Vol.2013-ARC-205 No.7
Vol.2013-OS-125 No.7
2013/4/26

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[1]　佐藤 未来子[1]　並木 美太郎[1]

概要：マルチコアプロセッサを用いた組込み向けハードウェアの性能向上において，リアルタイム処理とITシステムを並列動作させる要求が高まっている．本発表では，マルチコアプロセッサ上のVMMであるOptimus Virtual Machine(OVM)を提案する．OVMでは，リアルタイム制約のない汎用OS，ソフトリアルタイムのOS，ハードリアルタイムのOSないしOSのないハードウェアを直接制御するプログラムの三種類のパラダイムに対して，EDFによるデッドラインスケジューリングによるVM，またはコア固定による最高優先度のVMなどをVMMで管理することにより，異種のリアルタイムに対する要求を適切に調停する．本VMMをARMマルチコアプロセッサをハードコアとして有するXilinx社のZynq上に実装し，遅延やオーバーヘッドを抑えつつ動作することを検証できた．今後の課題として，VMのメモリ管理が挙げられる．

キーワード：マルチコアプロセッサ，仮想化，VMM，組込みシステム

1. はじめに

人工衛星からロボット，携帯電話にいたるまで幅広い分野でマイクロプロセッサが組み込まれているが，組込みシステムにおいてもマルチコアプロセッサや大容量のRAMなどの高い性能をもつハードウェアが採用され[1]，センシングやアクチュエータの制御のようなリアルタイム処理とともにLinux等のITシステムを並列動作させるといった組込みシステムに対する要求が複雑かつ高度になってきている．例えば，ロボティクスのような分野において，体制を制御しつつ画像処理や自然言語処理をするといったことが挙げられる[2]．また，汎用システムにおいてはマルチコアプロセッサと仮想化技術を用いてサーバの仮想化が行われるようになり物理サーバを集約して，複数のシステムを安全に並列動作させることに成功している[3]．

そこで，マルチコアプロセッサと仮想化技術を用いて組込みシステムの要求に応える．異種のリアルタイム性を持つシステムを並列動作させる要求に対して，システムの要求するリアルタイム性を保証する課題がある．また，複数のシステムを並列動作するようハードウェアを仮想化して，ハードウェア資源へのアクセスを調停する課題がある．

そのため，筆者らはマルチコアプロセッサを用いた組込みシステム向けVMM「OVM（Optimus Virtual Machine）」を提案する．そして，マルチコアプロセッサ，メモリおよびI/Oの各要素に対して，リアルタイムに関する属性を与えて，システムの要求するリアルタイム性に応じた資源管理を行うOVMの設計と実装を示す．OVMの評価を行うことで，システムの要求するリアルタイム性を保証して，異種のリアルタイム性を持つシステムを並行・並列動作させる手法について述べる．

以下，第2章でマルチコアプロセッサを用いた組込みシステム向けVMM「OVM」の概要について述べ，第3章でOVMの設計を述べ，第4章でOVMが提供するハイパバイザコールについて述べ，第5章で実装について述べる．そして，第6章でOVMの評価と考察を示し，第7章で関連研究を示した後，第8章でまとめる．

2. 組込みシステム向けVMM「OVM」

本章ではマルチコアプロセッサを用いた組込みシステム向けVMM「OVM」について述べる．

2.1 本研究の課題

仮想化技術により異種のリアルタイム性を持つシステムを並列・並行動作させた際に，それぞれのシステムの持つリアルタイム性を保証するという課題がある．リアルタイム性については，アクチュエータの制御のように実時間と

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

ⓒ 2013 Information Processing Society of Japan

高い精度で同期して処理を行う必要があるため，ハードウェア割り込みの応答時間やタスクの持つデッドラインが指標となる．

仮想化技術を用いる場合，仮想化処理の遅延やオーバーヘッドが生じる．特にシングルコアプロセッサでは，複数のシステムを同時に動作させる際に VM の切り替えにより，ハードリアルタイム性を要求する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，コアごとに VM を分けることのできるマルチコアプロセッサの活用が想定される．また，VM ごとに利用するコアを分ける VMM や優先度に基づいてコアを共有する VMM は存在するが [8][9]，コアごとに占有や共有といった方式を分けることのできる VMM は少なく，メモリや I/O も同様にシステムの要求するリアルタイム性に応じて仮想化する VMM は存在しない．より多くのシステムのリアルタイム性を保証するには，計算機を構成する要素おのおののリアルタイム性に合わせた仮想化が必要である．

そのため，限られたハードウェア資源に対して遅延やオーバーヘッドを抑え，異種のリアルタイム性を持つ複数のシステムの要求するリアルタイム性を保証することが本研究の課題である．

2.2 OVM の目標

OVM は，2.1 節に示した課題に対して次の目標を達成する．

(1) システムの要求するリアルタイム性の保証

組込みシステムでは，センシングやアクチュエータ制御のようにナノ秒オーダーのハードリアルタイム処理やマイクロ秒オーダーのソフトリアルタイム処理を行うため，ハードウェア割り込みの最悪応答時間やタスクのデッドラインミスといったリアルタイム性を保証する必要がある．また，IT 処理ではリアルタイム性よりも，システムの平均応答時間を向上させ，全体のスループットを改善することが必要とされる．そのため，組込みシステムを構成する各々の要素が要求するリアルタイム性を保証することで複数のパラダイムを実現することが OVM の目標である．

