide.” Xilinx Inc. 2014.
[9] ”マルチコアプロセッサを用いた組込みシステム向け VMM の設計と実装”, 小倉　佑太，佐藤未来子，並木美太郎，研究報告システムソフトウェアとオペレーティング・システム（OS）2013-OS-125, 7, pp.1 - 8, 2013-04-18