Vol.2017-OS-141 No.10
2017/7/26

FPGAオフローディングを提供する組込みシステム向けVMMの設計

前田　剛志[1]　並木　美太郎[1]

概要：近年，組込みシステムへの要求が高度化しており，VMM を用いて情報システムとリアルタイム処理を同時に実行することが考えられている．そのような組込みシステムの高度化に伴い，ハードウェアの処理能力の高度化が求められており，Xilinx Zynq のような FPGA アクセラレーションを可能とする SoC も登場している．本研究では，組込み向け VMM 上にてゲスト OS の FPGA を利用したオフローディングを可能にすることを目標とする．FPGA コントロールの処理を VMM により隠蔽し，ゲスト OS への変更を加えることなくオフローディングを可能にすることを提案する．この提案を実現するため，VMM による FPGA アクセラレータ制御，ゲスト OS からのアクセスリクエストの処理，およびゲスト OS から FPGA オフローディングを利用するための API について設計を行う．

Designing VMM-Based Embedded System Providing FPGA Offloading

Maeda Takeshi[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 RTOS 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，仮想化技術を用いる実行基盤の研究 [1][2][3] 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．

また，システムの高度化に伴い，ハードウェアの処理能力の高度化が求められている．これに対する解決策の一つとして，プロセッサに加えてアクセラレータを利用することがある．サーバー産業では Graphical Processing Unit(GPU) や Field Programmable Gate Array(FPGA) をハードウェアアクセラレータとして利用することも増えてきている．そうした中で，ゲスト OS に対して FPGA アクセラレータの使用を提供するような研究 [5] も存在し，特定のアプリケーションに対する FPGA アクセラレータの使用により高速化を実現している．大手 FPGA ベンダも組込み向けプロセッサと FPGA を搭載した SoC を提供しており，Intel の Intel Soc FPGA や Xilinx の Zynq 等が代表的である．マイクロコントローラのみでは実現が困難だったソフトウェアもアクセラレータを利用することで現実的になっている．

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

そこで本研究では，特定のアプリケーションの実行が行われる組込みシステムにおいて，上記 VMM に FPGA アクセラレータによるオフローディングをゲスト OS に対して提供することを考える．特定の処理が行われる組込みシステムにおいて，この構成は有効に働くことが予想される．

本研究では,CPU，メモリ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および，MVM のゲスト OS への FPGA オフロード機能の提供についての設計を行った．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 章に述べる．3 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．4 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基本設計について述べ，5 章にて MVM のゲスト OS への FPGA オフロード機能の提供についての設計について述べる．最後に 6 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，第 1 章で触れたように，組込みシステムにてアクセラレータとして FPGA を利用したいという需要が高まっている．そこで，本研究にて開発する VMM がゲスト OS に対し FPGA オフローディングする機能を提供することを考える．その際の課題としては，FPGA オフローディングの利用に，プログラマが VMM や他のゲスト OS の存在を考慮する必要を与えないということがある．そのため，VMM により FPGA コントロールを隠蔽し，管理する必要がある．

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．また，VMM がゲスト OS に FPGA オフローディング機能を提供することで，アプリケーションの高速化を支援することを考えている．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 への FPGA オフローディング機能の提供**

これらの課題を解決するため，以降の章にて基盤システムの設計を行う．

3. ARM の仮想化機能

この章では使用するプロセッサの紹介と，利用する仮想化機能についての説明を行う．

3.1 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．

3.2 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非ワールド

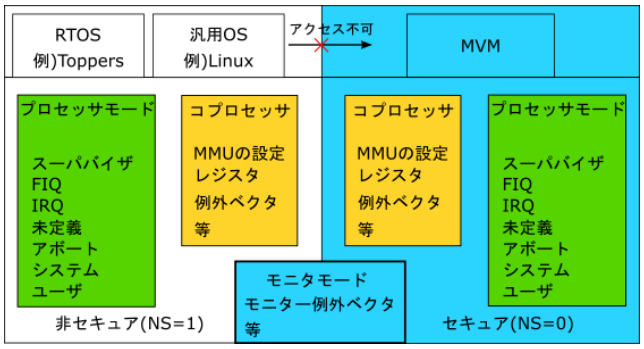

図 1　MVM の概要図

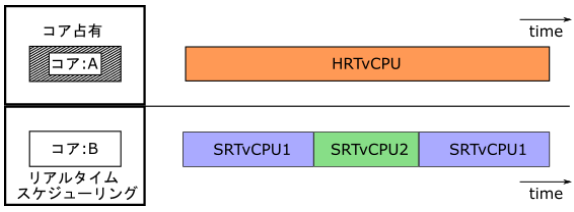

図 2　CPU 仮想化の図

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，プロセッサが非セキュア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セキュアなハードウェア資源に対してはプロセッサがセキュアな状態であるときにのみアクセスが可能である．また，プロセッサのセキュア，非セキュア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，非セキュアワールドにて汎用 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[4] 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．

