# 組込み仮想化のための高機能割込みコントローラ

請園 智玲[1,a)]

**概要**：外部割込みが発生した際に，組込み仮想化を実現するハイパーバイザは代表して外部割込みの受付を行う．その後，ハイパーバイザは管理する VM 内の適切な割込みハンドラを決定し，間接的にそのハンドラの呼び出しを行う．これを行うことで，ハイパーバイザは単純な割込みコントローラを用いて，複数の VM 間の割込み処理の調停を行っている．しかしながら，仮想化上での割込み処理のフローは，ソフトウェアオーバヘッドをもつ．それ故に，システムが高い割込み応答性が要求する場合，ハイパーバイザは組込みシステム開発において仮想化の導入を妨げる．本研究はこのオーバーヘッドに着目し，仮想化を導入することを前提とした SoC 設計において，仮想化対応の高機能の割込みコントローラを提案する．この割込みコントローラを利用することでソフトウェアオーバーヘッドを削減することが可能となる．

**キーワード**：割込みコントローラ，組込み仮想化

# A Sophisticated Interrupt Controller for Embedded Virtualizations

Tomoaki Ukezono[1,a)]

***Abstract:*** When an external interrupt occurs, hypervisors which are required for embedded virtualizations receive external interrupts as a representative of VMs. And then, the hypervisors decide the appropriate interrupt handler inside VMs, and call the handler indirectly. Therefore, hypervisors can arbitrate interrupt handlings between multiple VMs using the conventional interrupt controller. However, the flow of interrupt handling on virtualization has software overheads. Consequently, if systems are required to high responsibility for the interrupt, hypervisor prevent introducing embedded system development to virtualizations. In this paper, we focus on the software overheads of interrupt handling on hypervisors, and propose sophisticated interrupt controller which is compatible with virtualization. By using the interrupt controller, it is possible to eliminate the software overheads.

***Keywords:*** Embedded Virtualizations, Interrupt Controller

## 1. はじめに

### 1.1 組込み仮想化と先行研究

組込み仮想化を用いることで，ソフトウェアポータビリティの向上，堅牢性の向上，信頼性の向上，セキュアシステムの構築といった仮想化ならではの恩恵を組込みシステムに与えることができる．その一方で，組込み仮想化はソフトウェアで提供されるハイパーバイザ [1][2][3][4] をシステムに組込むことが必須であることから，ソフトウェアオーバヘッドが避けれない問題となる．

我々は先行研究 [5] で，組込み仮想化の I/O ポートアドレス参照時のハイパーバイザによるソフトウェアオーバヘッドに関して議論し，SoC の機能部品の一つとして，ハイパーバイザの機能の一部である I/O ポートアドレス変換のための付加的なハードウェアを提案し，その効果を示した．提案した I/O ポートアドレス変換ハードウェアは仮想化が前提である組込み SoC の設計において，I/O ポートアドレスアドレス変換に関係する仮想化のソフトウェアオーバヘッドを除去することができる．

### 1.2 ペリフェラルとの通信と割込み

図 1 に I/O ペリフェラルとの通信の概要を示す．図 1 の上部の図は I/O ペリフェラルへのデータの送信を示しており，下部の図はデータの受信を示している．I/O ペリフェラルへのデータの送受信は物理アドレスマップされた I/O ポートアドレスを用いる．データの送信は CPU が I/O ポートアドレスにマップされたデバイスレジスタに書き込みを行うことで可能である．一方，一般的なデータの

[1] 北陸先端科学技術大学院大学
JAIST, 1-1, Asahidai, Nomi, Ishikawa, Japan
[a)] t-ukezo@jaist.ac.jp

