社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－112　(21)
2003／11／28

マルチプロセッサSoPCにおけるHardware/Software一体構成技術

西山　零士†　　本田　晋也†　　高田　広章††

† 豊橋技術科学大学 情報工学系　〒441-8580 愛知県豊橋市天伯町雲雀ヶ丘1-1
†† 名古屋大学大学院 情報科学研究科　〒464-8608 愛知県名古屋市千種区不老町
E-mail: †{reiji,honda,hiro}@ertl.jp

あらまし　FPGAの集積度の向上とともに，FPGAにより機能分散型マルチプロセッサシステムを実現するマルチプロセッサSoPC（MPSoPC）が可能になってきている．MPSoPC用のRTOSは，MPSoPCの多様なシステムアーキテクチャに合わせてコンフィギュアラブルである必要がある．我々は，RTOSのコンフィギュレーションに伴う作業の軽減を図るために，システムコンフィギュレーション記述と呼ぶ一つの記述からシステムアーキテクチャとRTOSを同時にコンフィギュレーションするHardware/Software一体構成技術について研究を行っている．本論文では，MPSoPCの特徴とMPSoPC用のRTOSに求められる要件およびHardware/ Software一体構成技術ついて論じる．本研究の評価環境を整えるため，MPSoPC対応RTOSの開発を行った．次に，一体構成を行うシステムコンフィギュレータの実装を行い，実際にシステムコンフィギュレーションから一体構成可能か検証を行う予定である．
キーワード　FPGA，マルチプロセッサ，SoPC，RTOS，Hardware/Software，一体構成技術

Hardware/Software Co-Configuration for Multiprocessor SoPC

Reiji NISHIYAMA†, Shinya HONDA†, and Hiroaki TAKADA††

† Department of Information and Computer Sciences, Toyohasi University of Technology
Hibarigaoka 1-1, Tenpaku-cho, Toyohashi, Aichi, 441-8580 Japan
†† Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Aichi, 464-8603 Japan
E-mail: †{reiji,honda,hiro}@ertl.jp

Abstract　Recent advances in FPGA technologies make it possible to implement a whole embedded system using multiple processors on a FPGA. We call a multiprocessors embedded system implemented on a FPGA multiprocessors System-on-a-Programmable-Chip(MPSoPC). Real-time operating systems(RTOS) used for such MPSoPC should be configurable to a wide range of MPSoPC architecture. However, it is troublesome to configure RTOS to MPSoPC architecture. Therefore, we propose Hardware/Software Co-Configuration. In this paper, we represent the property required of RTOS for MPSoPC, and Hardware/Software Co-Configuration. We first developed configurable RTOS for MPSoPC, and then we intend to implement Co-Configuration tool called System Configurator and verify whether it is possible to co-configure MPSoPC architecture and RTOS using this tool.
Key words　FPGA, SoPC, Hardware/Software Co-Configuration, RTOS, Embedded Systems

1. はじめに

近年，FPGAの集積度の向上により，System-on-a-Programmable-Chip（SoPC）と呼ばれるように単一のFPGA上にシステム全体を実現することが可能になっている．さらに，複数のプロセッサを含むシステムを構築するマルチプロセッサSoPC（MPSoPC）も可能となっている．MPSoPCの大きな利点は，設計変更に対する柔軟性であり，MPSoPCを用いたシステム設計では，プロセッサの数や種類，バス構成，プロセッサ間通信の方法などを応用毎に最適なものを選択することが可能である．

このような特徴をもつMPSoPC用のRTOSは，マルチプロセッサ対応でかつSoPCの幅広いシステムアーキテクチャ（プロセッサの数や種類，周辺デバイス，バス構成）に対応してコンフィギュアラブルである必要がある．また，アプリケーションソフトウェアの開発を効率化するために，プロセッサ間の通

信はアプリケーションレベルではなくOSレベルでサポートする必要がある[1][2]．プロセッサ間通信をOSレベルで効率よくサポートするためには，システムアーキテクチャに応じて各プロセッサ毎にOSを適切にコンフィギュレーションする必要がある．