4. 仮想マシンモニタ MVM の設計

この章では，仮想マシンモニタ MVM の設計について述べる．MVM は Type1 仮想マシンモニタであり，セキュアに MVM を，非セキュアにゲスト OS を配置する．このような配置にした理由は，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，セキュア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好ましいと考えたためである．

4.1 全体概要

MVM は高度な組込みシステムの実行基盤を提供するために，基本として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，MVM 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，MVM 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．

MVM 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 MVM 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．

4.2 リアルタイムレベル

MVM 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．

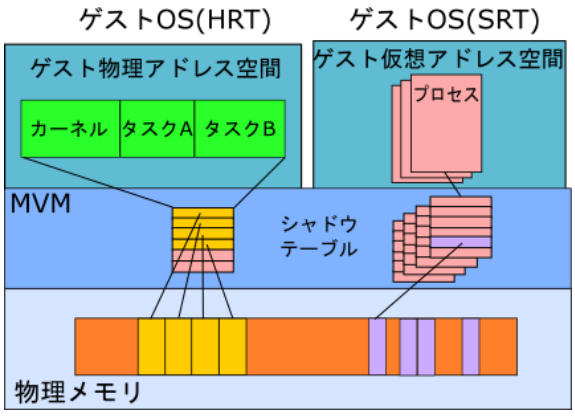

図 3　メモリ仮想化の図

4.3　CPU 仮想化手法

CPU を仮想化し，よりリアルタイム性の要求されるものに対し CPU の占有率を高めることでリアルタイム要求の異なるゲスト OS を動作させる．リアルタイムレベルの考えに則り，以下の二つの手法を提供する．概要は図 2 のようになる．

4.3.1　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，MVM 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．

4.3.2　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，EDF スケジューリングを行う．EDF スケジューリングに則り占有されていない CPU を仮想化することで，優先度の高いゲスト OS が優先実行される．

4.4　メモリ仮想化手法

メモリ仮想化に関しても，CPU の仮想化と同様に二つの手法を提供することにより，異なるリアルタイム要求のゲスト OS に対応する．概要を図 3 に示す．

4.4.1　ダイレクトマップ

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 MVM 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．

4.4.2　シャドウページング

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．

4.5　I/O について

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，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．

4.6　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．

4.7　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

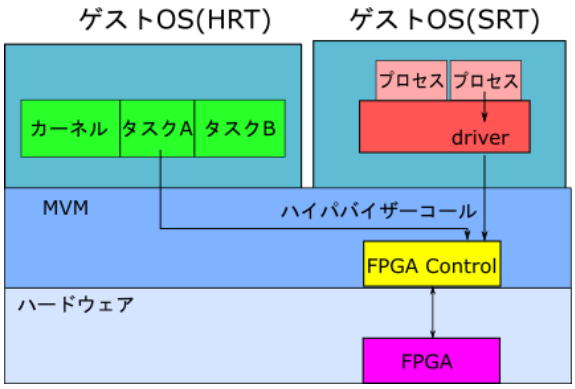

図 4　FPGA オフローディング機能の呼び出し

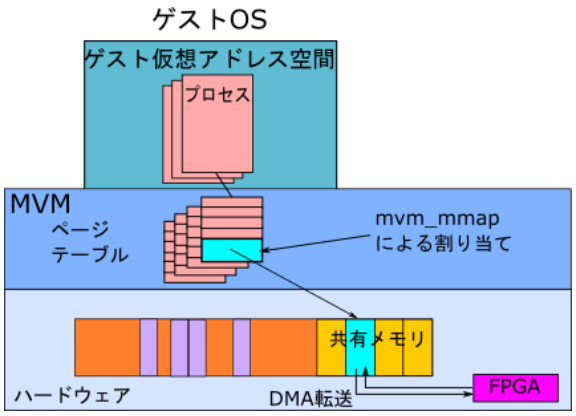

図 5　mvm_mmap と共有メモリ

機能である．このハイパバイザーコールの機能を用いて各種仮想化機能や後述の FPGA オフローディング機能の呼び出しを実現する．

5. FPGA オフローディング機能の設計

本章では，MVM の FPGA オフローディング機能の実現に向けた設計を行う．まず全体構成について述べ，次に API，そして MVM での FPGA コントロール部について述べる．

