Vol.2013-SLDM-160 No.28
Vol.2013-EMB-28 No.28
2013/3/14

# リアルタイム組込みシステム向けの リソースパーティショニング用ハードウェア支援技術の開発

本村哲朗[†1]　近藤雄樹[†1]　山田哲也[†1]　高田雅士[†1]　仁藤拓実[†1]
野尻徹[†1]　十山圭介[†1]　斎藤靖彦[†2]　西博史[†2]
佐藤未来子[†3]　並木美太郎[†3]

今日の組込みシステムは，リアルタイム制御と情報処理のような独立の複数の機能を扱う必要があり，マルチコア上への搭載が有効である．この時，リソース保護のため，仮想化の一技術であるリソースパーティショニングが必要となる．我々は，リアルタイム性の実現に向けリソースパーティショニングのオーバヘッドを削減する，ハードウェア支援技術 ExVisor/XVS を開発した．その主要技術は物理アドレス管理モジュール PAM*で，組込みシステムのメモリ利用方法の特徴を活かした階層のないページテーブルによるダイレクトなアドレス変換で高速化を図る．RTL シミュレーションと FPGA 実装で評価を行った結果，シングルコアと比較してリソースアクセス時のオーバヘッドは高々5.6%であることを確認した．*PAM : Physical Address Management module

# Hardware Support for Resource Partitioning in Real-Time Embedded System

Tetsuro Honmura[†1], Yuki Kondoh[†1], Tetsuya Yamada[†1], Masashi Takada[†1],
Takumi Nitoh[†1], Tohru Nojiri[†1], Keisuke Toyama,
Yasuhiko Saitoh[†2], Hirofumi Nishi[†2], Mikiko Satoh[†3], and Mitaro Namiki[†3]

**Abstract** Today's embedded systems require multiple functions such as real-time control and information technology and integrating these functions on a multi-core processor is one effective solution. However, this increases overhead as it is necessary to partition resources in this approach to protect them. We developed hardware support called ExVisor/XVS to reduce the overhead of partitioning resources to achieve real-time characteristics. This features a physical address management module (PAM) that uses direct address translation by using a single level page table based on an embedded system's memory usage. We evaluated the overhead in a virtual machine's (VM) resource access through register transfer level (RTL) simulation and implementation on a field-programmable gate array (FPGA), and it was only less than 5.6% compared with the resource access time by a single core processor.

## 1. はじめに

組込みシステムは対象アプリケーションが広がっており，その機能と複雑さは増加の一途をたどっている．従来の用途である車載制御，産業分野などのリアルタイム制御だけでなく，車載情報処理，マルティメディアなどの情報処理にも利用は広がっている．これらを組込みシステムの低コスト化の要求の元で実現するには，新たなソリューションが望まれる．このソリューションとして，異なるシステムを，マルチコア上に搭載するアプローチが有効である[1]．

- 制御システムのリアルタイム性の実現のため，処理時間の予測性に優れる AMP[2]構成をとる．制御システムと情報システムは別々のコアで動作する．
- リソースは低レイテンシと処理時間の予測性のため，極力，各システム専用とする．
- 共通に利用するメモリなどのリソースは，領域を各システム用に分割するが，低コスト化のため統合する．

†1 (株)日立製作所　中央研究所
Hitachi Ltd., Central Research Laboratory
†2 ルネサスエレクトロニクス㈱
Renesas Electronics Corporation
†3 東京農工大学　大学院　工学研究院
Tokyo University of Agriculture and Technology,
Graduate School of Engineering

このリソース統合により，各システムが独立に管理していたリソースが共通となり，システムの独立性は破れる．これにより，システムの障害が相互に影響しあい信頼性が低下しソフトの再利用も困難となる[1,3]．

この問題を解決するため，各システムの分離・保護が必要となり,PC サーバで流布している仮想化技術の利用が有効である．仮想化技術は，仮想ハード環境 VM(Virtual Machine)上で独立にシステムを構築する技術で，VM にリソースを割り当てるリソースパーティショニングと，実リソースと異なる機能を実現するエミュレーションからなる．ここでは同一機能のリソース割り当てを想定しており，リソースパーティショニングのみを対象とする．

