Vol.2012-SLDM-155 No.28
Vol.2012-EMB-24 No.28
2012/3/3

FPGA 上で動作可能なマルチプロセッサ RTOS の提案

古谷　拓之[†1]　　北道　淳司[†1]

近年, ワンチップ上で複数プロセッサを実装するマルチプロセッサは組み込みシステムの分野でも注目を集めている. 本論文では, 様々なマルチプロセッサ構成が実装可能な Field Programmable Gate Array(FPGA) に着目し, マルチプロセッサに対応している Real Time Operating System（RTOS）の 1 つである TOPPERS/FMP を FPGA 上の多様なシステム構成に対応させる方法を提案する. 我々は Altera 社の FPGA および NIOS II プロセッサを採用し, 設計者が構築したさまざまなシステム構成に対応するような,TOPPERS/FMP におけるデバイス依存部分を設計し, 開発している.

Porposal of Real Time OS for Multi-processor on FPGA

Hiroyuki Kotani[†1] and Junji Kitamichi[†1]

Recently, multi-core processors which consist of several processors on one chip are featured at the embedded field. In this paper, we propose a Real Time Operating System(RTOS) for multi-core processor architecture on Field Programmable Gate Array (FPGA), where we can implement various system construction. We adopt TOPPERS/FMP kernel, which is a RTOS, as a core kernel, and propose the method which enables TOPPERS/FMP to operate for various system construction on FPGA. We adopt Altera FPGA and NIOS II processor, and design and are implementing the device-dependent modules for TOPPERS/FMP to enable the correspondence to the various designed system construction on FPGA.

†1 会津大学コンピュータ理工学研究科
The University of Aizu, Graduate School of Computer Science and Engineering

1. はじめに

近年, 1 つのチップ上に複数のプロセッサを実装するマルチプロセッサは組込みシステムの分野においても注目を集めている[1). 現在の VLSI 製造技術を用いた場合, 処理性能の向上のためにはクロック周波数を上げるよりもプロセッサ数を増やした方が有利であることがあげられる. 組み込みシステムにおいて, アプリケーションの複雑化により, 高い処理性能が必要であり, マルチプロセッサは処理性能, さらに消費電力の面からも利点が大きく, 広域な組込みシステムへの適応が期待される. マルチプロセッサの構成としては, 対称型あるいは機能分散型, 密結合型あるいは疎結合型など多くの分類がある. 組み込みシステムは, 汎用の計算機システムに比べると, 機能があらかじめ決まっていて専用化されていることが多い. 組み込みシステムの分野では, 機能分散型の密結合マルチプロセッサが用いられることが多い. マルチプロセッサの CPU リソースを効率的に運用するという意味で, マルチプロセッサに対する OS の役割は極めて大きい. マルチプロセッサに対応した OS としては,Linux(Andoroid), Windows Embedded, ITRON 仕様を実装した OS などが開発されている.

我々は, 様々な構成のマルチプロセッサを用いて, システム実装を行いシステムパフォーマンスを評価する環境を整えたいと考えている. それらの環境はマルチプロセッサあるいはオペレーティングシステムに関する研究に利用されるだけではなく, 大学あるいは大学院におけるマルチプロセッサ, プロセッサシステム, オペレーティングシステムおよび並列プログラミングなどの演習あるいは実験において利用することも考えられる. 様々なマルチプロセッサシステムを構築するためにデバイスとして,Field Programmable Gate Array(FPGA) を採用する. 本研究では,Altera 社の FPGA およびソフトウエアコア NIOS II[2)] プロセッサを採用する.Altera 社が提供する開発環境 QuartusII, SOPC builder および NIOS II-IDE を用いることにより,FPGA 上の柔軟なシステム構成からソフトウエア開発までが実現できる. そのシステム上で動作する RTOS のコア部分として, オープンソースであり Real Time Operating System(RTOS) の 1 つである TOPPERS/FMP を採用する. 現在公開され利用できる NIOS II 用の TOPPERS/FMP 環境は,Altera 社の FPGA を搭載した特定の FPGA ボードにのみ対応しており, それ以外の FPGA ボードに対応させるには, デバイス依存部分を再構築する必要がある. 本研究では, このデバイス依存部分を, 設計者が構築したさまざまなシステム構成に対応するため, 容易にパラメータ変更が可能なデバイス依存部分を提案する.