5.1 全体構成

まず前提として，FPGA は単一の機能を持つこととし，複数機能や，リコンフィギュレーション等については考えないものとする．将来的には複数機能やリコンフィギュレーションにも対応したいが，現段階では簡単のため，単一機能に絞って設計を行う．またデータの転送は Direct Memory Access(DMA) にて行うものとする．

MVM が管理する機能としては，共有メモリの管理，FPGA へのデータの転送，FPGA からアプリケーションへの演算結果の返却，FPGA オフローディングリクエストのスケジューリングである．

FPGA を直接操作するのは MVM で，FPGA をコントロールするデバイスドライバは MVM が持つ．そのため，ゲスト OS が MVM の提供する FPGA オフローディング機能を利用するにあたって，その機能の呼び出しはハイパバイザーコールによって呼び出す．機能の呼び出しのフローは図 5 のようになる．

5.2 API

この節ではゲスト OS から発行する API についての設計を述べる．以下の 2 つの API を用意する．

5.2.1 mvm_mmap

データの転送を実現するため，MVM の管理する共有メモリをゲスト OS 空間にマップする要求を送る．この取得したメモリ空間に対し，転送するデータの準備，演算結果の返却が行われる．引数としてサイズを指定し，戻り値としてメモリアドレスを返却する．

5.2.2 mvm_request

MVM に FPGA オフローディングの実行リクエストを送る．引数として転送するデータのソースアドレスとデータサイズ，返却される演算結果のディスティネーションアドレスを指定する．

この API の実行により，実行したプロセスもしくはタスクは待機状態になり，結果の返却を待つ．

5.3 FPGA コントロール部

この節では MVM による FPGA のコントロール部についての設計を示す．コントロール部は主に共有メモリの管理，データの転送，演算結果の返却，FPGA オフローディングリクエストのスケジューリングからなる．

5.3.1 共有メモリの管理

この項では MVM の共有メモリの管理について述べる．共有メモリは FPGA へのデータの転送，返却のために使われる．共有メモリを利用することで，ゲスト OS から MVM へのデータの転送の処理が省けるという点，物理的に連続なアドレス空間が利用できるという利点がある．共有メモリにはラージページを用いることでデータサイズが大きい場合にもアドレス変換が少なくなるようにする．

共有メモリの要求が来た際，MVM は管理するゲスト OS のページテーブルに必要な共有メモリをマップする．

5.3.2 FPGA へのデータの転送，演算結果の返却

データの転送はリクエストが発行され次第指定されたソースアドレスとサイズをもとに DMA にて行われる．演算結果の返却はディスティネーションアドレスに格納され，転送が終わり次第ゲスト OS に割り込みを発行し，プロセスもしくはタスクを起床する．

5.3.3 FPGA オフローディングリクエストのスケジューリング

この項では，FPGA オフローディングリクエストが複数来た場合の処理について述べる．FPGA にて演算中に新たなリクエストが来た場合，そのリクエストは MVM の管理

するキューに挿入され，FPGA の処理が終了するまで待つ．処理される順番は FCFS(First Come First Served) とし，処理が終了したリクエストはキューから外され，キューの先頭のリクエストが実行される．キューの処理はリクエストが来た際と FPGA の処理が終了した際である．この仕組みにより，複数ゲスト OS からの FPGA オフローディング機能の要求に対応する．

6. 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，およびゲスト OS に対して FPGA オフローディング機能を提供する設計について述べた．

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，ゲスト OS が FPGA オフローディング機能を利用するために，MVM 側での処理と，ゲスト OS から利用するための API について提案した．

今後の課題は，設計した仮想マシンモニタ MVM 上に FPGA オフローディング機能を実装し，オーバヘッドの調査等評価を行う必要がある．

参考文献

[1] ”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.

[2] ”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.

[3] ”Making a Virtual Machine Monitor Interruptible”, Ito Megumi, Oikawa Shuich, Information and Media Technologies 6(4), pp.1139-1148, 2011

[4] ”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.

[5] Wei Wang, Miodrag Bolic, and Jonathan Parri. 2013. pvFPGA: accessing an FPGA-based hardware accelerator in a paravirtualized environment. In Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS ’13). IEEE Press, Piscataway, NJ, USA, , Article 10 , 9 pages.

[6] Muhammad Bilal Anwer, Ankur Nayak, Nick Feamster, and Ling Liu. 2010. Network I/O fairness in virtual machines. In Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures (VISA ’10). ACM, New York, NY, USA, 73-80. DOI=http://dx.doi.org/10.1145/1851399.1851412

[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環境における組込みシステム向け VMM の研究”, 前田剛志，小柴篤志，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）2016-OS-138, 13, pp.1 - 7, 2016-08-01