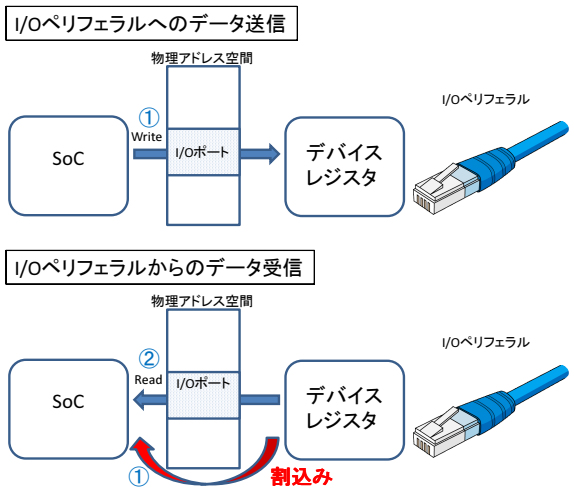


**図 1** I/O ペリフェラルとの通信の概要.

受信は，ペリフェラルがデータ送信の準備を完了した後に，割込みで CPU に合図を送り，その後，CPU が I/O ポートアドレスを介して，デバイスレジスタを読み込む事により行われる．

割込みを使用しないデータ受信の方法として，CPU が常に I/O ポートアドレスを読み出し続けてデバイスレジスタの値の変化を監視する方法 (ポーリング) があるが，その場合，ペリフェラルの処理中に CPU が他の処理が行えず，待機状態になる．このため，特殊な用途のシステムを除いてポーリングによるペリフェラル制御は現実的ではなく，割込みを用いることが一般的である．

また，図 1 で示したデータ授受の方法の他に，デバイスレジスタを介さずに，主記憶経由でデータを授受する方法 (DMA) が存在する．DMA は容量の大きい主記憶をバッファとするため，一度の割込みで大容量のデータ通信行えるが，CPU への割込み回数が削減する効果があることを除き，図 1 の方法と差異がなく，本質的には割込みを起点とするデータの送受信という点で同様である．加えて，DMA に用いるバッファ領域への主記憶参照に伴うハイパーバイザのアドレス変換は Intel VT-d[6] などのハードウェアが既に存在する．

### 1.3 ハイパーバイザの割込み制御

1.2 節で示したように，ペリフェラルとの通信では必ず割込みが使用されることから，仮想化環境では複数のゲスト OS(VM) の I/O 制御を調停するためにハイパーバイザが代表して割込みの制御を行う必要がある．図 2 に仮想化環境におけるハイパーバイザの割込み制御と提案手法の概要を示す．図はハードウェア (紺色矩形) 上にハイパーバイザ (水色の角取り矩形) が実行され，その上で 2 つの VM(緑色の角取り矩形) が実行されている 3 階層の図を示している．図では矩形をハードウェア，角取り矩形をソフトウェアとして表現している．また，赤い矢印は割込み処理のフローを，橙色の矢印はデータ通信のフローを表している．我々の先行研究であるハードウェアアドレスマップ変換のためのハードウェア [5] はペリフェラルとのデータ通信に必要な I/O ポートアドレスの変換を行うハイパーバイザのソフトウェア機能をハードウェア化した提案であるため，ハイパーバイザに食い込む形のハードウェアとして表した．