しかし，仮想化技術はオーバヘッドを伴う．リアルタイム性の遵守が必要な制御システムでは，シングルコア並の性能が望まれる．従来，仮想化の高性能化に対しては，ハードウェア支援機構が開発されているが[4-8],このような厳しい条件では課題は残されていると考えている．

そこで，我々は，組込みシステム向けのハードウェア支援技術 ExVisor/XVS を開発した．その特徴は，オーバヘッドの少ないシンプルなハード構成にあり，リソースアクセスを支援する PAM(Physical Address Management Module)と割り込みの分配機構が主要技術である．ここで，本技術

Vol.2013-SLDM-160 No.28
Vol.2013-EMB-28 No.28
2013/3/14

はフル仮想化技術にカテゴライズされ，従来我々が開発した準仮想化技術[1]をベースに開発した．PAM のハードウェア容量低減も新たな論点となっている．

以下では，第 2 章で背景と関連研究を述べ，第 3 章と第 4 章で ExVisor/XVS 技術と評価結果を示す．最後に第 5 章で結言を述べる．

## 2.背景と関連研究

### 2.1 マルチコアシステムの構成と課題

第 1 章で述べたように，低コストとリアルタイム性を重視する組込みシステムでのマルチコアの適用例として，車載用システム[1]の例を図 1 に示す．

CPU core は二つあり AMP[2]構成をとる．core0 は制御システム用で core1 は情報システム用である．リソースは，極力各システム専用に割振る．図 1 では MemC，DU，PCIex が core1 専用に割振られる．両 core が利用するリソースは共通リソースとして統合し領域やチャネルを分割する．ROM，SRAM，INTC，Timer，DMAC がこれにあたる．

このリソース統合により，システムの独立性が破れる．図２に示すように，元々，各 CPU が独立なメモリと I/O にアクセスし，独立に割り込み要求を受けていたが，メモリと I/O の統合により独立性が阻害される．

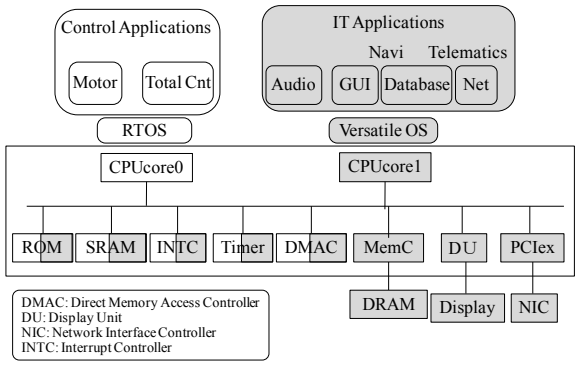


図 1 AMP’s example

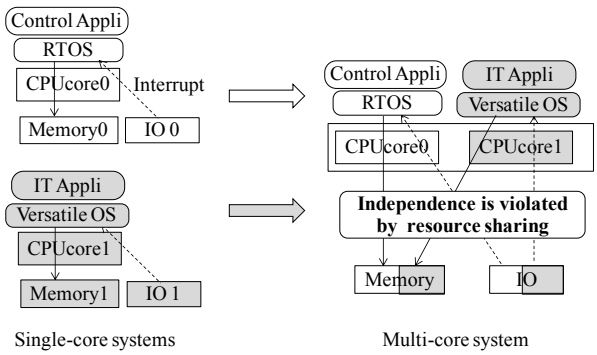


図 2 AMP’s issue

この独立性の阻害により，システムの信頼性が低下しソフトの再利用も困難となる [1,3]．一方のシステムの障害で他方のシステムのリソースを書き換えると，他方へも障害を引き起こす．基盤ソフトは統合前のアドレスでは再利用できなくなり，システム統合の都度改造の必要が生じる．

### 2.2 仮想化技術の適用と課題

（１）物理パーティショニング

前節で述べた課題を解決するためには，第 1 章で述べたリソースパーティションニングが必要となり，低レイテンシ化のため物理パーティショニングを採用する．

VM 上で図 1 の各 OS を実行し統合リソースを独立した仮想リソースに見せる．物理パーティショニングは，VM と実リソースを仲介する VMM(VM Monitor)レイヤで実現する．

