組込み向け CPU 仮想化技術対応ハイパーバイザの設計

茂田井 寛隆† 山本 整† 落合 真一†
安達 浩次†† 鈴木 均†† 城倉 梨香†† 福井 昭也†††
小川 敏行†† 田原 康宏†† 奥村 直人††

産業機器や車載システムなどのリアルタイム制御が重視される分野では，複数のコントローラを用いて制御処理を実現してきた．これに対し，複数の制御処理を 1 つの CPU で実行するための CPU 仮想化技術を開発している．本論文では，上記 CPU 仮想化技術であるハードウェアマルチスレッド（クロックレベルでの切替え実行）および仮想マシン（ハードウェアリソースの分割占有）と，それら特徴を活かすハイパーバイザの設計内容を述べる．

本ハイパーバイザは，ハイパーバイザ機能をハードウェアマルチスレッドにて動作させることに特長がある．本特長により，ハイパーバイザ機能実行中でも他の OS は継続動作できるため，従来課題であったハイパーバイザ動作による OS のリアルタイム性への影響を抑えることができると考える．

A Design of Hypervisor Using Virtualization Technology for Embedded CPU architecture

Hirotaka Motai† Hitoshi Yamamoto† Shinichi Ochiai†
Koji Adachi†† Hitoshi Suzuki†† Rika Joukura†† Akiya Fukui†††
Toshiyuki Ogawa†† Yasuhiro Tawara†† Naoto Okumura††

A virtualization technology have been developed for embedded CPU architecture that allocates CPU time and system resources for independent processes in a control system. This technology dramatically reduces process interference and facilitates high-speed execution of diverse tasks.

We have designed a hypervisor with such technology. The new hypervisor has a special feature that run several hypervisor functions only on a hardware thread, in order to eliminate the influence by the hypervisor execution.

1. はじめに

産業機器や車載システムなどのリアルタイム制御が重視される分野では，多様な制御処理を複数のコントローラを用いて実現してきた．これに対し，複数の制御処理を 1 つの CPU で実行する CPU 仮想化技術を開発している．[1]

一般に，複数の制御処理を 1 つの CPU 上にて動作させる方法として仮想化技術がある．仮想化技術はサーバ分野で利用されてきた技術であるが，近年組込み機器に向けた適用研究が行われている．例えば，サーバ分野で利用されている Xen[2]をリアルタイム化する研究[3]や，組込み機器向け CPU である SH にて軽量な仮想化技術を実現する研究[4]がある．これら関連研究はマルチコア CPU の CPU コアを OS に占有させることでリアルタイム性を確保しているが，仮想化技術を管理するハイパーバイザが動作している間は OS の動作が保留されるため，OS のリアルタイム性の低下を招いている．我々は，OS 間の保護分離や OS 間通信などの基本的な機能を備えるとともに，ハードウェアマルチスレッド技術を活用し，ハイパーバイザ動作による OS のリアルタイム性の低下を一定範囲内に抑えるハイパーバイザを考案した．

本論文では，ハードウェアマルチスレッド技術を活用した上記ハイパーバイザの設計について述べる．本設計により，従来問題であったハイパーバイザ動作によるオーバヘッドを一定時間に制限でき，OS のリアルタイム

† 三菱電機株式会社, Mitsubishi Electric Corporation
†† ルネサス エレクトロニクス株式会社, Renesas Electronics Corporation
††† 株式会社ルネサスソリューションズ, Renesas Solutions Corporation

性の低下を防止することができると考えている.

本論文の構成は以下のとおりである. まず2章で組込み機器向け CPU 仮想化支援機能について述べる. 次に3章でハイパーバイザの課題について述べ, 4章で課題を解決するための設計について述べる. そして, 5章で設計に基づいた実装に関して述べ, 6章でまとめと今後の課題を述べる.

2. CPU 仮想化支援機能

本章では, 本開発で使用する組込み機器向け CPU の仮想化支援機能について説明する.