しかしながら，このコンフィギュレーションは非常に手間を要する．

そこで我々は，この作業の軽減を図ることを目的に，ハードウェアとOSを含むソフトウェアを一体に構成する手法について研究を行っている．本手法では，システムコンフィギュレーション記述と呼ぶ一つの記述からシステムアーキテクチャとRTOSのコンフィギュレーションを同時に行うアプローチをとり，これを実現するためのツールをシステムコンフィギュレータと呼ぶ．

本稿では，まずMPSoPCの特徴について述べ，MPSoPC用RTOSの必要性とそのRTOSに求められる要件を明らかする．さらに，MPSoPCとRTOSのコンフィギュレーションに伴う問題点を指摘し，この問題を解決するための方法として我々が提案する，Hardware/Software一体構成技術ついて論じる．

我々は，これまでにMPSoPC用のRTOSを開発し，現在，一体構成技術により図3に示すようなマルチバス構成の機能分散型共有メモリマルチプロセッサシステムとそのRTOSを一体に構成するシステムコンフィギュレータの実装を行っている．

2. 組込み用マルチプロセッサシステム

本章では，近年の組込みシステムで用いられるマルチプロセッサシステムの特徴と，それに対して求められるRTOSの要件について述べる．

2.1 組込み向けマルチプロセッサシステム

携帯電話に代表されるような大規模な組込みシステムでは，処理能力の向上を図るためにしばしばマルチプロセッサシステムが用いられる。

一般的に組込みステムにおいては，設計段階で必要となる処理の最大処理量や負荷の見積りが可能である．そのため，負荷分散の余地が大きい場合に有効となる対称型の構成よりも，音声・画像，ユーザインターフェイス，入出力といった処理を行うプロセッサを限定した，機能分散型のマルチプロセッサシステムが有力なアプローチとして用いられる[1][2]．

機能分散型の構成による組込みシステムの設計においては，各プロセッサ間の接続に対して，共有メモリ接続，メインバスとI/Oバスといったマルチバス接続，デュアルポートメモリ接続，あるいは専用チャネルによる接続などさまざまな方法が用いられる．

さらに，プロセッサについても，汎用プロセッサのみならず，汎用プロセッサにDSPあるいは他の特殊用途プロセッサといった異種のプロセッサの組合わせで採用されることが多い（ヘテロジニアスマルチプロセッサと呼ぶ）．

2.2 マルチプロセッサSoPC

これまで，マルチプロセッサシステムの多くは，複数のチップを基板上に載せることで実現していたが，近年のLSI技術の発達によりこれらをまとめて1チップで実現することが可能になっている．さらにFPGAの集積度の向上により，System-on-a-Programmable-Chip（SoPC）と呼ばれるように単一のFPGA上でも同様にマルチプロセッサシステムを実現することが可能となった．我々は，特にこれをマルチプロセッサSoPC（MPSoPC）と呼ぶ．

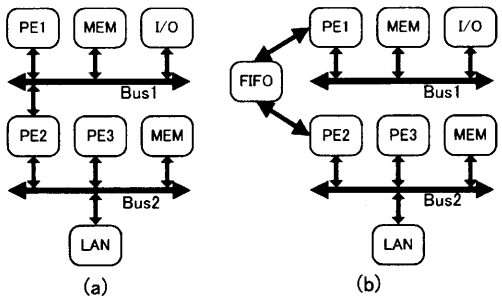

図1　システムアーキテクチャの最適化

MPSoPCではシステム要求の変更に伴う設計の変更に対し柔軟性を持つため，プロセッサとそのペリフェラルの数や種類，さらにプロセッサ間の接続方法（ないしはプロセッサ間通信の方法）等は，応用毎にその構成を容易に変更可能である．そのため，OSを含むソフトウェア（アプリケーション）の構成に応じて最適なシステムアーキテクチャをとることができる．

例えば，図1（a）のシステムにおいては，Processing Element（PE）1はBus1のメモリ，I/Oにはアクセスするが，Bus2にあるリソースにはアクセスしない．それに対して，PE2はBus2のリソースにはアクセスするがBus1のリソースにはアクセスしない．しかしながら，PE1とPE2の間はプロセッサ間通信が行われているため，PE2はバス1に接続されている．このような場合，図1（b）のようにPE1-PE2間通信専用ハードウェアを設けることで，PE2はバス1から切り離され，バス1の使用効率を上げるとともに，リアルタイムシステムでは重要な要素である予測可能性も向上すると考えられる．