表 1 Functions of physical partitioning

| No. | Case | Functions |
|---|---|---|
| 1 | Resource access from VM | Physical address management<br>・address translation<br>from VM’s physical address to real physical address<br>・VM’s access privilege check for this real physical address |
| 2 | Interrupt request from I/O | Partitioning of I/O’s interruptions to correspondent VM which includes this I/O or to VMM |

物理パーティショニングは，VM のリソースアクセスと I/O の VM への割り込要求に対して，表１に示す物理アドレス管理と割り込み分配が必要となる．関連デバイスとして，リソースアクセスでは，CPU core，I/O，DMAC，割り込み分配では，INTC と CPU core が挙げられる

（２）課題

課題は，リアルタイム性と低コスト化のため，シングルコアなみの低レイテンシを目指すオーバヘッド削減と，ハードウェア容量の低減である．

特に，物理アドレス管理が問題となる．これは MMU と類似機能であるが，MMU は TLB（Translation Lookaside Buffer）でミスヒット時に大きなオーバヘッドを生じるため，同様な実現手段を用いると問題となる．この問題を小容量のハードウェアで解決する必要がある．

### 2.3 従来技術

従来技術として，オーバヘッドを削減する仮想化のハードウェア支援技術[4-8]を述べる．いずれもフル仮想化技術にカテゴライズされる．

まず，CPU core 上で VMM の処理を VM が意識せず実行するため，VM より上位権限の VMM 用モードが必要となる．これに伴い，VMM 遷移時の状態退避レジスタなどの各種レジスタと VMM 用命令が装備されている．

次に物理アドレス管理について述べる．

CPU core からのアクセス時は，VM 上の MMU で論理アドレスから仮想の物理アドレスへ変換され，仮想の物理アドレスは VMM 用のページテーブルで実物理アドレスに変換される．ページテーブルは複数階層からなり，テーブルウォークはハードで実現する．この変換の過程で実物理アドレスへの権限チェックも行う．TLB はこの 2 段階のアドレス変換のキャッシュで，VM と VMM の区分用タグを装備している．

Vol.2013-SLDM-160 No.28
Vol.2013-EMB-28 No.28
2013/3/14

物理アドレスのビット数は 40 ビット以上で，ビット数の違いでテーブルの階層数は 3 から 5 と異なる．ページサイズは高々64KB であり，テーブル容量の削減には複数階層のページテーブルが必要となる．これらの技術は TLB キャッシュミスヒット時の VMM モード遷移を含むハンドリングと複数のテーブルアクセスがオーバヘッドとなる．

I/O からのアクセス時は，DMA からの出力となる仮想の物理アドレスを入力して CPU core と同様な機構により物理アドレス管理を行う．X86 系では PCI バスの出口で一括して管理し，ARM 社仕様では DMAC の出口で管理する．オーバヘッド要因は CPU core の場合と同様である．

最後に，割り込み要求の VM への分配は，割り込み要求を VMM で受けたあと，ソフトウェアで対象の VM の割り込みルーチンを起動することにより行われる．

以上の従来技術は，物理アドレス管理でのキャッシュミスヒット時のハンドリングと複数階層のページテーブルへのメモリアクセス，および割り込み分配の VMM 上ソフト実行が，オーバヘッドとなっている．

## 3. ExVisor/XVS 技術

これまで述べたように，ExVisor/XVS はマルチコア向けのリソース保護のため，物理パーティショニングを支援するハードウェア技術であり，その機能は物理アドレス管理と割り込みの分配である．課題は，オーバヘッドを小容量のハードウェアで削減することにある．

以下では，VMM 用モード，物理アドレス管理モジュール PAM，および割り込みの分配機構を述べる．

### 3.1 VMM 用モード

VMM 処理のため，従来と同様に，特権より上位権限の VMM 用モードとして XVS モードを設け，XVS モード用のレジスタと命令を装備する．

XVS モードでは，VMM 用リソースの初期設定と例外処理の受付を行う．XVS モードにより，VM 上の OS は，従来のシングルコア上の OS がそのまま再利用できる．μITRON などの RTOS も LINUX などの汎用 OS も再利用できる．