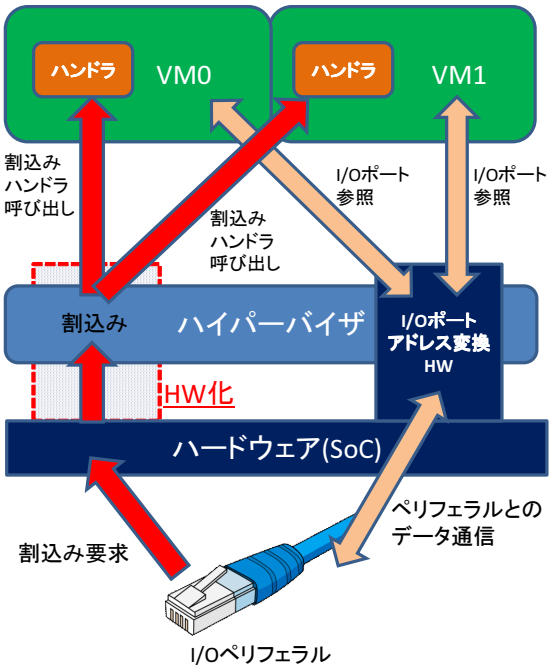


**図 2** ハイパーバイザの割込み制御と提案手法の概要.

ペリフェラルから割込み要求があった場合，非仮想化の状態であれば，CPU の割込み／例外処理ハードウェアにより OS が管理するトラップベクトルに強制的に実行が遷移し，OS の割込みハンドラが実行される．しかしながら，仮想化の場合は VM が複数存在しており，かつ，ハードウェア自身は仮想化で利用されていることを認識できないため，直接 VM 内のハンドラを呼び出すことができない．このため，ハイパーバイザが全ての割込み要求で代表して起動され，PIC(Programmable Interrupt Controller) に入力される割込み信号線により指定された割込み番号[*1]と現在の VM 稼動状態を判断の上，適切な VM のハンドラを呼び出すか，一旦マスク処理を行い当該割込みをペンディング状態にするかの判断を行う．当然ながら，これらの処理はソフトウェアで行われるため，ソフトウェアオーバヘッドが存在する．

本研究は上記のハイパーバイザ (ソフトウェア) の割込み処理をハードウェア化し，このハードウェアを SoC 設計時の選択可能な機能部品 (ソフトマクロ) として位置づけることを提案する．これにより，ハイパーバイザによるソ

[*1] 解析方法はシステムに実装された PIC の構成に依存する．例えば，PIC が Intel 8259[7] 互換を用いた実装であれば，マスタ・スレーブのカスケード構成により最大 64 の IRQ が提供され，APIC 実装の場合は IO APIC[8] のリダイレクションテーブルの解析が必要となる

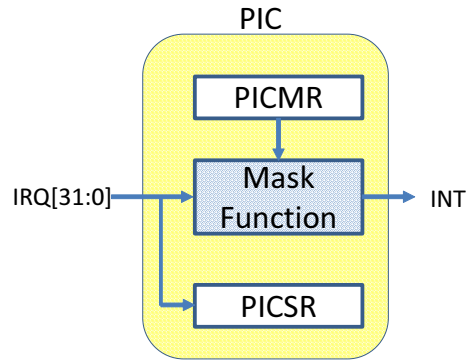


**図 3** OpenRISC における通常の PIC の構造.

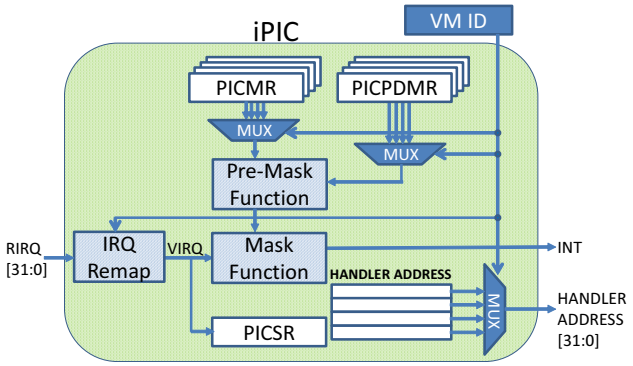


**図 4** 提案する iPIC の構造.

フトウェアオーバヘッドの削減を実現する．本研究では便宜上，このハードウェアを iPIC(intelligent Programmable Interrupt Controller) と呼ぶ．iPIC は仮想化を前提とした組込みシステムの SoC にのみ選択されて実装されるハードウェアである．

本研究は以下の章で構成される，本章は本研究の背景としてハイパーバイザの割込み制御のためのソフトウェアオーバヘッドについて議論し，提案手法の概要と目的を示した．次章では提案ハードウェアである iPIC の詳細について議論する．3 章では，提案手法を実機ベースで評価し，性能を議論する．4 章では，本研究の関連研究を紹介する．最後に 5 章で本研究をまとめる．