2.1. ハードウェアマルチスレッド

本 CPU は複数のハードウェアスレッド(以下, スレッドと記す)を持つ. スレッドはソフトウェアの実行単位であり, プログラム実行に必要不可欠なレジスタを持つ. 例えば, 汎用レジスタ, システム制御レジスタなどである. 更に, 本 CPU はクロック毎に実行するスレッドを切り替える機能を持つ. これにより, 複数のスレッドを並行実行できる.

スレッドの切り替えはハードウェアスケジューラによって自動で行われる. スレッドの実行順序は任意にソフトウェアから設定可能であり, 一定時間におけるスレッドの実行回数(以下, 実行率と記す)を調整できる. ユーザは実行率を任意に設定できるため, 異なる周波数の CPU コアを持つ仮想的なマルチコア CPU を用意でき, ソフトウェア処理負荷にあった性能配分が可能となる. 例えば, 本機能を搭載した CPU 300MHz にて, 従来 CPU 周波数が 150MHz のシステム A と 75MHz のシステム B と C とを動作させる場合, 表 1に示すスレッドの実行率を設定することにより, 仮想的に 150MHz の CPU コアを 1 つと, 75MHz の CPU コアを 2 つ持つマルチコア CPU 相当となる.

表 1 クロック性能比較

システム	A	B	C
従来 CPU 周波数	150MHz	75MHz	75MHz
実行率	50%	25%	25%

本例において, システム A, B, C をそれぞれスレッド 0, 1, 2 で動作させた場合の実行スケジュール例を図 1に示す. ユーザはスレッドの実行順序を指定できるため, 図 1以外の順番で実行することもできる.

従来, CPU 性能を分割しユーザが設定したい割合でそれぞれの OS に性能を割当てるためには, ソフトウェアにて OS の実行時間調整や実行 OS の切替えを行う必要があった. その際不可欠であった実行時間管理や OS 切り替えにかかる CPU の処理時間は本 CPU のハードウェアスケジューラにより不要となる. ソフトウェアによる切替えでは切替え粒度の最小化に限界があったが, ハードウェアマルチスレッド技術により 1 クロック単位で切替え可能となり, リアルタイム性を損なわずに複数 OS の並行実行ができる.

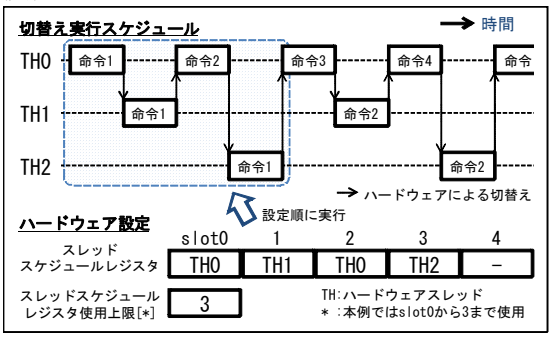

図 1 実行スケジュール

2.2. 仮想マシン

本 CPU はハードウェアに搭載されているデバイスの中から使用するデバイスを選択し, 任意の仮想的なハードウェア(以下, 仮想マシンと記す)を構築する機能を持つ. 選択対象は, スレッド, メモリ領域やデバイス, 割込みである. スレッドは, 1 つの仮想マシンに対して, 単一または複数割り当てることができる. メモリ領域はアクセス可能な範囲をアドレス範囲で仮想マシンに設定できる. 特定領域を複数の仮想マシンからアクセス可能と設定することもできる. デバイスおよび割込みは単一の仮想マシンに割当てる.

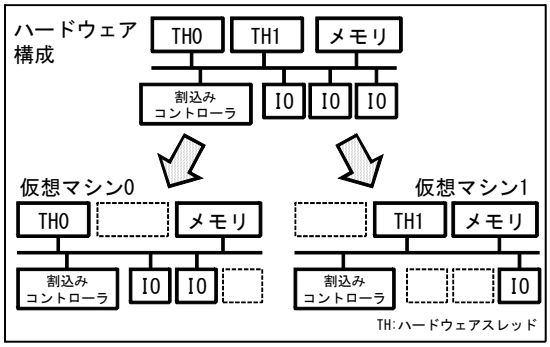

図 2 仮想マシン

仮想マシンを実現するために 2 つのハードウェア機能が搭載されている. 1 つは, 仮想マシン間での論理アドレス重複やメモリ資源の誤操作を防ぐアドレス変換機能である. これにより, OS のコード領域やデータ領域などのメモリ資源を, 適切に分離・保護できる. もう 1 つは, 仮想マシン間で独立した優先度管理を行うことが可能な割込み管理機能である. これにより, 仮想マシン毎に専用の割込みコントローラを持つことができる. この 2 つ