(2) 複数のパラダイムのシステムの並行・並列動作

今後の組込みシステムでは，リアルタイム処理と IT 処理を同時に並列で処理できる必要がある．そこで，リアルタイム制約のない汎用 OS，ソフトリアルタイムの OS，ハードリアルタイム OS ないしハードウェアを直接制御するプログラムといったリアルタイム性について異なる要求を持つシステムをマルチコアプロセッサの仮想化により並行・並列動作させて，リアルタイム処理と IT 処理を組合せたシステムを構築できることが OVM の目標である．

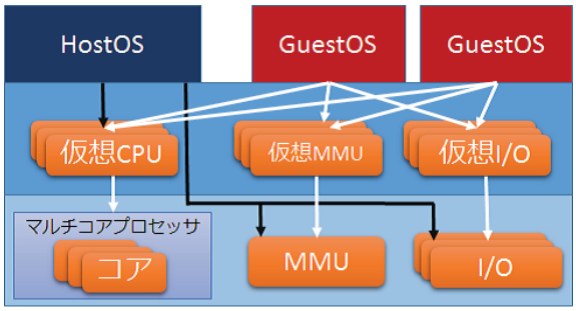

図 1 OVM の仮想化方式

2.3 OVM の概要

OVM はシステムの要求するリアルタイム性を保証するために，各々のハードウェアリソースに対する遅延やオーバーヘッドを抑える必要がある．そのため，OVM はゲスト OS に修正を加えることで遅延やオーバーヘッドを改善する準仮想化方式をとる．具体的には図 1 のような準仮想化ネイティブハイパバイザ (type1) の方式である．

OVM の機能として，マルチコアプロセッサの仮想化，メモリの仮想化および I/O の仮想化を行う機能が複数あり，システムの要求するリアルタイム性に応じて仮想化を行う機能を選択できる構成をとる．具体的には，仮想 CPU や I/O 要求のリアルタイムを意識したスケジューリングアルゴリズムによるデバイスを共有する方式とマルチコアプロセッサの利用するコアを分けるといったパーティショニングによるハードリアルタイムな方式である．

また，OVM の独立性や移植性を高めると共にバイナリサイズを抑えるために，OVM は仮想化機能に特化して，OVM の管理インタフェースや余剰ハードウェアリソースの管理の役割をホスト OS が持つ．さらに，I/O の仮想化方式の変更に柔軟に対応するために，ホスト OS が仮想化処理を代行する仮想化支援の役割もホスト OS が持つ．

そのため，OVM は KVM[4] のようにホスト OS のカーネルモジュールとして動作する方式をとり，ホスト OS と連携しやすく，ハードウェアを直接制御できる位置につく．KVM と異なるのはマルチコアプロセッサの仮想化方式であり，KVM のようにホスト OS のプロセスとして仮想 CPU を管理する方式では，システムの要求するリアルタイム性を保証できないため，OVM が仮想 CPU を管理して，ホスト OS はゲスト OS のように仮想 CPU により資源割り当てが行われる点が異なる．ホスト OS にはリアルタイム制約のない Linux のような汎用 OS を想定し，汎用向け仮想 CPU が OVM から提供される．

3. OVM の設計

本章では，OVM におけるシステムのリアルタイム性に応じた計算機の仮想化方式について述べる．

図 2 ソフトリアルタイムの資源割り当ての優先度の決定

3.1 リアルタイムレベル

2.1 節で示したとおり，より多くのシステムのリアルタイム性を保証するためには，システムが要求するリアルタイム性に応じた資源割り当て方式を選択する必要がある．

そこで，OVM では仮想化対象のハードウェア資源に対して，次に示す三種類のリアルタイム性に関する属性を付与する．リアルタイム性に関する属性に応じて仮想化方式を決定する．

- ハードリアルタイム向け (HRT)
  資源を共有する仮想化アルゴリズムがシステムに致命的な事態を引き起こすシステムがある．そのため，他のシステムから独立してリアルタイム性を保証するマルチコアプロセッサのコア占有や I/O パススルーなどのパーティショニングによる仮想化を行う．
- ソフトリアルタイム向け (SRT)
  他のシステムと資源を共有しても，システムの要求するリアルタイム性を保証できるシステムがある．そのため，優先的にハードウェアリソースを獲得しつつも，利用していない時間や領域は他のシステムに譲り，ハードウェアリソースを共有するアルゴリズムによる仮想化を行う．特にソフトリアルタイム向けの場合，図 2 のようにゲスト OS が OVM に対してデッドライン等のリアルタイム情報を提供することで，ハードウェアリソースの割り当ての優先度を決定する．
- 汎用向け (NRT)
  リアルタイム性を必要とせず平均応答時間の向上を要求するシステムがある．そのため，他のシステムと平等な資源分配により平均応答時間を向上させ，システム全体のスループットを改善するアルゴリズムによる仮想化を行う．