## 2. iPIC

### 2.1 ORPSoC における iPIC 実装の概要

PIC 制御は実装される組込みシステムのハードウェア構成により異なるため，本研究では例示として OpenRISC 命令セットアーキテクチャ [9] を採用した OR1200[10] プロセッ サをコアに持つ ORPSoC[11][12] が備える PIC を対象に議論する．図 3 に OpenRISC における通常 PIC の構造を示す．OpenRISC の PIC は Intel 8259[7] よりも単純な PIC の機能をもつ上に，シングルプロセッサ構成において必要十分な機能を備えている．また，当該 PIC は Intel 8259 と同様にエッジ及びレベル割込みトリガーモードをサポートしている．ORPSoC は 3 章の評価にも使用される．

当該 PIC は PICMR(PIC Mask Register)，PICSR(PIC Status Register) の 2 つのレジスタと Mask Function のロジックから構成される．PICMR は 32 ビットの特殊レジスタであり，ソフトウェアから 32 の割込みソースのマスク処理の指定をすることができる．PICSR は 32 ビットの特殊レジスタであり PIC に対するそれぞれの割込み入力を識別することができる．Mask Function は外部からの割込み信号 (IRQ: Interrupt ReQuest) と PICMR を入力として，論理積でマスク処理を行い，1 つでも有効なビットが有れば，割込み信号 (INT) を CPU に伝える．本研究はこの OpenRISC の PIC のアーキテクチャをもとに，実行中の VM を識別する機能を付加した iPIC を提案する．

### 2.2 iPIC の詳細

図 4 に 4 つの VM を並行して実行するための iPIC の構造を示す．iPIC は従来の PIC のアーキテクチャ に比べ，4 点において大きく異なる．1 点目は PICMR を多重化して仮想化に対応していることである．この多重化は VM の並行実行数分必要となる．各VMはそれぞれに割り当てられた物理的な PICMR を持ち，VM 内のホスト OS はそれぞれの PICMR を制御し，割込みハンドリングを行う．複数ある PICMR の切り替えには VM ID レジスタの値を用いる．VM ID レジスタは我々の先行研究 [5] で用いた現在実行中の VM を識別するレジスタである．VM ID を用いることで，複数存在する PICMR を従来と同一のソフトウェアインターフェースで制御可能となり，iPIC 導入のためのゲスト OS の修正を必要としない．

2 点目は PICPDMR(PIC Pre-Defined Mask Register) と Pre-Mask Function である．本研究では，通常の仮想化処理における Direct I/O 方式 [15] を完全ハードウェアで実現することを目的とするため，I/O ペリフェラルの VM 間共有を行わない．このため，現在実行中の VM 以外が管轄する I/O ペリフェラルからの割込みが発生した際に，現在実行中の VM の割込みハンドラを駆動させてはいけない．この問題を解決するために PICPDMR は用いられる．PICPDMR は移植前のハードウェアで使用していなかった IRQ に，実行前に，あらかじめ 0 を立ててセットおく．Pre-Mask Function は PICMR と PICPDMR からの入力の論理積を演算し，現在実行中の VM が管轄していないペリフェラルからの割込みをマスクする処理を行う．また，PICPDMR は PICMR と同様に並行実行数分用意し，VM ID で選択される必要がある．

3 点目は IRQ Remap である．VM 移植前の IRQ と移植後の IRQ に同一のペリフェラルが割り当てられている保証はない．また，異なるペリフェラルからの割込みを受け付けるために，移植前のハードウェアの制約から，VM 間で同一の IRQ が競合する可能性がある．この問題を解決

するために，iPIC は IRQ Remap を備える．IRQ Remap は VM ID と RIRQ(Real IRQ:移植後ハードウェアの IRQ) を入力として，VIRQ(Virtual IRQ:指定された VM 毎の移植前ハードウェアの IRQ) を出力とする．IRQ Remap は VM ID によって切り替わる IRQ 信号のルーティングの役割を果たす．