の機能により，他の仮想マシンから機能的な制限を受けない仮想マシンを構築できる．

2.3. ハイパーバイザ特権

本 CPU は，従来の 2 段階のモード（ユーザモードとスーパバイザモード）に加えて，新たにハイパーバイザ特権を持つ．この特権は仮想マシンやスレッドに対する操作を行うために必要な権限である．ハイパーバイザ特権は，仮想化機能を有効にする前のネイティブモードのスーパバイザモード，もしくは特定の仮想マシンのスーパバイザモードに対して付与される．またハイパーバイザコール実行中にも一時的に付与される（図 3）．

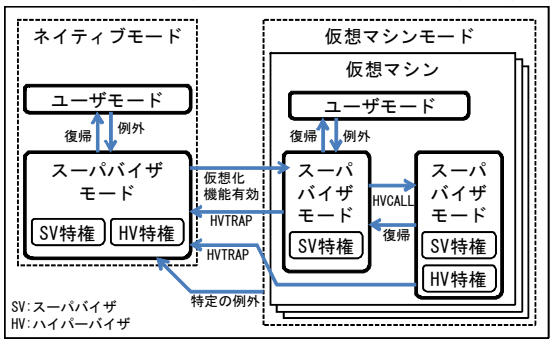

図 3 CPU 動作モードと特権

ハイパーバイザを呼び出すための仕組みとして，ハイパーバイザトラップ（HVTRAP）とハイパーバイザコール（HVCALL）が用意されている．共に専用のソフトウェア例外を発生させる命令である．HVTRAP は全ての仮想マシンを停止させ，ネイティブモードに移行する．HVCALL はスレッドにハイパーバイザ特権が付与され実行される．ハイパーバイザ特権が付与されると，そのスレッドは全ての領域へのアクセスが可能となる．つまり，HVTRAP の場合は全 OS を止め，HVCALL の場合は他の OS を動作させたままとなる．（図 4）

特定の割込み・例外発生によりハイパーバイザトラップを発生させることができる．これは，ハイパーバイザが直接割込みを処理するための仕組みである

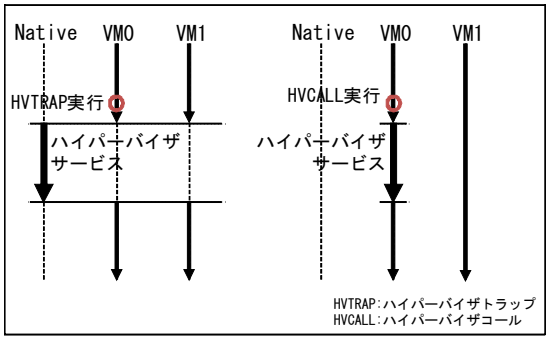

図 4 ハイパーバイザ呼出し方法

3. 課題

2章で述べた組込み機器向け仮想化支援機能を用いて，リアルタイム OS（RTOS）を複数動作させるハイパーバイザの実現課題を整理した．本章では，それら課題について述べる．

3.1. ハイパーバイザ動作時間の制限

RTOS 上のタスクは決められた時間内に処理を完了する必要がある．そのため，RTOS 動作時間内に RTOS 動作を保留してハイパーバイザが動作する場合，その時間も加味して時間設計を行う必要がある．

ハイパーバイザの動作は，同期サービスと非同期サービスに分けられる．同期サービスは RTOS からの要求のタイミングで行われる処理である．例えば，OS 間通信の送受信要求は同期サービスである．非同期サービスは RTOS 以外からの要求のタイミングで行われる処理である．例えば，ハイパーバイザに割当てたデバイスから割込みが発生したタイミングで実行される．