### 3.2 物理アドレス管理モジュール PAM

#### 3.2.1 機能と課題

PAM は，第 2 章の表 1 で示した物理アドレス管理を行う．従来は，MMU と同様な実現手段を採っていたため TLB のミスヒット時に大きなオーバヘッドを生む．PAM の課題は，この問題を小容量のハードウェアで解決することにある．

#### 3.2.2 基本思想

前節の課題に対して，PAM は，組込みシステムのメモリ利用方法の特質を生かして，階層のないページテーブルをレジスタで実装し，このテーブルによるダイレクトなアドレス変換でキャッシュ同等の性能を達成する．

一般にはこの方法を採ると多大なレジスタが必要となる．例えば 32 ビットアドレス領域に対して 64KB のページサイズでは 64K のページ数が必要となる．テーブルエントリを 16B 取ると 32bit レジスタが 9.6 万個以上必要となる．この問題を以下のように解決した．

従来の物理アドレス管理では，ページテーブルは複数階層となっている．アドレス領域が広くページサイズが小さくても，小容量でページテーブルを実現するためである．

逆に言うと，対象のアドレス領域が狭いか，ページサイズを大きくできれば，階層のないページテーブルを実現できる．組み込みシステムではいずれかが可能である．

アドレス領域を狭くすることは制御システムなどでは可能となる．高々数 10MB しかメモリを利用しないケースでは，ページサイズを 64KB に設定してもページ数は 256 個以下で，階層なしのページテーブルをレジスタで実現できる．

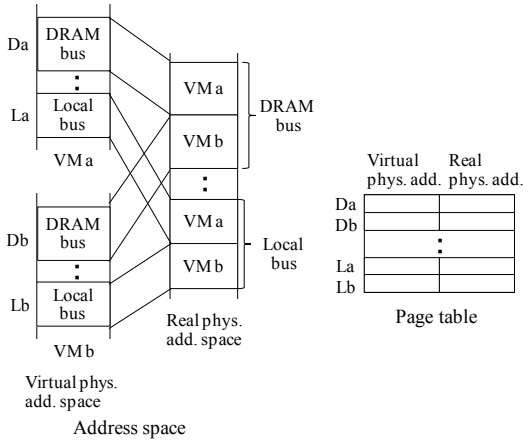


図 4　Large page size

ページサイズを大きくすることは，アドレス領域が広い情報システムで必要となる．組込みシステムでは全領域をメモリに常駐するため，スワッピングによる性能低下がなくページサイズを大きくとることができる．図 4 に示すように，バス領域を VM ごとに分割して，一つの VM 領域を一つのページにするような大きなページサイズにする．16MB のページサイズでは，4GB のアドレス領域に対してもページテーブルのエントリは 256 個以下となり，階層のないページテーブルをレジスタで実現できる．

以上の考え方に基づき，PAM は階層のないページテーブルによる低レイテンシ化を小容量のレジスタで実現する．

#### 3.2.3 仕様

（1）位置づけ

PAM は，図 5 に示すように，CPU core, I/O, DMAC からのリソースアクセスを独立なモジュールとして一括して受ける．これにより関連デバイスの改造は最低限となる．リソースはバスに接続されるため，PAM はバス単位に装備する．

CPU core からのアクセス時は，図 6 に示すように，VM 上の MMU と CPU core 外部の PAM による 2 段階のアドレス管理となる．TLB は VM 専用のアドレス変換キャッシュとなり，VMM も利用するためエントリーをタグにより区別する．

Vol.2013-SLDM-160 No.28
Vol.2013-EMB-28 No.28
2013/3/14

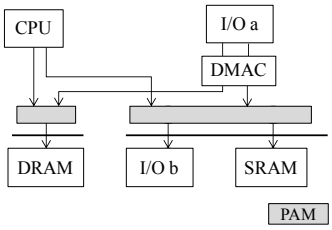


図 5 PAM's position

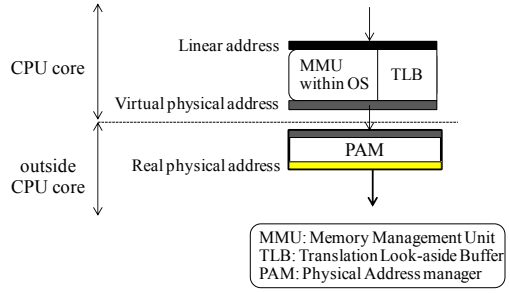


図 6 CPU's Physical address management

（2）実現方式

PAM の実現方式を図 7 に示す.