4 点目は VM 毎の割込みハンドラのエントリポイントのアドレス (HANDLER ADDRESS) を記憶するテーブルを持つことである．従来の PIC をもつ ORPSoC では，出力 INT がアサートされる時，トラップベクトルのベースアドレス[*2]に規定の例外種を示す定数 (ベクトルアドレス)[*3] が加算され，そのアドレス値が PC にセットされる．これに対し，iPIC の実装の場合は，そのアドレス値に代わり，HANDLER ADDRESS の値をテーブルから読み出し，PC にセットする．このテーブルからの読み出しにも，PICMR の切り替えと同様に，VM ID を用いる．テーブル内の HANDLER ADDRESS は各 VM の外部割込みのベクトルアドレスがセットされており，I/O ペリフェラルから割込み要求があった際に，ハードウェアから現在実行中の VM の割込みハンドラを直接起動することができる．

## 3. 評価

### 3.1 評価環境

評価環境として Digilent 社製 Atlys$^{TM}$ Spartan-6 FPGA 開発ボード [13] の FPGA 上に OpenRISC 1000 命令セットを採用した OR1200 プロセッサと ORPSoC を実装し，構築した．Spartan-6 FPGA 開発ボードには DDR2 SDRAM が 128MB，1Gbit ether phy，HDMI 入出力，AC-97 audio，SPI Flash メモリが 16MB，USB UART，USB HID Host，LED，スライドスイッチ，プッシュボタンの I/O が用意されている．本研究の評価では，I/O 性能を評価する対象としてイーサネットに着目した．OR1200 プロセッサ及び ORPSoC は OpenCores プロジェクトのホームページより Verilog HDL で記述された RTL ソースを取得し，使用した．当該 FPGA 開発ボードは ORPSoC のリファレンスプラットフォームであるため，各種 I/O を制御するためのコントローラが Verilog HDL の RTL ソースで用意されている．イーサネットコントローラはこの ORPSoC 内の RTL ソースで提供されるものを使用し，Spartan-6 上に実装した．本研究の評価で使用したハードウェア設計環境を表 1 に示す．

VM となる OS には Linux を選択した．これは性能評価で一般的に普及・利用されているアプリケーションをベンチマークとして採用できるためである．OpenCores プロジェクトによって GNU ツールチェインが移植されたため，Linux はカーネル 3.1 リリースから OpenRISC に正式対応し，OR1200 プロセッサ上で実行可能である．

VMM は評価対象となる部分である割込みの判断と VM 内のハンドラ呼び出し部の機能のみを OpenRISC のアセンブリ言語を用いて実装した．割込み駆動時に，図 4 で示したハードウェアの処理と同様のソフトウェア処理を行うように設計した．

イーサネットは図 5 で示す接続を行い通信させた．通信性能を計測するためにアプリケーションソフトウェア nuttcp[14] をベンチマークとして使用した．nuttcp は TCP／UDP のネットワークテストツールであり，インターネット上のホスト間のネットワーク状態の調査・検証に一般的に用いられるツールである．nuttcp はクライアントーサーバ間でのイーサネット接続の 1 秒間のデータ通信量を計測することができる．本研究の評価では，UDP 接続において，クライアントからサーバへの送信を 100 回計測し，通信速度 (Mbps) の算術平均をとって評価した．

**表 1** ハードウェア設計環境.

| | |
|---|---|
| 論理合成ツール | Xilinx xst14.4 |
| マッピングツール | Xilinx map14.4 |
| 配線ツール | Xilinx par14.4 |
| ロジック・アナライザ | Xilinx ChipScope Pro 14.4 |

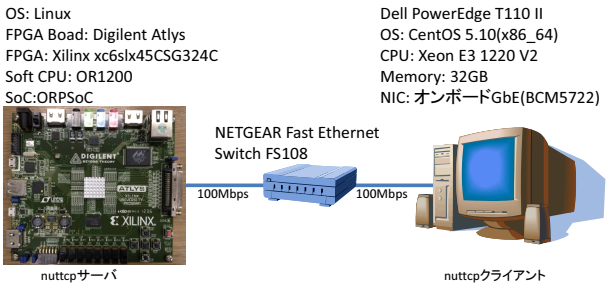


**図 5** I/O 性能評価環境.

**表 2** ハードウェア量の比較.

| | VMM 実装 | iPIC 実装 |
|---|---|---|
| スライス 数 | 6,754 | 6,712 |
| LUT 数 | 12,544 | 12,516 |
| RAM16BWER 数 | 41 | 41 |
| RAM8BWER 数 | 3 | 3 |
| DSP481A1 数 | 4 | 4 |