ハイパーバイザの同期サービスは発生するタイミングが予測でき，要求から処理完了までの実行最悪時間を測定することにより，時間設計に含めることができる．しかし，非同期サービスは発生タイミングが予想できない．更に，システム全体に関わる処理をハイパーバイザは行うため，RTOS より動作が優先される．よって，非同期サービスの動作タイミングが重なった場合，ハイパーバイザの動作時間が増え，RTOS のリアルタイム性低下を招く可能性がある．

本課題を解決するために，非同期サービスの動作時間に上限を設け，RTOS への影響を抑える．

3.2. OS 実行環境の構築

従来と同等のシステム性能を満たすために，RTOS へ適切にハードウェアリソースを割当てる必要がある．ハードウェアリソースとは，CPU 使用率，メモリおよびデバイス領域，デバイス割込みを指す．割当てられたリソースは RTOS が占有利用でき，かつ，他の RTOS によるアクセスから保護される必要がある．

本 CPU の CPU 使用率は実行率である．従来システムの CPU クロック比などの性能比に従い，RTOS のスレッド実行率を決定する．

それぞれの RTOS に対し，使用可能なメモリ領域，デバイス領域を独立に割当てる．各 RTOS は割当てられた領域にて動作するように設計変更するか，ハードウェアのアドレス変換機能を活用して，それぞれの RTOS が参照する論理アドレス空間を変更する．また，他の RTOS からデータが破壊されないようにハードウェアのアドレス空間保護機能によりアクセス保護を行う．

RTOS に割当てたデバイスの割込みは仮想マシンの割込みコントローラにバインドする．これにより，デバイスから割込みコントローラへ通知された割込みは仮想マシンの割込みコントローラを経由し，直接スレッドに通知される．RTOS は仮想マシンの割込みコントローラに対し，デバイスからの割込み優先度を任意に設定する．割込み優先度は仮想マシン間の割込みコントローラで競合しないため，RTOS は他の OS やハイパーバイザの動作の影響を受けることなく割込みを処理できる．

上記に述べたとおり，適切にハードウェアリソースを RTOS に割当て，RTOS が動作可能な環境を構築する．

3.3. OS 間通信

従来機器間で行っていた通信をハイパーバイザ上の OS 間で行う必要がある．

OS 間で通信を行う方法として，(a)ハイパーバイザ上に従来の通信デバイスを模擬する方法，(b)各 OS 間で同一のメモリ範囲（共有メモリ）を参照する方法，(c)ハイパーバイザが通信サービスを提供する方法が考えられる．

方法(a)は通信デバイスを模擬するためにオーバヘッドが発生する．組込み機器の限られた性能制限の中で実現するのは難しい．方法(b)は RTOS より直接データを参照するため，高速に通信できる．しかし，OS 間で決められた手順を考える必要がある．方法(c)はハイパーバイザに通信を要求する処理にオーバヘッドが発生するが，ハイパーバイザにより管理データが保護されるため，安全に通信を行うことができる．

我々は，従来と同等以上の性能と安全性を考え，方法(c)にて OS 間通信機能を実現する．

3.4. ハードウェアリソース不足の解消

ひとつのハードウェア上で複数の RTOS が動作するために，それぞれの OS が使用するデバイスを搭載する必要がある．しかし，搭載 OS 数の増加などの構成変更に対して，追加 OS が使用するデバイスをハードウェアに追加することは現実的には難しい．そのため，リアルタイム処理に関係ない同一用途のデバイスに関しては，OS 間で共有して利用できると望ましい．

デバイスを共有する方法として，(a)ハイパーバイザがメモリ上に疑似的なデバイスを模擬し，それぞれの OS のデバイスドライバから利用する方法と，(b)ハイパーバイザがデバイスドライバを持ち，OS にデバイスアクセス I/F を提供する方法と，(c)ひとつの OS（管理 OS）がデバイス操作を行い，他の OS は管理 OS にデバイス要求を伝達する方法が考えられる．