リアルタイムレベルは，ゲスト OS のパラダイムに対応している．ハードリアルタイム OS やハードウェアを直接制御するプログラムは VM の切り替えによるコストを無視できないので HRT に該当し，ソフトリアルタイム OS は必要な時に優先的に資源を要求するので SRT に該当し，リアルタイム制約のない汎用 OS はリアルタイム性よりもスループットを要求するので NRT に該当する．

3.2 マルチコアプロセッサの仮想化

異なるパラダイムのゲスト OS を並行・並列動作させるためにはマルチコアプロセッサの仮想化が必要不可欠である．そこで，OVM ではマルチコアプロセッサのコアを仮想化して，仮想 CPU を提供する．仮想 CPU はリアルタイムレベルに応じて次に示す機能を提供する．

3.2.1 コア占有機能

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タスクはシステム全体に致命的な事態を引き起こす．マルチコアプロセッサの仮想化を行う際に，コアの実行権が時間とともに遷移する方式をとると，仮想 CPU のスケジューリングやディスパッチの際に，遅延やオーバーヘッドが生じて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，ハードリアルタイム向けの仮想 CPU はマルチコアプロセッサの特定のコアを占有して，仮想化による遅延やオーバーヘッドをなくし，システムの最悪応答時間の増加を抑える．ハードリアルタイム向けの仮想 CPU が遅延やオーバーヘッドなしに物理コアを利用する機能を提供する．なお，ハードリアルタイムでない OS はコア専有せずに仮想 CPU 上で実行する．

3.2.2 リアルタイムスケジューリング機能

OVM ではゲスト OS のリアルタイム情報を用いて，リアルタイム性を保証するスケジューリングアルゴリズム [5] によって仮想化を行う．リアルタイムスケジューリング機能は，ソフトリアルタイム向け仮想 CPU および汎用向け仮想 CPU の機能であり，ソフトリアルタイム向け仮想 CPU が優先的にハードウェアリソースを獲得するための機能である．

特定のシステムがマルチコアプロセッサの特定のコアを占有し続けると，コアの処理能力が不要になる時間においても，他のシステムに実行権が移らず，システム全体のスループットが低下するおそれがある．そのため，実 CPU を複数のゲスト OS で共有するための仮想 CPU のスケジューリング機能が必要となる．しかし，KVM などの既存の VMM のラウンドロビンに代表されるスケジューリングアルゴリズムでは，システムの要求するリアルタイム性を保証することができない．そこで，リアルタイム性を保証するアルゴリズムの一つである EDF（Earliest Deadline First）によるスケジューリング機能を設ける．

3.2.3 割り込みスケジューリング機能

プロセッサには，割り込みと呼ばれる実行中の処理を中断して，強制的に指定した処理を実行させる機能がある．OVM では，複数のパラダイムのシステムが並行・並列動作する場合がある．このとき，割り込み処理を行うべきゲスト OS の仮想 CPU がコアの実行権を持っていない場合があり，いつ割り込み処理を行うべきかという問題がある．例えば，割り込み対象のコアで実行中の処理の方が割り込み処理よりも優先度が高い場合である．

そのため，OVM は割り込み処理を実行するタイミングを調整する割り込みスケジューリング機能を持つ．この機能はリアルタイムスケジューリングにより決定された優先度を基に割り込みタイミングを決定するもので，対象のコ

アに割り当てられている仮想 CPU の処理が一番優先度を高いものとする．コアが割り当てられている仮想 CPU の割り込み処理は即時に行い，コアが割り当てられていない仮想 CPU の割込み処理は，仮想 CPU のディスパッチタイミングで行う．この割込みスケジューリングによって，システムの要求するリアルタイム性を保証しつつ割込み処理を実現する．

3.3 メモリの仮想化

組込みシステムに対して大容量メモリを用いる例が増えてきている．複数のシステムを並行・並列動作させるためには，ゲスト OS のメモリアクセスを調停する必要がある．そこで，OVM では物理アドレス空間を仮想化して，仮想 MMU を提供する．仮想 MMU はリアルタイムレベルに応じて次に示す機能を提供する．

3.3.1 メモリセパレーション

ハードリアルタイム性を要求するシステムでは，シャドウページングによる仮想化処理を行うと，ページテーブルの一貫性管理やページテーブルの操作などのオーバーヘッドが生じてしまい，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，OVM ではまずアドレスを連続領域として割り当てる．メモリセパレーションを行うことで，ハードリアルタイム性を保証する．ハードリアルタイム向け仮想 MMU が仮想化による遅延やオーバーヘッドなしに，メモリ領域を提供するための機能である．

3.3.2 シャドウページング

リアルタイム性の高い OS はメモリセパレーションで実行するが，NRT および SRT のリアルタイムレベルを要求する OS はメモリの利用効率が重要である．複数のシステムが並行・並列動作する際に同一のアドレス空間を扱うと，アドレスの衝突が発生して安定性に問題が生じる．また，物理メモリのどこにでもアクセスできるので悪意のあるアクセスを防ぐことができず，安全性に問題が生じる．