### 3.2 ハードウェア量及び遅延量の評価

#### 3.2.1 ハードウェア量の評価と見積もり

表 2 に図 4 で示した iPIC を組み込んだ ORPSoC のハードウェア量を示す． 比較対象となる VMM 実装手法は通常の PIC を組み込んだ場合のハードウェア量である．iPIC

[*2] OpenRISC1000 の場合は，SR レジスタの EPH が示す Exceptions Vectors のスタートアドレス
[*3] OpenRISC1000 の場合は，外部割込みのベクトルアドレスは 0x800 に定義されている [9]

**表 3** 最大動作周波数の比較.

| | VMM 実装 | iPIC |
|---|---|---|
| 最大動作周波数 (MHz) | 45.421 | 45.407 |

**表 4** UDP 通信速度の計測結果.

| | VMM 実装 | iPIC |
|---|---|---|
| 通信速度 (MHz) | 0.95186 | 0.95688 |

の実装は，同時に 8 つの VM を稼働することを前提に 8 エントリのアドレステーブルを持つこととした．ORPSoC の物理アドレスが 32 ビットであるため，アドレステーブルを構成するためのメモリ資源は $32bit \times 8entry = 256bit$ が必要となる．また，IRQ の入力が 32 ビットであるから，アドレステーブルと同様に，PICMR と PICPDMR の多重化にも 256 ビット必要となる．これに加え，IRQ Remap 内のルーティング情報を記述するために 256 ビットが必要となる．これらのメモリ資源の合計は 1024 ビット (128 バイト) である．また，メモリ資源の他に，アドレステーブルへの書き込み／読み出し論理と多重化した PICMR と PICPDMR の読み出し論理，IRQ Remap 内のマルチプレクサ，プログラムカウンタへの入力のマルチプレクサを追加する必要がある．

上記のことから，ハードウェア量の増加が予想されたが，実際の配置敗戦後の計測レポートでは iPIC 実装は使用スライス数で約-0.34%減り，LUT 数で約-0.22%減少する結果となった．回路を追加したのにも関わらず，スライス数と LUT 数がごく微量に減少した原因は論理合成および配置配線の最適化によるものである．まず，本計測の実装では iPIC に要するメモリ資源は Verilog HDL 上で予約後の `reg` を用いて定義した．通常，この記述はそのまま，メモリ資源として FPGA 上に確保されるが，本計測は簡単のために，Reset がアサートされる際に当該メモリに書き込む初期値を定数とした．このため，論理合成／配置配線ツールは FSM 解析により，メモリ資源ではなく ROM として生成されたと考えられる．このことから，計測したハードウェア量は iPIC のランダムロジック部の増加分のみが反映されていると考察した．計測はハードウェア量の減少という結果となったが，この変動は 1%未満であることから，最適化の誤差の範囲内であり，iPIC を追加することによるランダムロジック部のハードウェア量の増加は，配置配線の最適化処理の誤差に隠れる程度の規模であることが考察できる．

また，もし，メモリ領域をスライス内に確保したと仮定した場合，iPIC に必要な 128 バイトのメモリ領域は，最低限の見積もりで換算すると，128 スライスの増加で実現できる．これは Spartan 6 の 1 スライス内のストレージエレメント全てをエッジトリガ D フリップフロップとして使用した場合，8 ビット (1 バイト) 分を提供できる試算にもとづく．この場合，約 1.90%のハードウェア量増加を見積もることができる．