方法(a)は共有デバイスを模擬するためにオーバヘッドが発生する．組込み機器の限られたリソースで実現するのは難しい．方法(b)はハイパーバイザにデバイスドライバを実装することと，ハイパーバイザ提供 I/F を使用する簡易的なデバイスドライバが OS に必要となる．方法(c)は OS 間での要求データ通信の仕組みが必要となる．方法(b)はハイパーバイザにて，方法(c)は管理 OS にてデバイスの状態管理や使用権管理，送受信データ転送，割込み模擬といった処理が必要で，オーバヘッドが発生する．

我々は，管理 OS に非同期なオーバヘッドが生じる方法(c)は望ましくないと考える．また，3.1節で述べたハイパーバイザ動作時間の制限により，OS のリアルタイム性への影響を防止できるため，方式(b)にてデバイス共有を実現する．

4. 設計

本章では，課題を解決するためのハイパーバイザの設計について述べる．

4.1. ソフトウェア構成

本ハイパーバイザは，RTOS のリアルタイム性を確保するため，非同期サービスの動作時間を制限する．非同期サービスの処理時間を制限する方法としてハードウェアマルチスレッドの仕組みを利用する．具体的には，ハイパーバイザの非同期サービスをスレッドにて実行し，本スレッドの実行回数を RTOS の実行率を守れる範囲に設定することで，ハイパーバイザ非同期サービスの動作時間上限を設定する．本方式により，RTOS のリアルタイム性を守りつつ，ハイパーバイザの非同期サービスを実行できる．ソフトウェア構成を図 5に示す．

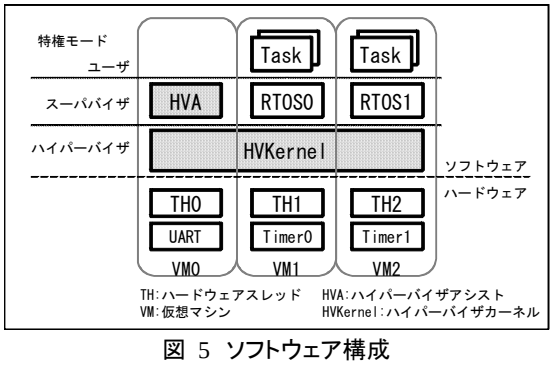

図 5 ソフトウェア構成

各 RTOS はそれぞれ 1 つの仮想マシン内で動作する．ハイパーバイザはハイパーバイザカーネルおよびハイパーバイザアシストにて構成される．ハイパーバイザカーネルはハードウェア起動時，および HVCALL による RTOS からハイパーバイザ呼び出しにより実行され

るプログラムである．ハイパーバイザアシストは RTOS と同じように 1 つの仮想マシンにて動作するプログラムである．ハイパーバイザ機能の内，非同期サービスは全てハイパーバイザアシストが実行する．

4.2. リソース配分

RTOS が動作するために，スレッド実行率，メモリおよびデバイス領域割当て，デバイス割込みのバインドを設定する．

RTOS が必要とする CPU 性能として，今回は評価ボードの CPU 性能に対する RTOS0 の必要性能が 50%以上，RTOS1 の必要性能が 25%以上として，スレッド実行率を設定した．スレッド実行順序は仮想マシン単位で指定し，VM0→VM1→VM2→VM1 の順にてスレッドスケジュールレジスタおよびスレッドスケジュールレジスタ使用上限に設定する．ハイパーバイザアシストを含めた各 RTOS の実行スケジュールを図 6に示す．