このようにSoPCでは，アプリケーションの要求に合わせてさまざまなシステムアーキテクチャをとりえる．

2.3 マルチプロセッサSoPC用のRTOS

本節では，リアルタイムシステムをMPSoPC上に実現する場合に必要となるRTOSの必要性と諸要件について，シングルプロセッサ対応のカーネルを各プロセッサで動作させるアプローチ[2]の問題点と，前節で述べたMPSoPCの特徴を踏まえ議論する．

2.3.1 シームレスなマルチプロセッサ環境

組込みシステムの多くは，高い実時間性が要求されるリアルタイムシステムである．こうしたリアルタイムシステムを機能分散型マルチプロセッサ上に実現する場合に採られる一つの方法として，シングルプロセッサ対応のカーネルを各プロセッサで動作させ，プロセッサ間の同期・通信機能が必要な場合はアプリケーションプログラムレベルで実現する方法がある[2]．

しかしながら，このアプローチは次に述べるような不都合が

あるため，我々は，MPSoPC を対称にシステムを構築するさいには MPSoPC 用の RTOS が必要不可欠であると考えている．

まず一つ目としてシームレスな環境ではないという問題がある．シームレスなマルチプロセッサ環境とは，他のプロセッサ上のタスクへの操作を，同じプロセッサにいるタスクへの操作との違いを意識することなく行える環境である．シームレスな環境を実現するためには，RTOS はマルチプロセッサ用に特別な API を持たせるのではなく，従来と同じ API で他プロセッサのタスクとプロセッサ間通信が行える機能を提供する．

二つ目の問題として，保守性の悪さが挙げられる．具体的には，システムに対する要求の変更などによりプログラムを含めたシステム全体の見直しに迫られ，タスクのプロセッサに対する割り付けが変更になった場合に，プログラムの多くの部分に改造が必要になる．これはプロセッサ内の同期・通信とプロセッサ間の同期・通信が，異なったインターフェイスで実現されていることに起因している．

この問題を解決するためには，プロセッサ間の通信は RTOS の機能を使い，どのプロセッサ上のタスクとでも同じインターフェイスで同期・通信ができるようにすればよい．RTOS でサポートすることで，システムの要求使用の変更などによりタスクの割り付け変更されても，プログラム修正の必要性を最小限に留めることが可能になる．

2.3.2 スケーラブルな RTOS モデル

リアルタイムシステムでは，プロセッサ数に対して最大実行時間または最大レスポンス時間のスケーラビリティを確保することが重要である．

これまでに，機能分散共有メモリマルチプロセッサ用の RTOS の，プロセッサ数に対してシステムコールの最大実行時間または最大レスポンス時間のスケーラビリティを重視した実装技術が，μITRON 仕様をベースに提案されている [1] [2]．また，一部のシステムコール（タスクの管理と付属同期機能の一部）のみではあるが，TOPPERS/JSP [3] [4] [5] をベースに実装評価も行われている．

[1] [2] で提案されたモデルでは，タスクをリアルタイム性能要求と共有資源の利用可能性に応じていくつかのクラス（プライベートクラス，ローカルクラス，グローバルクラス）に分類することで，競合するプロセッサ数に対するスケーラビリティの確保を行っている．また，タスク以外のカーネル資源，セマフォ，メイルボックスといった同期・通信オブジェクトについてもタスクからのアクセス可能性に応じて，クラス分類されている．

各プロセッサのローカルクラスのカーネル資源は他のプロセッサに対して公開することができ，他のプロセッサからこの公開資源にアクセスすることが可能である．言い換えると，ローカルタスクから他のプロセッサに属するローカルタスクに対してシステムコールが発行可能となる．

2.3.3 コンフィギュアラブルな RTOS

MPSoPC 用 RTOS は，MPSoPC の多様なシステムアーキテクチャに対応するためにコンフィギュアラブルであることが望ましい．例えば，複数のプロセッサにおいて，それらの上で実行されるタスクが共有メモリを介してタスク間通信をする場合，競合する可能性のあるタスクに対して RTOS は，整合性を欠かないように共有メモリ管理機能とさらにタスク間通信機能を提供する必要がある．また，データキューによりタスク間通信が行われる場合，それらのタスクが実行されるプロセッサ上の RTOS には，データキューによる通信機能が備わっていなければならない．またその一方で，それらの機能を必要としなければ RTOS の機能から削除するといったことが可能であることが望ましい．