#### 3.2.2 遅延量の評価

表 3 に配置配線後の最大動作周波数を示す．iPIC は通常の PIC に比べ，ハードウェアの追加に伴い，動作周波数が約 14KHz(約 0.03%) 減少している．これは命令フェッチのパイプラインステージ (IF Stage) に iPIC の持つアドレステーブルからの読み出しデータパスを追加したことによる遅延の増加が原因である．しかしながら，この遅延量増加も最適化による誤差程度のものであり，問題とはならない．

### 3.3 性能評価

表 3 に nuttcp で計測した通信速度を示す．通信速度はクライアント側から iPIC が実装された Atlys の NIC に UDP パケットを送り，受け付けられたパケットの数によって決まる．通信環境は独立して安定した経路を提供したため，通信速度の差は iPIC による効率化によってどの程度，単位時間当たりにパケットを受け取れられたか (ビジーを回避できたか) で決まる．計測の結果，iPIC による通信速度向上は 0.52%に留まった．我々の先行研究 [5] である I/O ポートアドレス変換の効果が平均で約 10.03%であることから，これに比べ，iPIC の効果は非常に小さい．

これは，I/O ポートアドレス変換は I/O アドレスポートの参照毎に必要とされるのに対し，割込みが発生する回数は相対的に少ないことに起因している．このため，I/O ポートアドレス変換と同様にソフトウェアオーバヘッドの削減効果はあるものの，仮想化の割込み制御をハードウェア化することは大きな性能向上の要因とならないことが明らかとなった．

しかしながら，ソフトウェアで仮想化の割込み制御が実装される場合は，分岐命令や命令キャッシュミスなどの要因で実行時間が変動する可能性をもつ．このため，仮想化を導入したことが割込み応答時間の変動幅を大きくする要因となる．iPIC は全てハードウェアで実装される特性上，このような実行時間変動がない．組込みシステムではリアルタイム性を重視したシステムが存在するため，割込み応答時間の変動を抑制することができる iPIC が有用となる状況が存在すると考察する．

## 4. 関連研究

本研究に近い着眼点の先行研究として Domain Partitioning[16] がある．Domain Partitioning はマルチコア環境において，PPC(Physical Partitioning Controller) と呼ばれるアクセス分割器が内部システムバス上のターゲット

モジュールへの信頼性をチェックする機構を提案している．PPC はアドレスと SrcID と呼ばれるイニシエータモジュールと制御レジスタアドレスを対とした静的な ID を用いて，デバイスへの参照を制御する．Domain Partitioning と本研究の大きな違いとして，Logical Partitioning(仮想化)への適用が挙げられる．Domain Partitioning は Physical Partitioning と呼ばれる CPU コアや I/O ペリフェラル等の SoC の機能を用途毎に静的に分割実装する手法に着目し，その実装方式における効率的な資源割り当ての提案を行っている．

一方，本研究は仮想化に着目し，マルチコア環境でなくとも，単一のハードウェア資源に複数のオペレーティングシステム環境が構築でき，資源の分割・共有を行うフレキシブルなシステム上での I/O 性能向上を目的としている．先行研究では仮想化による目的の実現はソフトウェアオーバヘッドが大きいため，より現実的な Physical Partitioning を対象としたが，本研究はそのソフトウェアオーバヘッドの削減を対象としていることが，本研究の位置づけである．

## 5. おわりに

本研究は，仮想化の割込み制御をハードウェア化した高機能割込みコントローラ iPIC を提案し，性能を評価した．性能評価ではイーサネットの UDP 通信を行った．この結果，iPIC は大きな性能向上の要因とならない結果を得た．しかしながら，同時に，iPIC には割込み応答時間の変動を抑制する効果があり，リアルタイムシステムに有効な可能性があることを議論した．

先行研究 [5] を含む本研究の目的は組込み仮想化の全ての機能をハードウェア化し，SoC 設計時の機能部品の選択肢として提供することにある．これにより，仮想化のソフトウェアオーバヘッドを削減するとともに，実行時間変動を抑制し，リアルタイムシステムへの仮想化の適用を検討することが可能となる．