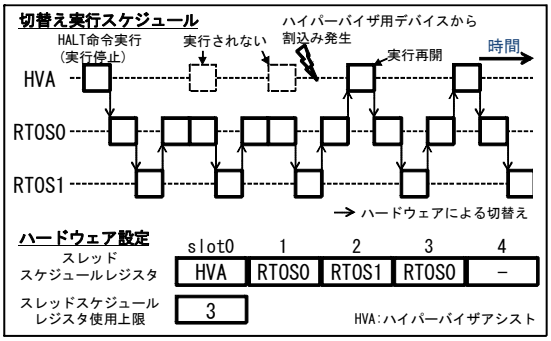

図 6 実行スケジュール

ハイパーバイザアシストは定常時は実行停止しており，その時間（スロット）は詰めて実行されるため，RTOS0 および RTOS1 は設定した実行率以上実行される．

メモリおよびデバイス領域の設定は，仮想マシンに対して行う．ハイパーバイザアシストはハイパーバイザ機能を実行するために全ての領域を参照できる必要がある．そのため，仮想マシン 0 は全ての領域をアクセス可能とする．また，ハイパーバイザ特権を付与する．仮想マシン 1 と仮想マシン 2 は，各 RTOS からそれぞれのコードが配置された ROM 領域とデータ用にアクセスする RAM 領域，割当てられたデバイス領域にアクセス可能とする．また，他の OS に割当てられた領域をアクセスできないよう保護を設定する．（図 7）

各デバイス割込みはそのデバイスを使用する OS の仮想マシンにバインドする．未使用のデバイスやスプリアス割込みは仮想マシン 0 にバインドし，ハイパーバイザアシストによりスプリアス割込みとして処理する．これにより，意図しない割込みによる HVTRAP 発生を抑制し，RTOS 動作を継続させる．

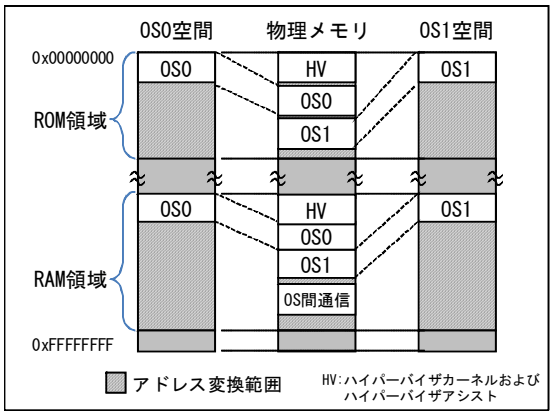

図 7 アドレスマップ

4.3. OS 間通信

ハイパーバイザが提供する OS 間通信機能は Emblix により策定された仕様[5]をベースとし，FIFO 機能および共有メモリ機能を持たせる．各 RTOS とハイパーバイザアシストはハイパーバイザコールにより，これら通信機能を利用する．

FIFO 機能は，OS 間で他の動作 OS に対して，同期または非同期に通知する仕組みである．OS 間で FIFO キューを 2 本用意し，数バイト程度のデータ転送を行う．各 OS への同期通知のために OS 間通信割込みを用いる．本割込みはハイパーバイザを経由せず，直接仮想マシンの割込みコントローラへ通知される．

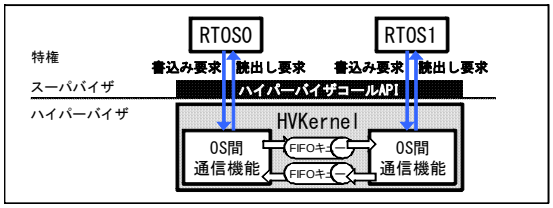

図 8 OS 間通信（FIFO）

共有メモリ機能は，OS 間で共有するメモリ領域を提供すると共に，OS 間の排他制御を目的としたバイナリセマフォを提供する．OS 間ではセマフォを獲得してから書込みを行う手順とすることで，データの一貫性を保つことができる．

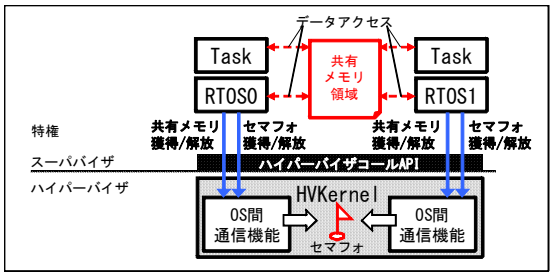

図 9 OS 間通信（共有メモリ）

4.4. デバイスの共有

デバイスの共有のためには，デバイスの状態管理や使用権管理，送受信データ転送，割込み模擬が必要である．これら処理はハイパーバイザアシストが行う．今回は UART 出力を対象とし，設計を行う．