3. Hardware/Software 一体構成技術

MPSoPC に RTOS を搭載する場合，各プロセッサバンク毎に(注1)その構成に合わせて RTOS をそれぞれでコンフィギュレーション(注2)を行う必要がある．また，カーネルの資源を外部に公開する場合には，各 PE の公開資源を管理する Class Control Block（CCB）への登録処理をターゲット依存ファイルに含め，競合する各プロセッサに排他制御のためのスピンロック機能を提供する必要がある．MPSoPC は，システムアーキテクチャに高い自由度を持つため，手作業による RTOS のコンフィギュレーションは大きな労力を要する．

本研究は，上記のコンフィギュレーション作業を自動化するツールの開発を目的とする．

本章では，本研究の位置付けをより明確にするため，MPSoPC とその RTOS のコンフィギュレーションを従来の手作業により行う場合の問題点を明らかにし，それを解決するために我々が提案する手法の詳細について述べる．

3.1 手作業によるコンフィギュレーションの問題点

一つ目の問題として，デバイスの割込み番号や，レジスタのアドレスといったパラメータのソフトウェアとハードウェア間での整合性の問題がある．これらのパラメータの設定は，内容自体にはあまり意味が持たずソフトウェアとハードウェア間での整合性が重要である．手作業によるコンフィギュレーションでは，ハードウェア側の構成を決めた後ソフトウェア側をそれに合わせることで整合性をとっている．

二つ目に，ペリフェラル IP のパラメータ設定個所が，ペリフェラル毎に異なり，設定がまとめて管理できないという問題がある．ペリフェラル IP は，使用目的に応じてパラメータを設定しなければならないが，その設定は，合成時にハードウェア記述として与える場合や，実行時にソフトウェアで設定する場合もあり，その設定個所はペリフェラル IP 毎に異なる．例えば，UART のペリフェラル IP を用いる場合，ボーレートの設定が必須であるが，このパラメータは UART の種類によって，ハードウェア合成時に指定するものとソフトウェアの初期化時に設定するものがある．

設計者は，各ペリフェラル IP のパラメータの記述場所とアドレスや割込み番号を把握する必要がある．その上パラメータの変更により，ソフトウェア側のどのファイルのどの部分を変

(注1)：PE とそのローカルバス，及び周辺ペリフェラルで構成された部分．
(注2)：主に周辺デバイスのアドレスや割込みの設定等を記したヘッダファイル等のターゲット依存ファイルや Makefile 等を作成する．

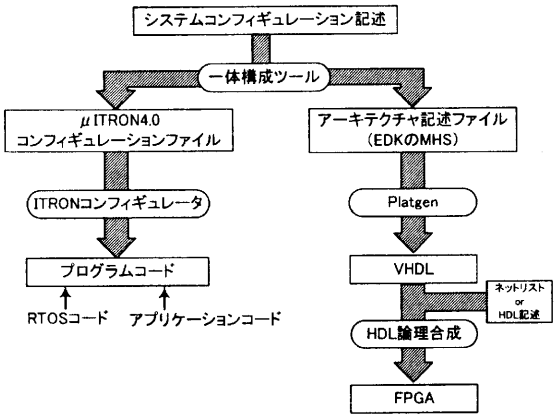

図 2　Hardware/Software 一体構成技術

更しなければならないかといったことも把握しておく必要がある．これらのことを一つ一つ確認しながら設定することは，設計者にとって非常に煩雑な作業となる．

三つ目の問題としては，図 1 のようにソフトウェアの構成に合わせて最適なシステムアーキテクチャを生成する場合には，一つ目の問題とは逆にソフトウェアの構成情報が必要になることである．一つ目と三つ目のこの問題は，言い換えるとハードウェアとソフトウェアの構成における依存性の問題といえる．