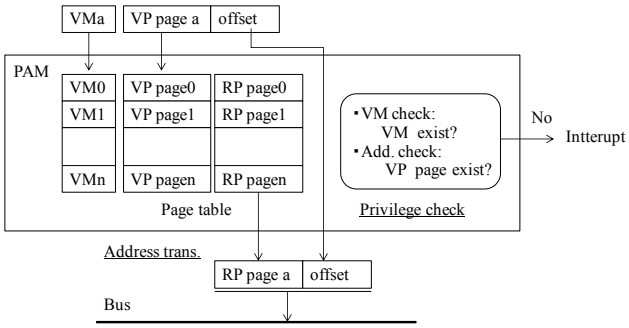


図 7 PAM's specification

アドレス変換は，仮想物理と実物理のページをペアにするテーブルをレジスタで実現する．対象バスに対するアクセス権限を持つ VM を，アクセス権限を持つページペア VP page と RP page と共に，あらかじめ登録しておく．

アクセスを行う VM が，その ID VMa と仮想物理のページ VP page a を入力すると，ページテーブルが実物理のページ RP page a に変換する．RP page a と offset で実物理アドレスとなり対象バスを介してリソースにアクセスする．

以上から,PAM の入力には VM の ID が必要となり,CPU core と DMAC は，PAM へは VM の ID も送付する．

VM のアクセス権限は，VM の ID と実物理ページ RP page に対応する仮想物理ページ VP page が登録されているかどうかでチェックする．VM の ID がなければ，対象バスに対してアクセス権限がなく，対応する仮想物理ページがなければ RP page へのアクセス権限がない．いずれかがなければ，エラーとして VMM へ割り込みを発生する．

（3）プログラミングモデル

PAM を利用するには三つの手順が必要となる．

まず，PAM の対象バスごとに実物理アドレスの割付を行う．第 3.2.2 節で述べた二つのメモリ利用方法によって割付方法が異なる．

狭いアドレス領域しか利用しない時には，ページサイズを小さくして割付に自由度を持たせることができる．リソースごとに連続領域を取りこれを VM に分割する自然な割付が可能となる．

広いアドレス領域を利用する時には，ページサイズを大きくとる．図 4 に示すように，VM ごとに連続領域をとりこれを少数のページに分割する.リソース領域は VM 領域内で分割する.同一リソースが異なる VM に渡る時は飛び領域になることもある．

次に，割付結果，ページテーブルのページサイズとエントリ数が決まるので，レジスタで実装する．同時に，実物理ページに対して VM の仮想物理ページを割り当てる．

最後に，VMM 上で動作する，ページテーブルの初期設定とアクセス権限エラー用割り込みのプログラムを開発する．

（4）ハード容量削減用の拡張 PAM

これまで述べた PAM のハード容量削減はアドレスの割付の工夫による，運用面の努力が前提となる．

運用面の工夫に加えて，PAM を搭載するバスの種別に照らして，ページテーブルの要素を削減する拡張 PAM を開発した.（2）で述べた PAM（full-PAM）と併せて表 2 に示す．表 2 のレジスタ数は，第 3.2.4 節で後述する．

表 2 PAM's variation

| No. | PAM type | Suitable bus type | Check access privilege | | Address trans-lation | Number of 32 bit registers |
|---|---|---|---|---|---|---|
| | | | VM/VM M check | Address check | | |
| 1 | full-PAM | Common-use | yes | yes | yes | 5/page |
| 2 | fa*1-PAM | Special-purpose | yes | yes | no | 4/page |
| 3 | vs*2-PAM | Exclusive-use | yes | no | no | 2/PAM |

*1 fixed address , *2 VM specific

No.1 の Common-use バスは VM 間で共通に使い，アドレスもシステム統合により変化し，full-PAM が必要となる．DRAM 用のバスがこれにあたる．