OVM で物理アドレス空間を仮想化して，ゲスト OS ごとのアドレス空間を提供して，OVM 内部でメモリアクセスを調停する必要がある．そこで，OVM はシャドウページング [6] の手法を用いて，ゲスト OS の仮想アドレスを OVM の物理アドレスに変換する機能を持つ．つまり，ソフトリアルタイム向け仮想 MMU および汎用向け仮想 MMU が安全で効率よくメモリ領域を活用するための機能となっている．

3.4 I/O の仮想化

複数のシステムを並行・並列動作させるためには，ゲスト OS の発行する I/O 要求を調停する必要があり，さらに，I/O デバイスのエミュレーションなどを要求される場合がある．そこで，OVM ではメモリマップド I/O およびハードウェア割込みを仮想化して，仮想 I/O を提供する．仮想 I/O はリアルタイムレベルに応じて次に示す機能を提供する．

3.4.1 OVM による I/O 代行

複数のシステムが並列・並行動作する際に同時に I/O アクセスにより，直接ハードウェアを制御した場合システムの管理者の意図通りに動作せず混乱が生じる可能性がある．

そのため，OVM ではゲスト OS の発行した I/O 要求を OVM 内部で I/O デバイスごとに用意された仮想化関数により仮想化を行う機能を持つ．この機能はスケジューリングによる時分割や帯域制御など，個々の I/O デバイスに応じた仮想化機能を拡張することで，システムの要求するリアルタイム性を保証しつつ，複数のシステムの I/O 要求を調停することで，全体の I/O を実現する．

3.4.2 ホスト OS による I/O 代行

OVM による I/O 代行では，OVM のプログラム自体を変更する必要があるため，I/O デバイスの構成の変化に柔軟に対応することができない．そのため，OVM ではホスト OS のプロセスを利用して，I/O デバイスのエミュレーションなどの I/O 仮想化を行い，I/O デバイスの構成の変化に対応することのできる I/O 仮想化を実現する．

具体的には，I/O 番号と VM 番号によって異なるプログラムによりサービスを提供する．また，ホスト OS-OVM 間通信を用いることで，仮想的なハードウェア割り込みをゲスト OS に提供する．

3.4.3 I/O パススルー

OVM やホスト OS による I/O 要求の代行による仮想化では，遅延やオーバーヘッドが生じて，ハードリアルタイム性を要求するシステムにおいては，致命的な事態を引き起こす可能性がある．

そのため，OVM ではあらかじめ定められたルールにより，特定のシステムのみが特定の I/O アクセスを行う場合にのみ，I/O 要求をパススルーして，ゲスト OS が I/O デバイスを直接制御することで，ハードリアルタイム性を要求するシステムのリアルタイム性を保証する．つまり，ハードリアルタイム向け仮想 I/O が遅延やオーバーヘッドなしに I/O アクセスを行うための機能である．

3.4.4 ハードウェア割り込みルーティング

複数のシステムが並列・並行動作している場合に，発生したハードウェア割り込みをどのシステムで処理するかという問題が生じる．

そこで OVM では，発生したハードウェア割り込みをルーティングして，特定のシステムに通知する機能を持つ．なお，通知された割り込みは割り込みスケジューリングによりスケジュールされた後に処理される．ただし，ハードリアルタイム向け仮想 I/O についてはルーティングせずにそのまま割り込み処理を行う．

表 1　ゲスト OS へ提供する OVM の HVC 一覧

名称	引数一覧	返り値
set_realtime_info	デッドライン，周期，実行時間	なし
仮想 CPU スケジューラにリアルタイム情報を伝える．		
vm_exit	なし	なし
ゲスト OS のタスクの周期処理が終了したことを通知する．		
output	メモリアドレス，I/O 要求	なし
OVM に I/O 要求を発行する．		
input	メモリアドレス	処理結果
OVM に I/O 要求を発行する．		

表 2　ホスト OS へ提供する OVM の HVC 一覧

名称	引数一覧	返り値
read_io	なし	vCPUID，アドレス，I/O 要求
汎用向け I/O が発行した I/O 要求を受け取る．		
write_io	vCPUID，I/O 結果	なし
汎用向け I/O が発行した I/O 要求に応える．		
register	VM の構成ファイル	なし
VM 構成ファイルから VM を構築する．		
info	なし	VM の動作状況
動作中の VM と仮想資源の一覧を取得する．		

4. OVM のハイパバイザコール

仮想 CPU スケジューラなどゲスト OS のリアルタイム情報と連携して，リアルタイム性を保証する．そのため，ゲスト OS-OVM 間で通信をする必要がある．表 1 にゲスト OS に提供する OVM の API であるハイパバイザコール（HVC：HyperVisor Call）の一覧を示す．

前章までに述べたように，ホスト OS は OVM の管理インタフェースやデバイスエミュレーションなどの仮想化支援機能を持つ．そのため，ホスト OS-OVM 間で通信をする必要がある．表 2 にホスト OS に提供する OVM の HVC 一覧を示す．