以下 2 章では μ ITRON 仕様と TOPPERS/FMP, 3 章では提案手法および今後の課題に

Vol.2012-SLDM-155 No.28
Vol.2012-EMB-24 No.28
2012/3/3

ついて述べ, 4 章において結論を述べる.

2. μ ITRON 仕様と TOPPERS/FMP

μ ITRON は, 組み込みシステムのためのリアルタイムオペレーティングシステムの仕様であり,ITRON プロジェクトにより策定されている[3]. 現在μ ITRON4.0(Ver.4.02.00) 仕様が公開されている. μ ITRON4.0 仕様は, スタンダードプロファイル, 自動車制御プロファイルなどがあり, 以下ではスタンダードプロファイルについてのみ議論する.

スタンダードプロファイルは標準的な機能セットであり, 各実装に最低限の互換性を持たせることを試みている. スタンダードプロファイルにおいては, システム全体は 1 つのモジュールにリンクされ, カーネルオブジェクトはすべて静的 API として生成される. システム全体が 1 つのモジュールにリンクされるため, システムコールはサブルーチンコールとして呼び出される. また, プロテクションの機能はない.

Toyohashi OPen Platform for Embedded Real-time Systems(TOPPERS) はμ ITRON 仕様にもとづいた実装の 1 つである. TOPPERS は TOPPERS プロジェクトにより開発されている一連の RTOS の総称である.TOPPERS カーネルは, シングルプロセッサ対応 RTOS の TOPPERS/Advanced Standard Profile(ASP) カーネルを基本として, メモリ保護, マルチプロセッサ, カーネルオブジェクトの動的生成, 機能安全などに対応している.TOPPERS の仕様は, 文献[4](以下, 統合仕様書) としてまとめられている.

Flexible Multiprocessor Profile(FMP) カーネルは,ASP カーネルを拡張することにより開発されたマルチプロセッサのための RTOS であり, 対称型 (以下 SMP 型) システムにおける動的なタスクの移動が可能という利点と, 機能分散型 (以下 AMP 型) システムにおけるリアルタイム性を保証しやすいという両方の特徴を目指している[5],[6]. FMP カーネルは, 基本的にはシステム構成時に設計者がタスクをプロセッサに割り当て, タスクを実行するプロセッサを静的に決定する. 図 1 に FMP カーネル, プロセッサおよびタスクの動きの概要を示す. FMP カーネルでは, プロセッサごとにレディキューが存在している. レディキューは個別に管理されており, 各プロセッサがプロセッサに閉じた処理を実行している限りは他のプロセッサの影響を受けることはないため,OS 内での排他制御が必要とない. そのためシステムコール発行時の平均的なスループットが向上する. また, プロセッサごとに, プリエンプティブな優先度ベースのスケジュールを行うので, シングルプロセッサの場合と同様のリアルタイム性を確保することができる. 各プロセッサのレディキューにあるタスクを別のプロセッサのレディキューに移動させる API を提供することにより, リアルタイム性を確保するための動的なロードバランスを実現しやすい. 図 1 では, プロセッサ 1 のレディキューからプロセッサ 2 のレディキューへの Task12 の移動を表している.

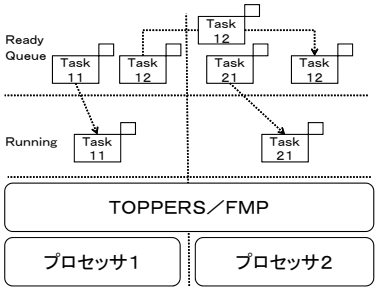

図 1 TOPPERS/FMP カーネル

FMP カーネルが提供しているマルチプロセッサに特有の API について述べる.