No.2 の Special purpose バスは複数の VM が利用するが，基本機能や特殊機能のモジュールを接続するため，アドレスは固定の予約領域として設定できる．アドレス変換が不要な fa-PAM を利用でき，実物理ページ RP page は不要となる．INTC を接続するバスがこれにあたる．

ここで，fa-PAM はリソースに固定のアドレスを取るため，VM に自由に実物理アドレス領域を割り当てることはできず，VM ごとに領域を取るアドレス割付には採用できない．

No.3 の Exclusive use バスは特定の VM しか利用しないため，該当 VM か否かの権限チェックのみでよく，VM ID このみを搭載する vs-PAM を利用できる．情報システム用 VM のみがアクセスする PCI Express バスがこれにあたる．

### 3.2.4 実装例

第 3.2.3 節の仕様に従って実装した例を述べる．

図 8 に示すように，第 2.2 節のマルチコアシステムでは，PAM は，Internal System Bus に接続する各バスに配置する．

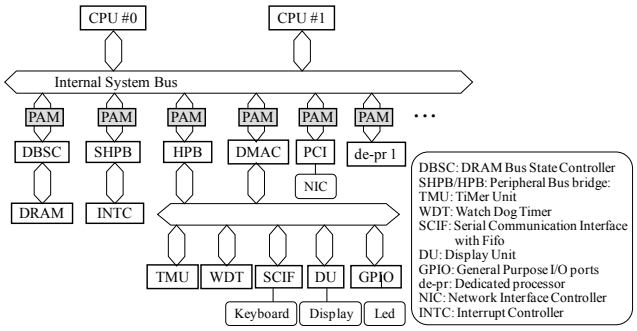


図 8　Example of PAM's placement

PAM の内部構成の実装例を図 9 に示す．VM の ID，仮想物理のページ Page V と実物理のページ Page R をペアで登録する．VM のマスクは複数の VM に共通のペアに対して利用する．ページマスクは，例えば，利用しないアドレスビットの指定を割愛するために利用する．

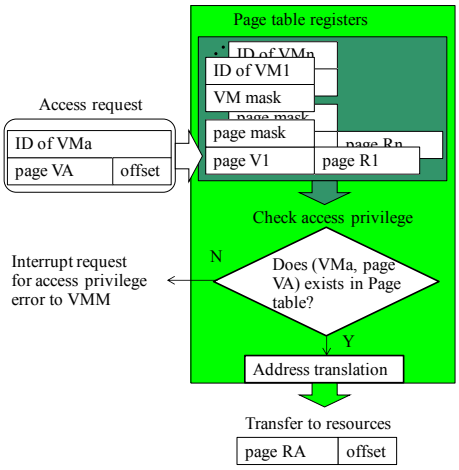


図 9　PAM's structure

VMa から Page VA でアクセスする場合，ページテーブルレジスタから VMa と Page VA を含むエントリを検索する．エントリがあれば対応する Page RA を offset と共にバスに転送してリソースへアクセスする．対応するエントリがないとアクセス権限エラーとして VMM へ割り込みをかける．

最後に，第 3.2.3 で述べた 3 種類の PAM のレジスタ数を述べ，各 PAM を図 8 のバスに当てはめる．full-PAM は図 9 に示すようにページあたりレジスタ数は 5 個で，DRAM 用バス DBS や各種 I/O が繋がる HPB に利用する．fa-PAM は Page R が不要でレジスタ数は 4 個となり，INTC や CPG を接続する SHPB や Video アクセラレータを接続する ICB に利用する．vs-PAM は，PAM 一個あたり VM に関するレジスタ 2 個のみでよく，DMAC や PCI 用のバスに利用する．

### 3.3　割り込み分配機構

割り込みは I/O が VM に要求する．本研究の想定では 1CPU core に 1 VM のみを搭載するため，この CPU core がわかれば割り込みは VM に届けられる．しかし，割り込みは PAM が VMM へも要求するため VM/VMM の区分も必要となる．

これらの区分をハード的に行い，VM/VMM に自動的に割り込みを分配する．INTC 内に割り込み要求ごとに CPU core と VM/VMM の ID を格納しておき，これらの識別信号を発行すると CPU core で対応する割り込みベクタに遷移する．