5. OVM の実装

本章では OVM の設計と Xilinx Zynq による実装について述べる．

5.1 実装環境

ARM デュアルコア Cortex-A9 搭載の組込みボードである Xilinx Zynq-7000 SoC ZC702 に OVM の実装を行う．特に Cortex-A9 に搭載されているセキュリティ拡張機能である TrustZone を活用して，割り込みやメモリアクセスの権限のコントロールを行うことで OVM の実現を図る．

また，ホスト OS として Xilinx Zynq Linux を用いる．Xilinx Zynq Linux は Linux Kernel 3.3.0 をベースにしたものである．

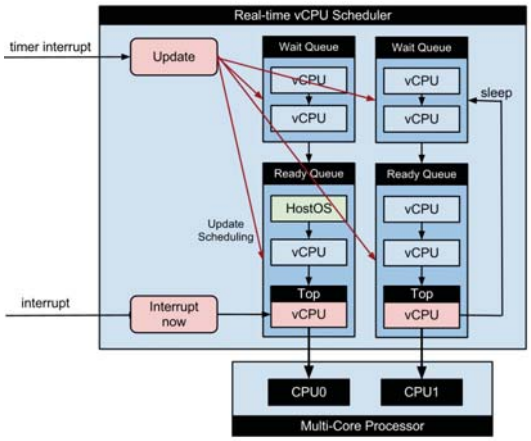

図 3　仮想 CPU スケジューラの構成

5.2 仮想 CPU の実装

5.2.1 仮想 CPU スケジューラの実装

OVM は複数のシステムを並行動作させるために仮想 CPU スケジューラを持つ．また，システムの要求するリアルタイム性を保証するために，リアルタイム性を意識したアルゴリズムである Earliest Deadline First(EDF) のアルゴリズムによるリアルタイムスケジューリングを行う．よって，仮想 CPU スケジューラは図 3 のような構成をとる．

仮想 CPU スケジューラは優先度付キューによって優先度を決定し，最高優先度の仮想 CPU を物理コアにディスパッチする．優先度については，デッドライン，タスクの実行時間，周期といったタスクのリアルタイム情報によって決定される．また，優先度付きキューは実行可能キューと待ちキューの二つを用意して，汎用向け仮想 CPU は優先度は低いが常に実行可能キューにおき，ソフトリアルタイム向け仮想 CPU は処理が終了すると待ちキューに移動し，時間とともに実行可能キューに戻る．このよう状態遷移を起こすことで，特定の区間ではソフトリアルタイム向け仮想 CPU が優先的に物理コアを獲得させ，処理の必要のない区間では汎用向け仮想 CPU が処理を行う．

この仮想 CPU スケジューリングにより，システムの要求するリアルタイム性を保証しつつ，平均応答時間を向上させてスループットを改善する．

5.2.2 コア占有機能の実装

コア占有機能を実現するために，特定のコアにホスト OS が動作するようにホスト OS の CPU マスクを変更する．また，コア占有機能を用いる仮想 CPU で処理しないハードウェア割り込みについても割り込み先のコアを変更する必要がある．

OVM 内部では，物理コアの状態を記憶するテーブルを用意して，その物理コアに割り当てられる可能性のある仮想 CPU の中で一番高いリアルタイム性を要求するシステムを記憶する．そして，空のキューをもつ物理コアに対し