3.2　問題解決へのアプローチ

前述の問題を解決するため，ソフトウェアとハードウェアの構成を予め定義した記述方法に従って一箇所に記述し（システムコンフィギュレーション記述と呼ぶ），ツール（システムコンフィギュレータと呼ぶ）によりハードウェア，ソフトウェアのコンフィギュレーションを同時に行う方法をとる．つまり，システムコンフィギュレータ側でハードウェア，ソフトウェア構成の依存関係を吸収して解消を行う．

我々はこの手法を，Hardware/Software 一体構成技術と呼ぶ．図 2 に一体構成技術のフローを示す．図の右側のフローは EDK [6] によるシステムアーキテクチャの生成フローである．左側のフローは RTOS のコンフィギュレーションを示したものである．

Hardware/Software 一体構成技術により，手作業によるコンフィギュレーションの問題は次のように解決される．

一つ目の問題に対しては，設計者がデバイスレジスタのアドレスや割込み番号を知らなくても困ることはなく，ハードウェアと RTOS 側で一貫していればよい．そこでこれらを自動的に割付け両方同時に設定すれば作業が軽減されるといえる．

二つ目のパラメータの設定個所が異なる問題は，システムコンフィギュレータがソフトウェア，ハードウェアのどちら側で設定すればよいのかを判断することで解決される．

三つ目のソフトウェアとハードウェアの構成における依存性の問題は，システムコンフィギュレーション記述よりハードウェア，ソフトウェアのコンフィギュレーションファイル（アーキテクチャ記述ファイルと RTOS コンフィギュレーションファイル）を同時に生成することで解決される．

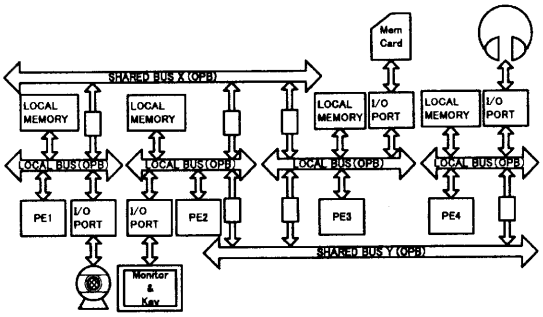

図 3　システムアーキテクチャの構成

4.　実　　装

本章では，前節で述べた要件を満たすシステムコンフィギュレータ及び MPSoPC 用の RTOS の実装について述べる．

4.1　実現範囲

本研究では，システムコンフィギュレータと MPSoPC の持つ機能のうち，基本的なものを実装する．MPSoPC については，分散共有メモリとスピンロックを用いたプロセス間通信のみを実装し，RTOS の機能をサポートする専用ハードウェアを用いる手法については今後の課題とする．

一体構成技術の適用範囲としては，図 3 に示すようなマルチバスの分散共有メモリマルチプロセッサを対象とした．

図 3 の各プロセッサバンクは，ローカルバスと他の PE からもアクセス可能なローカルメモリなどで構成される．各プロセッサバンクは，バスブリッジを介して共有バスに接続され，同じ共有バスに接続されているプロセッサバンクにアクセスできる．

PE1 からは PE2 と PE3 の公開資源にアクセス可能で，システムコールを PE2 と PE3 に対して発行できるようになっている．PE2 と PE3 は全てのプロセッサの公開されている資源に対してアクセスができる．同様に，PE4 は PE2 と PE3 の公開資源にアクセス可能である．

同じ共有バスに接続されている PE 群を，ドメインという単位で扱う．カーネル資源の公開はこのドメイン内で行われる．図 3 では，PE1，PE2 及び PE3 が一つのドメインを形成する．また，PE2 と PE3 は，PE4 とのドメインにも属する．

4.2　ハードウェアプラットフォーム

本研究では，MPSoPC のプラットフォームとして Xilinx の Micorblaze システム [6] を用いる．また，Microblaze システムをマルチプロセッサ構成として動作させるための評価ボードを開発した．