- プロセッサ指定のタスク起動機能
  - mact_tsk(ID tskid, ID prcid)/imact_tsk(ID tskid, ID prcid)

プロセッサ指定のタスク起動機能 mact_tsk は, tskid で指定したタスクを prcid で指定したプロセッサで起動するためのシステムコールである.

- タスクマイグレーション機能
  - mig_tsk(ID tskid, ID prcid)

マイグレーション機能 mig_tsk は, tskid で指定したタスクを prcid で指定したプロセッサに移動させるシステムコールである. この機能は発行したプロセッサに割り付けられたタスクに対してのみ有効である. また, 非タスクコンテキストからはこの API は発行できない. これは, システムコールの最悪実行時間あるいは平均実行時間を抑えるための制約である.

ASP カーネルにおけるタスクの状態遷移図を 2 に示す. ASP カーネルには, カーネルがディスパッチを行わないディスパッチ保留状態があり, ディスパッチ保留状態のタスクは, 強制待ち状態に遷移しない. ディスパッチ保留状態のタスクは, ディスパッチ保留解除によって実行状態に戻る. その他のタスクの動作については省略する.

FMP カーネル上で動作するタスクは, 異なるプロセッサに割り付けられているタスクに対する操作が可能であり, ディスパッチ保留状態のタスクに対しても操作可能である. その

Vol.2012-SLDM-155 No.28
Vol.2012-EMB-24 No.28
2012/3/3

ため, ディスパッチ保留状態のタスクに対して, 別のプロセッサにあるタスクから強制待ち状態へ遷移させる API を呼び出した場合, 実行状態 (ディスパッチ状態) であったタスクはディスパッチも保留される. この間, カーネル管理上のタスクの状態は強制待ちとなる.FMP カーネルではこのような過渡的な状態が有り得るため強制待ち状態 [実行継続中] が追加されている. 強制待ち状態 [実行継続中] のタスクに対して再開の場合は, 実行状態 (ディスパッチ状態) に遷移し, ディスパッチ保留解除の場合は, 強制待ち状態に遷移する. FMP におけるタスクの状態遷移図を 3 に示す.

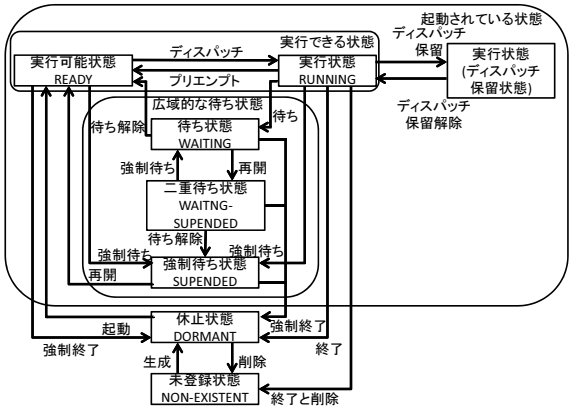

図 2 TOPPERS/ASP におけるタスクの状態遷移図

TOPPERS/FMP(Ver.1.2) は,ARM,NIOS II, SH2a,SH4a のマルチプロセッサに対応している. OS は, 共通のカーネル部分と, 個々のプロセッサに依存するデバイス依存部分からなる.

以下では TOPPERS/FMP における NIOS II プロセッサに関するデバイス依存部分について述べる.

NIOS II に関するデバイス依存部分は, nios2_dev_1s40 と nios2_dev_3c25 と nios2_dev_3c120 という 3 種の設定が用意されており, それぞれ 3 種の FPGA ボード上へのシステム実装がサポートされている. 設計を行う際に, 設計者はそのなかから 1 つを選択する.