ⓒ 2013 Information Processing Society of Japan

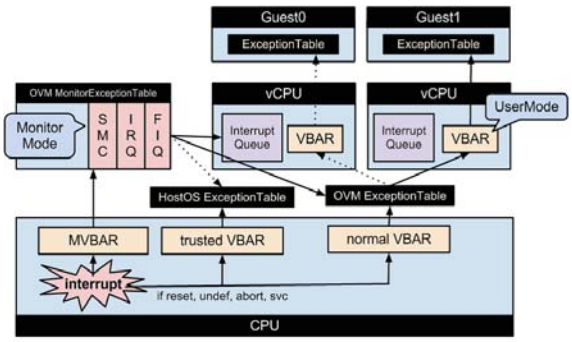

図 4　仮想割り込み機構

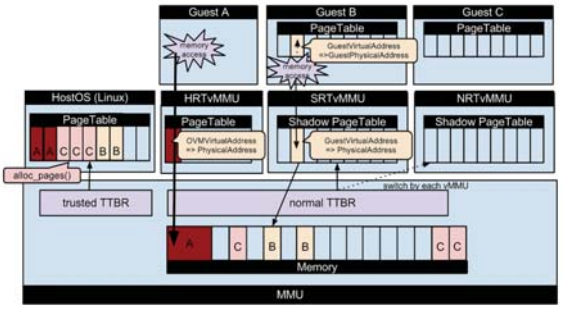

図 5　仮想 MMU 機構

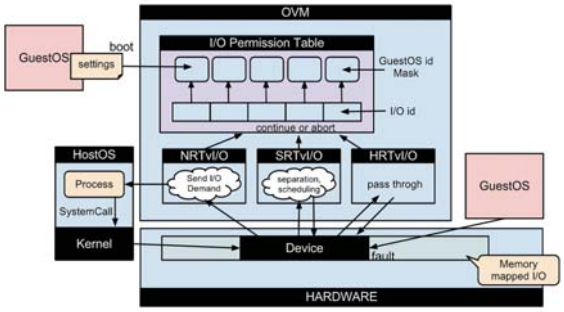

図 6　仮想 I/O 機構

て，ハードリアルタイム向け仮想 CPU を割り当て，新しく仮想 CPU を対象のキューに追加を禁止することで，コア占有機能を実現する．

5.2.3　仮想割り込みの実装

割り込み先の仮想 CPU に物理コアが割り当てられている場合はそのまま割り込み処理を行い，物理コアが割り当てられていない場合は，仮想 CPU ごとに用意されたキューに退避した後，仮想 CPU のディスパッチタイミングで割り込み処理を行う．

割り込みの仮想化を行うために，発生した割り込みを一旦 OVM が受け取って，割り込み処理を行うべき仮想 CPU へ伝達する必要がある．そのため，図 4 のように例外ベクタテーブルを TrustZone によって保護することで，OVM が処理する余地を残す．

TrustZone が搭載されている場合に割り込みが発生した際に，モニタモードと呼ばれる最も高い権限を持つモードで，特別な例外ベクタテーブルにより割り込み処理を行うことができる．そこで，ソフトリアルタイム向け仮想 CPU および汎用向け仮想 CPU の場合は，モニタモードで割り込みを受け取り，割り込み処理を行うべき仮想 CPU で処理できるように伝達する．

5.3　仮想 MMU の実装

ハードリアルタイム性を要求するシステムでは，ゲスト OS の仮想アドレスを OVM の物理アドレスに変換するページング処理による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．そこでハードリアルタイム向け仮想 MMU は，メモリセパレーションの機能を持つ．このメモリセパレーションはページテーブルのラージページを直接割り当てることで，オーバーヘッドを抑える実装とした．

また，ソフトリアルタイム向け仮想 MMU や汎用向け仮想 MMU は，安定性や安全性の観点からメモリ保護の要求が高まっている．そのため，OVM でシャドウページングを行い，ゲスト OS の仮想アドレスを OVM の物理アドレスに変換する．シャドウページングの機構は，ゲスト OS 内部に持つゲスト OS の仮想アドレスをゲスト OS の物理アドレスに変換するページテーブルとは別に，ゲスト OS ごとにゲスト OS の仮想アドレスを OVM の物理アドレスに変換するページテーブルを OVM 内に持たせ，ゲスト OS 内部のページテーブルのエントリの追加や更新を検知して，OVM のページテーブルを更新することでシャドウページングを実現する．よって，仮想 MMU は図 5 のような構成をとる．なお，シャドウページングは未実装となっている．

5.4　仮想 I/O の実装

すべてのパラダイムの仮想 I/O について，メモリマップド I/O を仮想化して，ゲスト OS に対して仮想的なメモリマップド I/O を提供する．そのため，メモリアクセスを仮想化することと密接に関係しており，特定の領域を保護して例外として OVM が受け取ることで仮想化する．この機能を実現するには TrustZone を用いる．TrustZone では非セキュア状態の時に例外を起こすようにページテーブルのエントリに設定できる．そのため，ゲスト OS を非セキュア状態で動作させて，メモリアクセスによる I/O 要求を例外として OVM が受け取り，図 6 のように，ハードリアルタイム向け仮想 I/O では I/O 要求をハードウェアにパススルーする．ソフトリアルタイム向け仮想 I/O では OVM 内部の I/O デバイスごとの仮想化機能により仮想化処理を行う．汎用向け仮想 I/O では，ホスト OS のプロセスに I/O 要求の代行処理を行うことで I/O デバイスのエミュレーションなどの仮想化処理をする．

5.5 ゲスト OS-OVM 間通信の実装

OVM では，ゲスト OS と OVM 間でデッドラインなどのリアルタイム情報を交換する必要がある．ゲスト OS-OVM 間通信では，TrustZone の機能の一つであるセキュアモニタコールを用いる．また，例外ベクタテーブルの SMC 例外の部分を書き換え，ハイパバイザコールの関数ポインタのテーブルに対して，汎用レジスタ r0 をインデックスとしてジャンプするように実装した．

5.6 ホスト OS-OVM 間通信の実装

OVM では，ホスト OS と OVM 間で VM の構成情報や稼働している VM の情報等をやりとりする必要がある．ホスト OS-OVM 間通信では，Linux の/proc ファイルシステムを用いる．OVM をインストールする際に，ホスト OS の Linux の/proc 以下に/proc/ovm を作り，この/proc/ovm を通じて通信を行えるようにした．

6. 評価と考察

本章では OVM のマルチコアプロセッサの仮想化機能についての評価と考察を述べる．

6.1 評価対象

OVM の最も重要な機能であるマルチコアプロセッサの仮想化を中心に Xilinx Zynq に実装して評価を行う．マルチコアプロセッサの仮想化機能のすべてと，割り込みの仮想化機能，メモリセパレーションおよび I/O パススルー機能，ゲスト OS-OVM 間通信およびホスト OS-OVM 間通信について実装して評価を行った．実装の結果，ソースコードは 3965 行でバイナリは 36KB となった．