4.2.1　Microblaze システム

Microblaze システムでは，Microblaze と呼ばれるソフトプロセッサ及びそのバスとペリフェラル IP が，HDL もしくはネットリストで提供されている．Microblaze プロセッサは，32bit の RISC プロセッサで，ハーバードアーキテクチャを採用している．また FPGA 用に最適化され非常にコンパクトであり，中規模以上の FPGA では複数個実装することが可能で

```
# Common Global Port
PORT sys_clk = sys_clk, DIR = IN, SIGIS = CLK
PORT sys_reset = sys_reset, DIR = IN

# Sub Components
BEGIN microblaze
 PARAMETER INSTANCE = pe1_Microblaze
 PARAMETER HW_VER = 2.00.a
 PORT CLK = sys_clk
 PORT INTERRUPT = pe1_interrupt
 BUS_INTERFACE DLMB = pe1_d_lmb
 BUS_INTERFACE ILMB = pe1_i_lmb
 BUS_INTERFACE DOPB = pe1_opb
 BUS_INTERFACE IOPB = pe1_opb
END

BEGIN opb_v20
 PARAMETER INSTANCE = pe1_opb
 PARAMETER HW_VER = 1.10.b
 PARAMETER C_BASEADDR = 0xFFFFFFFF
 PARAMETER C_HIGHADDR = 0x00000000
 PORT OPB_Clk = sys_clk
 PORT SYS_Rst = sys_reset
END

BEGIN opb_intc
 PARAMETER INSTANCE = pe1_opb_intc
 PARAMETER HW_VER = 1.00.c
 PARAMETER C_BASEADDR = 0xF1001000
 PARAMETER C_HIGHADDR = 0xF10010FF
 PORT Intr = pe1_opb_timer_Interrupt
 PORT Irq = pe1_interrupt
 BUS_INTERFACE SOPB = pe1_opb
END

BEGIN opb_timer
 PARAMETER INSTANCE = pe1_opb_timer
 PARAMETER HW_VER = 1.00.b
 PARAMETER C_BASEADDR = 0xF1000000
 PARAMETER C_HIGHADDR = 0xF10000FF
 PORT Interrupt = pe1_opb_timer_Interrupt
 BUS_INTERFACE SOPB = pe1_opb
END

BEGIN bram_block
 PARAMETER INSTANCE = pe1_bram_block
 PARAMETER HW_VER = 1.00.a
 BUS_INTERFACE PORTA = pe1_ilmb_porta
 BUS_INTERFACE PORTB = pe1_dlmb_portb
END
BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = pe1_ilbm
 PARAMETER HW_VER = 1.00.b
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x00007fff
 BUS_INTERFACE SLMB = pe1_i_lmb
 BUS_INTERFACE BRAM_PORT =
pe1_ilmb_porta
END

BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = pe1_dslmb
 PARAMETER HW_VER = 1.00.b
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x00007fff
 BUS_INTERFACE SLMB = pe1_d_lmb
 BUS_INTERFACE BRAM_PORT =
pe1_dlmb_portb
END
```

図 4　MHS の記述例

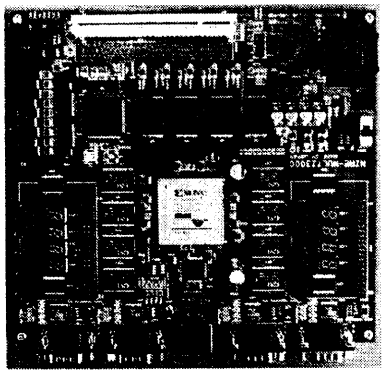

図 5　開発した FPGA ボードの外観

ある．例えば，4 つのプロセッサ及びその周辺ペリフェラルを VirtexII-3000 に実装した場合の使用ゲート数は全体の 30% 程度である．

ソフトプロセッサとペリフェラル IP を用いたシステムの設計環境として Embedded Development Kit（EDK）と呼ばれる環境が用意されている．

まず，システムアーキテクチャを MicroBlaze Hardware Specification（MHS）（図 4）に記述する．次に，この記述をプラットフォームジェネレータ（Platgen）が読込み，ソフトプロセッサとペリフェラル IP を接続する VHDL 記述を生成する．この VHDL 記述とソフトプロセッサ及びペリフェラル IP のネットリストもしくは HDL 記述を合成することにより FPGA 上に実装する．また，EDK にはユーザが設計したペリフェラル IP を容易に扱う仕組みが提供されている．

4.2.2　マルチプロセッサ SoPC 用評価ボード

本研究の評価を行うためのマルチプロセッサシステム構成が可能な評価ボードとしては，外部メモリを複数バンク備えたものが望ましい．しかしこの要件を満たす適当な市販ボードが見当たらなかったため，MPSoPC 用の，新たな FPGA ボードの