## 4. 評価結果

PAM と割り込み分配のオーバヘッドを評価し，PAM は第 3.2.2 節で述べた 2 種類のアドレス割付とページサイズに対して，アドレス領域に依存するハードウェア容量としてレジスタ数を評価した． オーバヘッドは，ExVisor/XVS をルネサス社製 SH4-A のマルチコアと RTL 論理で試作し RTL シミュレーションと FPGA 実装の結果から評価した．論理合成では最大 648MHz の周波数での動作を確認している．

### 4.1　PAM のオーバヘッド

メモリアクセスに対するオーバヘッドを二つの方法で評価した．

評価 1 は，シミュレーションで検証した基本ステップのオーバヘッドサイクルを机上で足し合わせ，実シングルコアでのメモリアクセスサイクルに対する割合を算出した．結果は表 3 に示す通りである．

CPU core からのアクセスでは，TLB での VM ID のタグつきの実行と PAM にオーバヘッドが発生する可能性がある．後者が高々1 サイクルのオーバヘッドで，メモリアクセス全体に対して 5.6%以下のオーバヘッドとなった．I/O からのアクセスでも PAM でオーバヘッドがある．DMAC 利用のユースケースは，制御システムでは 1msec 程度の周期でセンシングデータ 1 個の ADC による取り込みを想定し，情報システムでは AV データ 10 個以上のバースト転送を想定した．共に 3.3%以下のオーバヘッドであった．参考までに，従来の物理アドレス管理方法では，TLB ミスヒット時に，最小 2 階層のテーブルで最低 2 回メモリアクセスが生じる．ミスヒットハンドリングを無視しても CPU core アクセスで 70%以上のオーバヘッドとなる．

表 3　Evaluation of overhead 1

| No. | Type of access | | Overhead | | |
|---|---|---|---|---|---|
| | | | Process | Number of cycles | Percentage to total cycles |
| 1 | Resource Access from VM | CPU core | TLB execution with ID tag of VM/VMM | 0 | 0-5.6% *1 |
| 2 | | | Physical address management by PAM | 0-1 bus cycle | |
| 3 | | IO | Transfer ID of VM/VMM | 0 | 0-3.3% *2 |
| 4 | | | Physical address management by PAM | Same with No.2<br>0-1 bus cycle | |
| 5 | Interrupt request from I/O | | Select target VM/VMM and transfer relevant signal | 0 | 0 |

*1 assume original total access consists of a few CPU cycles plus resource access cycles with more than 15 bus cycles, *2 original cycles are 2× n data transfer with more than 15 bus cycles plus 1 cycle of DMAC execution, n=1 for control system and n=10 for information system

Vol.2013-SLDM-160 No.28
Vol.2013-EMB-28 No.28
2013/3/14

評価 2 は PAM 実行の ON/OFF のケースに対して，CPU からのメモリの R/W を何度か行い比較した．データキャッシュオフの条件で,メモリアクセスを Loop の中に入れて実測したサイクル数と Loop のみで実測したサイクル数の差をアクセス回数で割った平均のアクセスサイクルを導出した．100 回程度のアクセスではバスコンテンションなどのゆらぎでオーバヘッドがマイナスになることもあるが，10000 回で相殺されオーバヘッドはほぼ 0 になった.PAM の ON/OFF 共に，リードで 10，ライトで 23 サイクルであった．

次に，割り込み分配のオーバヘッドを基本ステップの実行で評価した.表 3 の No.5 に示す通りオーバヘッドサイクルは 0 であった．

### 4.2 PAM のハードウェア容量

第 3.2.2 節で述べた 2 ケースのレジスタ数を評価した．

ケース 1 は，アドレス領域が狭い場合でページサイズは高々64KB である．試作はこのケースで行った．表 5 に示すように，約 1.0GB の領域に対して，レジスタ数は 1400 個弱であった．同じ PAM 構成でストリーミングの画像認識のデモ開発も行えた[9]．