6.2 ハードウェア割り込みの遅延時間

リアルタイム性を示す指標としてハードウェア割り込みの最悪応答時間がある．そのため，ハードウェア割り込みが発生してからゲスト OS の割り込み処理が開始されるまでの遅延時間を計測して，OVM がシステムの必要とするリアルタイム性を保証できることを示す．また，この遅延時間は短いほど高いリアルタイム性を発揮していることを示し，汎用，ソフトリアルタイム，ハードリアルタイムになるにつれ遅延時間が短くなると予想される．計測には Zynq のプロセッサコアごとに用意されているプライベートタイマを用いる．なお，プロセッサコアは 800MHz でプライベートタイマは 400MHz で動作する．

6.2.1 ハードリアルタイム向けの遅延時間

ハードリアルタイム向けの仮想 CPU における割り込み処理では，割り当てられている物理コアで発生した割り込みについては，仮想化を行わず，直接ゲスト OS の割り込み処理が開始される．

そして，ハードウェア割り込みを 1000 回発生させたところ，どの割り込みでも 70ns かかり，OVM のない状態においても例外ベクタテーブルへジャンプするのに 70ns かかるため，ハードリアルタイム向けのハードウェア割り込みでは遅延が発生しないことがわかる．

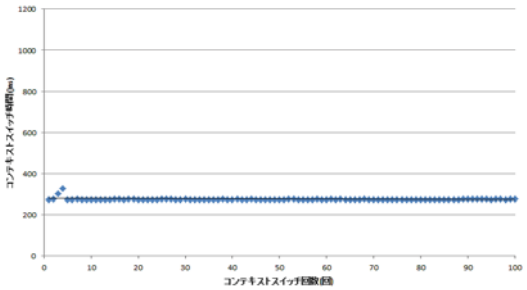

図 7 OVM のコンテキストスイッチ時間

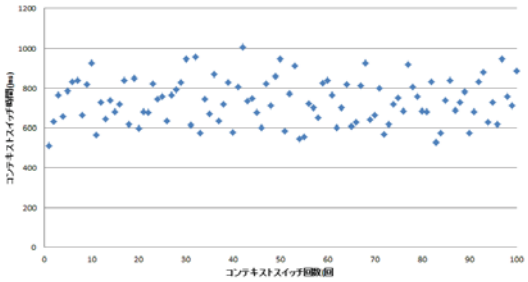

図 8 Linux のコンテキストスイッチ時間

6.2.2 ソフトリアルタイムおよび汎用向け

ソフトリアルタイムおよび汎用向け仮想 CPU においても同様に 1000 回計測したところ，最良 140ns 最悪 170ns 平均 150ns であった．マイクロオーダのアクチュエータの制御等には利用可能な遅延時間であることがわかる．

6.3 デッドラインミスの評価

ソフトリアルタイム向け仮想 CPU および汎用向け仮想 CPU は，タスクのリアルタイム情報によって仮想 CPU がスケジューリングされる．そのため，複数のゲスト OS が同一のコアで並行して動作するとき，デッドラインを持ったリアルタイムアプリケーションのリアルタイム性を保証できるか評価を行う．評価用のアプリケーションはアニメーションのように定期的に描画を行う処理を想定して，シリアル通信を用いて CUI で描画するプログラムを用いた．

100μs ごとのタイマ割り込みでスケジューリングを行う仮想 CPU スケジューラを用いてデッドラインミス率を計測したところ，200μs の範囲でデッドラインミスを捉えることできて，デッドラインミスハンドラを実行することができた．

6.4 コンテキストスイッチのオーバーヘッドの評価

ソフトリアルタイムおよび汎用向け仮想 CPU は，リアルタイム性を考慮したアルゴリズムによりスケジューリングされ，物理 CPU にディスパッチされる．そして，仮想

CPU のコンテキストスイッチはオーバーヘッドとなりリアルタイム性の保証に影響を与える．そこで，OVM の仮想 CPU のコンテキストスイッチ時間と既存の VMM である KVM を想定した Linux のプロセスのコンテキストスイッチを計測する．計測した結果，OVM は図 7 となり，Linux は図 8 となった．

OVM の仮想 CPU のコンテキストスイッチ時間は，最良 270ns 最悪 330ns 平均 280ns であった．Linux のプロセスのコンテキストスイッチ時間は，最良 510ns 最悪 1000ns 平均 720ns であった．このことから，KVM のような Linux のプロセスを利用する VMM に比べて，OVM は高速で動作しつつ，実行時間の予測可能性も高い．

7. 関連研究

SPUMONE[7] は単一の物理プロセッサを複数の仮想プロセッサに多重化する仮想化レイヤである．仮想プロセッサそれぞれを異なる OS に割り当てることによって，ひとつのシステムで複数の OS を実行することができる．SPUMONE はプロセッサに注目した仮想化方式であるのに対し，OVM はコンピュータの構成要素のそれぞれに対してシステムの要求するリアルタイム性を保証して複数のパラダイムの実現をする．