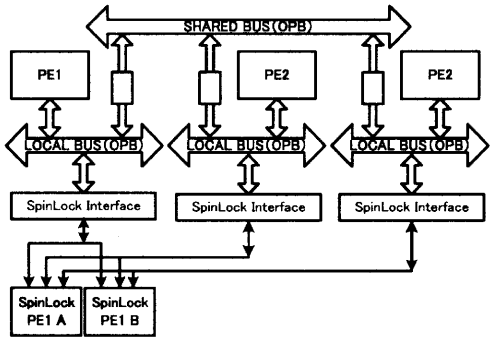

図 6　スピンロックハードウェアの接続

開発を行った（図 5）．このボードの使用を以下に示す．

- 300 万ゲート相当の FPGA（Xilinx VirtexII XC2V3000）
- 独立した SRAM バンク（1MB）× 4
- フラッシュメモリ（16MB）
- シリアルポート（RS-232C）× 8 ポート
- Ethernet（拡張ボードにより最大 3 ポート）

FPGA には少なくとも 4 つの Microblaze プロセッサが搭載可能である．これにより SRAM，フラッシュROM，シリアルポート 2 ポートをリソースとするプロセッサバンクを 4 個実現可能である．

4.2.3　公開資源の排他制御

公開資源にアクセスする場合のプロセッサ間の排他制御は，スピンロックにより実現する．リアルタイムシステムでは，ロックを取れるまでの時間に上限が与えられなければならない．上限を与えるスピンロックアルゴリズムとしては，ロック待ち状態のプロセッサをキューによって管理するものがあるが，共有メモリにアクセスする必要がある(注3)ため，共有バスの競合により予測可能性が下がる．

そこで，MPSoPC の特徴を活かしこれを回避するためのスピンロックハードウェアを作成し MPSoPC に実装した．テスト・アンド・セット方式をハードウェア化したもので，一回のリードで，リード・モディファイ・ライトが実現できるようになっている．

各スピンロックハードウェアは作成した SpinLock-Interface 回路を介して各 PE のローカルバスに接続するため，スピン時に共有バスの使用効率の低下を招ねくことはない．

一つのロック単位につき一つのスピンロックハードウェアにより排他制御される．[1] では，カーネル資源のアクセスパターンに応じて 2 つのロック単位を定義しているため，各 PE は 2 つのスピンロックハードウェアを所有する．

なお，カーネル資源が公開された場合には，一体構成技術により自動的にスピンロックハードウェアの生成を行う．

図 6 に，PE1 のカーネル資源を PE2，PE3 に公開した場合のスピンロックハードウェアの接続例を示す．この図では，PE2，PE3 のカーネル資源は公開しないため，スピンロックハード

(注3)：キューに繋ぐ時と，ロック待ち状態の次のプロセッサにロックを渡すとき

```
Processor PE1 {
        //PE1の資源を記述
        CRE_TSK          //μITRONの静的API
        CRE_SEM
        CRE_UARTLITE  //ペリフェラル使用記述
}
Processor PE2 {
        //PE2の資源を記述
        .....
}
Processor PE3 {
        //PE3の資源を記述
        .....
}
Bus X{
        //バスに接続するプロセッサを記述
        PE1
        PE2
}
Bus Y{
        PE2
        PE3
}
```

図 7　システムコンフィギュレーション記述定義

ウェアの生成は行われない.

4.2.4　プロセッサ間割込み

他の PE に対してシステムコールを発行したことにより，その PE において再スケジューリングが必要になった場合，プロセッサ間割込みによりこれを通知する.

しかし，Microblaze はプロセッサ間割込みの機能を持っていないため，これを実現するためのハードウェアを新たに作成した.

4.3　マルチプロセッサ SoPC 対応 RTOS の実装

2.3 節に挙げた要件に基づいて，TOPPERS カーネルをベースにマルチプロセッサ SoPC 対応 RTOS の作成を行った. [3] で開発されたコードを参考に残りのシステムコール，主にタスクの同期・通信機能を，他の PE 上のタスクに対しても発行できるように拡張した.