TOPPERS/FMP においてリソースに対する相互排除を実現するためのセマフォは Altera 社の SOPC builder で用いることができるライブラリのなかから Mutex を用い, プロセッサから他のプロセッサへの割り込みを用いた通信を実装するために Altera 社の CAD 環境で使用することができる割り込み用のハードウエアモジュール *prc_int* が提供されている. また, システムバージョン回路 *sysver* が提供されており, システムに組み込む必要がある.

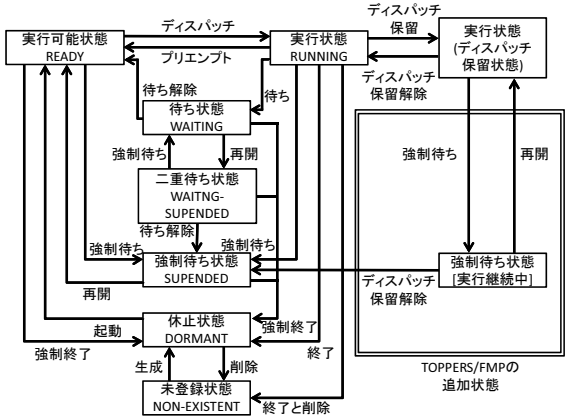

図 3 TOPPERS/FMP におけるタスクの状態遷移図

TOPPERS/FMP におけるデバイス依存部分は, 設定ごとに用いるプロセッサ数が固定されており, いくつかのデバイス用アドレスが固定されている. 例えば,nios2_dev_c120 の設定を使用した場合, 第 1 プロセッサの JTAG Debug Module の Base Address は 0x01000900 が指定されており, 第 2 プロセッサには 0x02000900 が指定されているなど, 使用するプロセッサなどハードウエアモジュールのパラメータに対する設定を確認する必要がある. 表 1 に, 主要なアドレスマップの一部を示す. nios2_dev_c120 の設定では, プロセッサ数は 4 と指定されているためプロセッサ数を 5 以上にする場合, 設計者は 5 番目以降のプロセッサに関するアドレスマップを追加し, デバイス依存部分の記述を追加する必要がある.

システムは,Nios II プロセッサ CPU1, CPU2, CPU1 のメモリ MEM1, CPU2 メモリ MEM2, および命令とデータを格納したメモリ Inst & data memory, 排他制御のための Mutex, 割り込みを実行するための *prc_int* , システムバージョン回路 *sysver* および timer 回路によって構成されている. 構成の概要を図 4 に示す.

Mutex は 2 つあり,TSK_Mutex と OBJ_Mutex である. プロセッサ 1 での *prc_int* のベースアドレスは 0x01000c00 を, TSK_Mutex のベースアドレスは番地 0x01000a00,OBJ_Mutex のベースアドレスは番地 0x01000b00 を用いる. 同様に, プロセッサ 2 では *prc_int* のアドレス

Vol.2012-SLDM-155 No.28
Vol.2012-EMB-24 No.28
2012/3/3

name	address map
JTAG_UART_1_BASE	0x01000900
TSK_MUTEX_1_BASE	0x01000a00
OBJ_MUTEX_1_BASE	0x01000b00
PRC_INT_1_BASE	0x01000c00
JTAG_UART_2_BASE	0x02000900
TSK_MUTEX_2_BASE	0x02000a00
OBJ_MUTEX_2_BASE	0x02000b00
PRC_INT_2_BASE	0x02000c00
JTAG_UART_3_BASE	0x03000900
TSK_MUTEX_3_BASE	0x03000a00
OBJ_MUTEX_3_BASE	0x03000b00
PRC_INT_3_BASE	0x03000c00
JTAG_UART_4_BASE	0x04000900
TSK_MUTEX_4_BASE	0x04000a00
OBJ_MUTEX_4_BASE	0x04000b00
PRC_INT_4_BASE	0x04000c00

表 1 nios2_dev_c120 において用いられるアドレスマップの例

マップは番地 0x02000c00, TSK_Mutex のアドレスマップは番地 0x02000a00,OBJ_Mutex のアドレスマップは番地 0x02000b00 を用いる.