表 5 PAM’ usage for small address space

| No. | Bus type | Bus | PAM-type | Address space | Page size | No. of registers |
|---|---|---|---|---|---|---|
| 1 | Common-use | DBSC<br>LBSC<br>HPB | full-PAM | 4MB<br>32KB<br>4MB | 64 KB<br>4 KB<br>4 KB | 320<br>40<br>320 |
| 2 | Special-purpose | SHPB<br>ICB<br>Local Mem | fa-PAM | 4KB<br>4MB<br>6MB | 64 B<br>64 KB<br>4 KB | 288<br>16<br>384 |
| 3 | Exclusive-use | DMAC<br>PCI, etc. | vs-PAM | 1.02GB | - | 22 |
| | | | Total | 1.024GB | - | 1390 |

ケース 2 は， 32 ビットアドレス領域を対象に，ページサイズを大きくして実現した．表 6 に示すように，ハード固有領域や使われていない領域以外の全領域，約 3.6GB を PAM で管理した．各バスの領域は 8 個以上の VM を管理できるようにページサイズを設定した.領域が広い DBSC と LBSC は 16MB ものページサイズを設定した．レジスタ数は 1500 個弱であり実用性を確認した．

表 6　PAM’s usage for large address space

| No. | Bus type | Bus | PAM-type | Address space (nearly 32bit total space) | Page size | No. of registers |
|---|---|---|---|---|---|---|
| 1 | Common-use | DBSC<br>LBSC<br>HPB | full-PAM | 2.2GB<br>194MB<br>4MB | 16MB<br>16MB<br>256KB | 710<br>65<br>80 |
| 2 | Special-purpose | SHPB<br>ICB<br>Local Mem | | 4MB<br>6MB<br>146MB | 256KB<br>256KB<br>4MB | 80<br>120<br>380 |
| 3 | Exclusive-use | DMAC<br>PCI, etc. | vs-PAM | 1.02GB | - | 22 |
| | | | Total | 3.57GB | - | 1477 |

## 5. おわりに

我々はリアルタイム性が重視される組込みマルチコア向けに，リソースパーティショニングのオーバヘッド削減を目的として，ハードウェア支援技術 ExVisor/XVS を開発した．主要技術は，物理アドレス管理モジュール PAM と割り込み分配機構である．

特徴技術である PAM は，組込みシステムのメモリの利用方法の特徴を活かして，階層がないページテーブルをレジスタで実現し，ダイレクトなアドレス変換でキャッシュなみの高速化を図った．

ExVisor/XVS をマルチコアと共に最大 648MHz の周波数で RTL 論理を試作し，RTL シミュレーションと FPGA 実装で評価した．割り込みのオーバヘッドは 0 に抑えられ，リソースアクセス時の PAM のオーバヘッドは 5.6%以下であることを確認した．

PAM のハードウェア容量は，狭いアドレス領域の場合と，32 ビット空間全てを利用するが大きなページサイズを採用する場合の双方で評価し，PAM のレジスタ数は高々1500 個で実用に耐えうることも確認した．

## 参考文献

[1] T. Nojiri et al., “Domain Partitioning Technology for Embedded Multi Core Processors”, IEEE Micro, vol. 29, issue 6, Nov./Dec. 2009.

[2] Hiroyuki Tomiyama, “Real-Time Operating Systems for Multicore LSI,” Tutorial at Asia and South Pacific Design Automation Conference (ASP-DAC), Yokoyama, Japan, Jan. 2009.

[3] M. Domeika, “Software Development for Embedded Multi-core Systems: A Practical Guide Using Embedded Intel Architecture”, Newnes, Apr. 2008.

[4] D. Abramson, et al., “Intel Virtualization Technology for Directed I/O”, Intel Technology Journal, vol. 10, issue 3, 2006.

[5] R. Uhlig et al.,“Intel Virtualization Technology”, Computer, vol. 38, no. 5, pp. 48-56. May 2005,

[6] “AMD-V Nesting Paging”, 2008.

[7] D. Brash, “Extensions to the ARMv7-A Architecture”, Hotchips 22, Aug. 2010.

[8] J. Goodacre, “Hardware accelerated Virtualization in the ARM Cortex™ Processors”, XenSummit Asia, Nov. 2011.

[9] 杉本健他，「組み込み向けマルチコアプロセッサにおける複数 OS 実行環境の構築技術」，情報処理学会計算機アーキテクチャ研究会報告，Vol.2010-ARC-189，No.3，pp.1-8，2010 年 4 月．