今後の課題として，割込み応答時間変動の抑制効果を我々の先行研究と合わせて検証することが挙げられる．

**謝辞** 本研究は公益財団法人　澁谷学術文化スポーツ振興財団　平成 25 年度 大学の新技術研究に対する奨励金　研究課題名：「ハードウェアで実現する組込みシステム向けハイパーバイザの開発」，並びに，公益財団法人　服部報公会 平成 26 年度 工学研究奨励援助金 研究課題名：「ハードウェアによる組込み仮想化の VM 間リアルタイム・タスクスケジューラの開発」により行われた．

また，本研究の評価にあたり，Xilinx FPGA 開発ツール群の使用方法に関してご指導を頂いた五大開発株式会社 技術研究所 荒木 光一 氏に深く感謝する．

## 参考文献

[1] R. Kaiser and S. Wagner, “Evolution of the PikeOS Microkernel”, Proc. of first International Workshop on Microkernels for Embedded Systems, 2007.

[2] G. Heiser, B. Leslie, “The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors”, Proc. of the first ACM Asia-Pacific Workshop on Systems, pp.19-24, 2010.

[3] U. Steinberg, B. Kauer, “NOVA: a Microhypervisor-based Secure Virtualization Architecture”, Proc of the 5th European conference on Computer systems, pp.209-222, 2010.

[4] “CODEZERO® Embedded Hypervisor”,B LABS, in http://b-labs.com/products/.

[5] 請園 智玲, “組込み仮想化のハードウェア I/O アドレスマップ変換”, 情報処理学会論文誌「組込みシステム工学」特集, Vol.55, No.8, pp.1830–1849, 2014.

[6] “Enabling Intel® Virtualization Technology and Benefits”, Intel White Paper, in http://www.intel.in/content/dam/www/public/us/en/documents/white-papers/virtualization-enabling-intel-virtualization -technology-features-and-benefits-paper.pdf, 2010.

[7] “8259A PROGRAMMABLE INTERRUPT CONTROLLER”, Datasheet of Intel Corporation, in http://pdos.csail.mit.edu/6.828/2010/readings/hardware/8259A.pdf

[8] “Intel®82093AA I/O Advanced Programmable Interrupt Controller (I/O APIC)”, Datasheet of Intel Corporation, in http://download.intel.com/design/chipsets/datashts/29056601.pdf

[9] D. Lampret, C. Chen, M. Mlinar, J. Rydberg, M. Ziv-Av, C. Ziomkowski, G. McGray, B. Gardner, R. Mathur and M. Bolado, OPENCORES.ORG, “OpenRISC Architecture Manual”, in http://www.da.isy.liu.se/courses/tsea44/OpenRISC/openrisc_arch3.pdf, 2003.

[10] D. Lampret, , OPENCORES.ORG, “OR1200 OpenRISC Processor” in http://opencores.org/or1k/OR1200_OpenRISC_Processor

[11] “ORPSoC HP”, OPENCORES.ORG, in http://opencores.org/or1k/ORPSoC

[12] J. Baxter, OPENCORES.ORG, “ORPSoC User Guide, Issue 4 for ORPSoC”, in http://realtimeembedded.com/sites/default/files/Blog/Modesty/orpsoc.pdf.

[13] “Atlys™ Board Reference Manual”, Digilent, Inc, http://www.digilentinc.com/Data/Products/ATLYS/Atlys_rm.pdf

[14] “nuttcp”, nuttcp development team, in http://nuttcp.org/nuttcp/Welcome%20Page.html

[15] “Intel Virtualization Technology for Directed I/O Architecture Specification Rev.2.2”, Intel Corporation, http://www.intel.com/content/dam/www/public/us/en/documents/ product-specifications/vt-directed-io-spec.pdf, Sep, 2013.

[16] T. Nojiri, Y. Kondo, N. Irie, M. Ito, H. Sasaki and H. Maejima, “Domain Partitioning Technology for Embedded Multicore Processor”, IEEE Micro, Vol. 29, No. 6, Nov/Dec, 2009