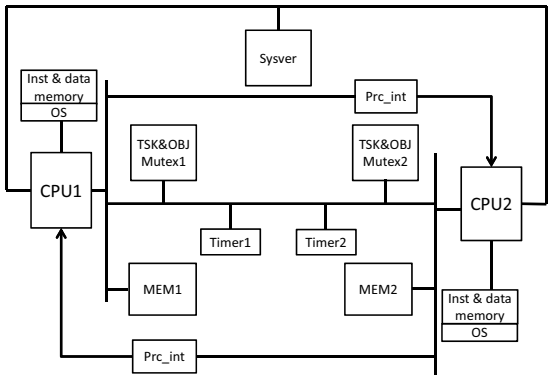

図 4 TOPPERS/FMP を用いたシステム構成の例

3. 提案する Real Time Operating System

我々は, 様々な構成のマルチプロセッサを用いて, システム実装を行いシステムパフォーマンスを評価するために, ターゲットデバイスとして,Altera 社の FPGA およびソフトウエアコア NIOS II プロセッサを採用する. また,RTOS のコア部分については TOPPERS/FMP カーネルを採用する.

Altera 社には多くの種類の FPGA デバイスおよび FPGA ボードが存在するが, 提案する OS は, ハードウエア的な制約をなるべく排除し, 多くのデバイスおよびボードで動作することを目的とする. デバイス依存部分を設計者が再構築する必要があるときも容易にパラメータ変更が可能あるいは自動的に設定可能なようにデバイス依存部分を実現し, カーネル部分の変更も最小限に, もしくはそのまま用いることにする. 目標とする項目は以下のとおりである.

(1)FPGA の種類, ボードの種類の廃止およびパラメータ設定の自動化
(2) ハードウエアモジュールに対する API の充実
(3) マルチプロセッサのためのシステム構成のためのハードウエアモジュールの追加
(4)FMP からのいくつかのリソース管理ポリシーの変更
(5)TOPPERS/FMP 以外の ITRON 仕様準拠 OS への対応

以下, 各項目について述べる.

(1)FPGA の種類, ボードの種類の廃止およびパラメータ設定の自動化

提案 OS では,SOPC builder によって生成された構成情報を元に, 設定するパラメータの妥当性をチェックし, 各種パラメータ設定をなるべく自動化する.

TOPPERS/FMP におけるデバイス依存部分の定義では, nios2_dev_1s40,nios2_dev_3c25, および nios2_dev_3c120 の 3 種類の特定のボードにのみに対応しているため設計者が自由に様々な構成を試すには, 各種パラメータ設定を変更あるいは追加する必要がある.

提案 OS では, 各ボードに依存している部分をまとめることにより, FPGA あるいはボードを特定せず, 設計者が自由に様々な構成を設定し試すことができるようにする. 例えば, デバイス依存部分の nios2_system.h 内に定義されている, プロセッサ数の定義では nios2_dev_1s40,nios2_dev_3c25, および nios2_dev_3c120 がすべて同じ変数名（TNUM_PRCID, 値は 2）で定義されており, 設計者がプロセッサ数を変更する仕様となっている.

各ボードにおいて同じ変数名でユーザが任意に指定する値を持つものを一つにまとめ設

Vol.2012-SLDM-155 No.28
Vol.2012-EMB-24 No.28
2012/3/3

定を簡略化する．例えば，タイマ値の内部表現とミリ表単位との変換（TIMER_CLOCK 50000U），微小時間待ちのための定義（SIL_DLY_TIM1 230, SIL_DLY_TIM2 70），ネイティブスピン方式の場合のスピンロックの最大数（TMAX_NATIVE_SPN 4），データセクションの初期化を行わない（NIOS2_OMIT_DATA_INIT），Interrupt Vector Instruction 命令を待つ（NIOS2_USE_INT_VEC_INST, NIOS2_INT_VEC_INST_NO 0），キャッシュサイズ（NIOS2 _ICACHE_SIZE 0x1000, NIOS2_ICACHE_LINE_SIZE 0x0010, NIOS2_DCACHE_SIZE 0x0000, NIOS2_DCACHE_LINE_SIZE），プロセッサ間割り込み HW（PRC_INT_INT 2U），Performance Counter のクロック（PERF_COUNTER_CLOCK 50U），の 8 個の宣言を定義をまとめる．別の値に定義する必要がある場合は，別途定義することもことができる．これらの定義をまとめることにより，ボードの変更時にそれぞれのボードで個別に定義している部分の共有部分を再定義する必要がないので，設計者の負担が減少する．