RT-VMM[8] は RTOS と高機能 OS を共存動作を可能にするシステムである．Xen を改良したシステムであり，ドメインと呼ばれる単位で管理している．プロセッサ数以上のゲスト OS が存在する場合のタイマ割り込み周期のゆらぎを解決する手法が盛り込まれている．OVM においても，コア数以上のゲスト OS が動作する場合があるため，活用できる可能性がある．

SageG[9] は組込みシステム向けのデュアル OS モニタである．SafeG では汎用 OS と RTOS を同時実行を実現するために，OVM 同様 TrustZone を用いている．SafeG は，二つの OS が同時に動作して，物理コアごとに独立して OS を切り替えることができる．対して，OVM は物理コアごとに OS を切り替えることが可能であると同時に，切り替えることのできる OS の数に制限がない．また，ゲスト OS のパラダイムに応じて，遅延，オーバーヘッドおよびスループットを考慮した仮想化処理を行うことができる．

BitVisor[10] はセキュア VM で，準パススルー型の仮想マシンモニタである．既存の OS に対して，ハードディスクの暗号化などのセキュリティ機能を拡張することができる．対して，OVM はハードウェアへのアクセスに対してセキュリティ機能ではなく，リアルタイム機能を加えたものとなっている．

8. まとめ

マルチコアプロセッサを用いた組込みシステム向け VMM「OVM」によって，システムの要求するリアルタイム性の保証および複数のパラダイムのシステムの並行・並列動作という問題を解決する手法について述べた．そして，組込みシステムに用いられるハードウェアを仮想化して，複数のパラダイムのゲスト OS が動作できるよう設計を示し，Zynq 上で試作し，方式を検証した．

今後の課題として，MMU および I/O の仮想化機能の実装と評価が挙げられる．具体的には，シャドウページングの実装と評価，OVM による I/O 要求の代行機能の実装と評価およびホスト OS のプロセスを活用した仮想化支援機能の実装と評価である．これらについても，同様に実装と評価を行い，システムの要求するリアルタイム性を保証しつつ，複数のパラダイムのゲスト OS が並行・並列動作できることを示すことが今後の課題である．

参考文献

[1] 阿部 剛，酒井 淳嗣：組込み向けマルチコアプロセッサ *MPCore* を用いた応答性/機能性両立環境評価 *:* 制御処理と情報処理の融合にむけて *(*マルチコア・マルチプロセッサ, 組込技術とネットワークに関するワークショップ *ETNET2008)*, 情報処理学会研究報告. EMB, 組込みシステム 2008(32), 19-24, 2008-03-27.

[2] 田中 穂積：言語理解とロボットの行動制御 *:* 音声認識から音声理解へ, 情報処理学会研究報告. SLP, 音声言語情報処理, pp. 37-42, 2000-12-14.

[3] 瀬川大勝，辻澤隆彦，辰己丈夫：仮想化技術を用いたサーバ集約と演習端末室の構築, 学術情報処理研究 No.15, pp. 134-141, 2011-09-14.

[4] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin：*kvm: the Linux Virtual Machine Monitor*, Proceedings of the Linux Symposium, Ottawa, Ontario, 2007, 2007.

[5] 松原 豊，本田 晋也，高田 広章：割込み処理を含むリアルタイムアプリケーション統合のための階層型スケジューリング, 情報処理学会研究報告. EMB, 組込みシステム 2009-EMB-14(7), pp. 1-9, 2009-07-17.

[6] 伊藤 愛，追川 修一：*Gandalf VMM* における *Shadow Paging* の実装と評価 *(*仮想化*)*, 情報処理学会論文誌. コンピューティングシステム 49(SIG_2(ACS_21)), pp. 98-112, 2008-03-15.

[7] Wataru Kanda, Yu Yumura, Yuki Kinebuchi, K Makijima, and Tatsuo Nakajima：*SPUMONE: Lightweight CPU Virtualization Layer for Embedded Systems*, Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on , vol. 1, pp. 144-151, Dec., 2008.

[8] 金城 聖，永島 力，元濱 努，片山 吉章，毛利 公一：リアルタイム仮想化ソフトウェア基盤におけるタイマ割込み通知機構 *(*リアルタイムシステム*)*, 情報処理学会研究報告. EMB, 組込みシステム 2008(116), pp. 9-16, 2008-11-20.

[9] Daniel Sangorrin, Shinya Honda and Hiroaki Takada：*Reliable Device Sharing Mechanisms for Dual-OS Embedded Trusted Computing*, Proceedings 5th International Conference on Trust and Trustworthy Computing, pp. 74-91, Vienna, Austria, Jun 2012.

[10] 保理江 高志，榮樂 英樹，品川 高廣，加藤 和彦：仮想マシンモニタ *BitVisor* と *I/O* 仮想化技術 *(*仮想化*)*, 情報処理学会研究報告. [システムソフトウェアとオペレーティング・システム] 2009(6), pp. 35-41, 2009-01-21.