クラスについては, [1] で定義されたローカルクラスのみを実装する．以下に，ローカルクラスの特徴を示す.

- 各プロセッサに一つ存在
- 他のプロセッサ上のカーネル資源にもアクセス可能

ここで他のプロセッサのカーネル資源とは，各プロセッサにおいて公開された資源であり，タスク管理オブジェクト，同期・通信オブジェクトを指す．すなわち，ローカルタスクから他のプロセッサに属するローカルタスクに対してシステムコールが発行可能となる．各プロセッサの公開資源は CCB テーブルで管理され，この情報を基に公開資源にアクセスする．また，CCB テーブルは各プロセッサのローカルメモリに保持される.

また，同期・通信機能などではスピンロック取得の関係により，デッドロックを引き起こす可能性があるため, [7] で提案されている手法に基づいてこれを回避した.

4.4　システムコンフィギュレーション記述

OS を含めた図 3 のシステムを，一体に構成するためのシステムコンフィギュレーション記述案を示す（図 7）.

図 7 の"Processor {}" で囲まれた部分には，OS を含めたプロセッサバンクのシステム構成を記述する．具体的には，PE が所有する資源（タスク管理オブジェクト，同期・通信オブジェクトなどのカーネル資源及びハードウェア資源）を記述する.

"Bus {}" で囲まれた部分では，共有バスに接続する PE を記述する．ここにより，ドメインが形成される.

4.5　システムコンフィギュレータの作成

現在，図 7 の記述からアーキテクチャ記述ファイル（MHS ファイル）と，各 PE の OS のコンフィギュレーションファイルを生成するためのシステムコンフィギュレータを開発中である.

現在の問題点としては，PE のカーネル資源を公開するための記述が未定義であるということである．資源を公開する場合に考慮しなければならない項目としては，同じドメイン内のどの PE に対して公開するのかといった状況を考える必要がある．現時点では，暫定的に全てのカーネル資源を全 PE に公開するものとして実装を行っている.

次の作業として，システムコンフィギュレータを完成させ，図 7 の記述から一体構成可能か検証し，システムコンフィギュレーション記述定義の不十分な点を含め必要な記述定義を行っていく.

5.　ま　と　め

本稿では，MPSoPC の特徴について述べ，MPSoPC 用 RTOS の必要性とその RTOS に求められる要件を明らかにした．また，MPSoPC と RTOS の手作業によるコンフィギュレーションに伴う問題点を指摘し，この問題を解決するための方法である Hardware /Software 一体構成技術ついて述べた.

これまでの成果としては，MPSoPC 用の RTOS を開発し，現在，システムコンフィギュレータの実装を行っている.

今後の課題としては，システムコンフィギュレータを完成させるとともに，現在，一体構成可能なシステムアーキテクチャに限定を与えているため，可能な限り MPSoPC の多様性に対応させていく.

文　献

[1] 高田広章, 坂村健, "非対称マルチプロセッサシステムのためのスケーラブルなリアルタイムカーネルの構想," 信学技報（1995 年実時間処理に関するワークショップ RTP'95）,vol.94,no.573,pp.1-8, 電子情報通信学会,Mar.1995

[2] 高田広章, 坂村健, "マルチプロセッサリアルタイムカーネルのスケーラビリティを重視した実現," 信学技報（1996 年実時間処理に関するワークショップ RTP'96）, vol.95,no.603,pp.1-6,Mar.1996

[3] 蛸井基明, 宮内新, 石川知雄, 高田広章, "マルチプロセッサ環境におけるマイグレート可能タスクの導入," （2002 年実時間処理に関するワークショップ RTP 2002）,vol.2002,pp.59-66,Mar.2002

[4] TOPPERS プロジェクト, http://www.toppers.jp

[5] 坂村健監修, 高田広章編, "μITRON4.0 仕様 Ver. 4.00.00," トロン協会,（1999）.

[6] Xilinx, http://www.xilinx.com

[7] 辰巳将司, 石川知雄, 宮内新, 高田広章, "ITRON 仕様 OS のマルチプロセッサ拡張におけるデッドロック回避の手法," （2001 年実時間処理に関するワークショップ RTP 2001）,vol.2001,no21,pp.115-122,Mar.2001