プロセッサごとに設定が必要となる非タスクコンテキスト用のスタックサイズの定義（DEFAULT_PRC_ISTKSZ），ペリフェラルのベースアドレス（CPU_PERI_BASEADDR），Interval Timer（SYS_CLK_TIMER_BASE, SYS_CLK_TIMER_INT），JTAG UART（JTAG_UART_BASE, JTAG_UART_INT），Mutex（TSK_MUTEX_BASE, OBJ_ MUTEX_BASE），プロセッサ間割り込み HW（PRC_INT_BASE），をカーネルにおけるプロセッサごとの個別定義名での呼び出しではなく，パラメータをもつ共有定義名での呼び出しとする．ユーザが 4 つ以上のプロセッサを使用するときに従来の TOPPERS/FMP ではユーザ側で 5 つ目以降のプロセッサのためのデバイス依存部分についての記述を行わなければならなかったがこのように変更することによってユーザが 5 つ目以降のプロセッサのためにデバイス依存部分について記述する必要がなくなり，ユーザの負担を減少することができる．例えば，FMP のデバイス依存部分では CPU のペリフェラルのベースアドレスは

#define CPU_1_PERI_BASEADDR 0x01000000
#define CPU_2_PERI_BASEADDR 0x02000000
:

のように定義されているが，これを

#define CPU_PERI_BASEADDR (N) (0x01000000 * N)

と CPU 番号を表す N を用いてパラメータ化する．

(2) ハードウエアモジュールに対する API の充実

NIOS II のデバイス依存部分には，タイマ，*prc_int* などのハードウエアモジュールに対する API が用意されている．例えばタイマに対しては，タイマの起動，停止およびハンドラ登録の API が用意されている．タイマレジスタの再起動，取得あるいは設定など実装されていないが，これらの有用と思われる API を実現し拡充する．

また，SOPC Builder で使用可能な DMA controller，PIO などのハードウエアモジュールを利用可能にするための API を開発する．例えば PIO は，FPGA ボード上の LED の点滅あるいはキー入力の受付などに利用できる．

(3) マルチプロセッサのためのシステム構成のためのハードウエアモジュールの追加

TOPPERS/FMP を用いてプロセッサ間の割り込みによる通信を実現するために，提供されている割り込み用のハードウエアモジュール *prc_int* を用いる必要がある．各プロセッサ間の任意の通信を実現するには，プロセッサ数を N とすると *prc_int* は N*(N-1) 必要となり，現実的ではない．複数のプロセッサ間の割り込みを実現する割り込み通信モジュールを実現し，提供する．

NiosII プロセッサには多くのパラメータがある．例えば MMU などへの対応を行う．デバイス依存部分をこれらの機能に対応させる．パラメータ設定については，SOPC Builder でシステムを構成して得られる構成情報を解析し，デバイス依存部分から利用できるようにする．

(4)FMP からのいくつかのタスク管理ポリシーの変更

AMP 型のシステム構成の場合，各プロセッサが共通に接続されるバスに Mutex を配置すると，ロックしたプロセッサと別のプロセッサのタスクがロック解除できるように，複数のプロセッサ間で正しい排他制御が実現できない．正しい排他制御を行うために，Mutex にロックを行った CPUID を保持するようにハードウエアモジュールを拡張し，セマフォのロックおよび解除のための API の内部において，CPUID をハードウエアモジュールに保持させ，解除の許可を判断させることによって他のプロセッサにあるタスクからの排他制御を正しく行えるようになる．