ハイパーバイザアシストは RTOS から UART 出力要求を受けると，ハイパーバイザアシストのデバイスドライバ経由でデータを出力する．使用権管理は出力の切替え単位を行単位（¥n の出力まで）とすることで，ハイパーバイザアシストにて自動に切り替える．また，出力文字列の先頭に"[RTOS0]"といった文字列を付与することで，出力データを簡単に分離できるようにする．送信データ転送は OS 間通信の FIFO にて行う．各 RTOS から UART に出力したい文字データは FIFO を通じてハイパーバイザアシストへ送信する．UART の書込み完了の割込み模擬は，FIFO の読出し完了割込みにて代用する．

5. 実装

設計の有効性を検証するため，4章にて述べた設計に基づき，ハイパーバイザを実装した．動作環境として，FPGA 上に仮想化支援機能をもつ CPU を実装した．各 RTOS に uT-Kernel を用いた．

FPGA 実装の都合により，仮想マシンとスレッドの上限数はそれぞれ 2 つである．ハードウェア仕様に対応するため，ソフトウェア構成を一部変更した．具体的には，RTOS0 にハイパーバイザアシスト機能を付与して動作させた．RTOS0 の動作時間にはハイパーバイザの非同期サービス処理を含むため，設計の有効性検証対象に適さないが，RTOS1 は元の設計と同等の環境を用意できており有効性検証対象として問題ないと考えている．今後，性能評価していく予定である．

ハイパーバイザおよび uT-Kernel の使用 ROM サイズと RAM サイズを表 2に示す．uT-Kernel に比べてハイパーバイザの使用サイズは共に小さく，組込み機器にも適用できる範囲と考えられる．

表 2 ROM および RAM の使用サイズ

	ハイパーバイザ	uT-Kernel（RTOS1）
使用 ROM サイズ	約 40.9KByte	約 57.0KByte
使用 RAM サイズ	約 3.8KByte	約 22.0KByte

6. おわりに

本論文では，我々が考える組込み機器向け CPU 仮想化支援機能とハイパーバイザの設計について述べた．本ハイパーバイザは，OS 間の保護分離や OS 間通信，ハイパーバイザコールなどの基本的な機能を備えるとともに，組込み機器向け仮想化支援機能を活用したソフトウェア構成とした．従来構成では，ハイパーバイザ動作中は OS 動作が保留され，OS のリアルタイム性を低下させていた．本構成では，ハイパーバイザの代替処理を行うプログラムを設け，ハードウェアマルチスレッド実行（クロックレベルでの切替え実行）により OS と共に切替え実行することで，ハイパーバイザ機能実行中でも他の OS は継続動作できるため，従来構成で課題であったハイパーバイザ動作による OS のリアルタイム性への影響を抑えることができると考える．

今後は，本論文で述べた設計の有効性を検証するために，実装したハイパーバイザおよび RTOS を用いて性能評価を行い，RTOS のリアルタイム性への影響を明らかにしていく予定である．

参考文献

[1] リアルタイム制御システムに適した V850 CPU 向け仮想化技術を開発 http://japan.renesas.com/press/news/2010/news20100929.jsp

[2] "Xen and the Art of Virtualization", Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield. In Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003, pp. 164–177（2003）

[3] 「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」，金城 聖，永島 力，元濱 努，片山 吉章，毛利 公一．情報処理学会研究報告. EMB，組込みシステム 2008, pp 9-16

[4] 「SPUMONE：軽量な CPU 仮想化手法」，湯村 悠，神田 渉，香取 知浩，杵渕 雄樹，中島 達夫，全国大会講演論文集 第70回,平成20年, pp "5-329"-"5-330"

[5] 「Linux における RTOS とのハイブリッド構成に関する仕様（第 1 版）」，日本エンベデッド・リナックス・コンソーシアム（Emblix）ハイブリッドアーキテクチャワーキンググループ活動報告書，2002 年 8 月