現在の TOPPERS/FMP において，プロセッサが起動する際に，すべてのプロセッサが起動することを保証するために同期をとってからプログラムが開始される．AMP 型のシステム構成の場合，あるプロセッサが先行してタスクを起動させたいような場合，この条件は不要と考えられ，システム起動時間を向上させることができる．提案 OS では，プロセッサ起動時の同期をとらないことにする．その場合例えば，先に起動したプロセッサ上のタスクが，まだ起動していない他のプロセッサに割り込みを送信した時，割り込みが有効にならない．そのため送信元のカーネルにステータスを持たせて起動するまで待機をさせる，割り込みをかけるが動作の確認を行わない，割り込みの反応があるまで数回割り込みを発生させるなどの

Vol.2012-SLDM-155 No.28
Vol.2012-EMB-24 No.28
2012/3/3

処理行わせるように各種 API の機能を変更する.

(5)TOPPERS/FMP 以外の ITRON 仕様準拠 OS への対応

ITRON 仕様に準拠する OS として,T-Engine が開発されている.T-Engine はハードウエア依存部を持たない. 今回提案する TOPPERS/FMP のためのデバイス依存部分は,T-Engine への組み込みも考慮して設計を行っている.

今後の課題は以下のとおりである. 現在, デバイス依存部分を開発しているところである. マルチプロセッサとその上で動作するシステムに関して, 評価環境の実現方法を検討する必要がある. 評価項目としてリアルタイム性能があげられる. 現在の TOPPERS/FMP のデバイス依存部分と比較を行い, タスク管理のポリシー変更などの影響を調べたい.

マルチプロセッサシステムを構築する上でよく用いられる多くの種類のハードウエアモジュールを組み込み, 現在会津大学で行っている”組み込みシステム”などシングルプロセッサを用いて講義および演習で行っている状況に, マルチプロセッサおよびオペレーティングシステムを適用するなど, 教育コンテンツとしての利用も考えたい.

4. おわりに

本研究では, さまざまなシステム構成を実現するためのプラットフォームである FPGA 上に実装されたマルチプロセッサシステムをサポートするための RTOS の提案を行った. 既存の TOPPERS/FMP のデバイス依存部分に対して, 設計されたシステム構成に柔軟に対応できるようにしている. 現在, デバイス依存部分を開発しているところであり, 今後評価環境を構築し, リアルタイム性能などを評価するとともに, より多くの有用なハードウエアモジュールを組み込めるよう API の拡充を行いたい.

参考文献

1) Geoffrey Blake, Ronald G. Dreslinski, and Trevor Mudge , “A Survey of Multicore Processors,” Signal Processing Magazine, IEEE, Vol.26,Issue 6.2009.
2) Altera Corporation, “Nios II Processor Reference Handbook”, http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf.
3) (社) トロン協会 ITRON 仕様検討グループ, “μ ITRON4.0 仕様 (Ver.4.02.00),”, http://www.ertl.jp/ITRON/SPEC/FILE/mitron-402j.pdf.
4) TOPPERS Project, “TOPPERS 新世代カーネル統合仕様書”, http://www.toppers.jp/docs/tech/ngki_spec120.pdf.
5) 本田 晋也, 高田 広章 , “ITRON 仕様 OS の機能分散マルチプロセッサ拡張” 電子情報通信学会論文誌, vol.J91-D, no.4, p934-944, 2008.
6) 本田 晋也, 高田 広章 , “機能分散マルチプロセッサ向けのリアルタイム OS” IPSJ Magazine, vol.47, no.1, p41-47, Jan, 2006 .
7) R. Brighwell, K. Pedretti, T. Hudson , “SMARTMAP: Operating System Support for Efficient Data Sharing Among Processes on a Multi-Core Processor” SC ’08 Proceedings of the 2008 ACM/IEEE conference on Supercomputing IEEE Press Piscataway, NJ, USA c2008.