システムソフトウェアと
オペレーティング・システム 77－9
マルチメディア通信と分散処理 87－9
（1998．2．26）

オンチップリアルタイムOSの構成法

岩崎 裕江　長沼 次郎　遠藤 真
NTTヒューマンインタフェース研究所

本論文では、コアCPUと専用ハードウェアから構成されるエンベディドシステムLSIのチップ上に搭載可能な小型のオンチップリアルタイムOSの構成法を提案し、その有効性を評価した。本オンチップリアルタイムOSは、ファームウェアに対する要求条件の分析に基づいて、タスク管理、割込管理、セマフォ管理の最小限の機能を実現している。リアルタイムのプロトコル処理を行うMEPG2多重LSIのファームウェアに適用した場合、実行時約2KB程度の使用メモリ量で実装することができる。本オンチップリアルタイムOSは、エンベディドシステムLSIの複雑なファームウェア開発の効率化を図ることができる。

An on-chip real-time OS for an Embedded System LSI

Hiroe Iwasaki　Jiro Naganuma　Makoto Endo
NTT Human Interface Laboratories

This paper proposes a small on-chip real-time OS for embedded system LSIs, and demonstrates its usefulness. The real-time OS implements the minimum set of task, interrupt, and semaphore managements on the basis of an analysis of software characteristics. It requires only 2Kbytes memory on run-time through implementing real-time software for an MPEG2 system protocol LSI. This on-chip real-time OS makes it efficient to develop complex software for embedded system LSIs.

1．はじめに

近年のデジタル映像通信・蓄積サービスなど、様々なマルチメディア関連アプリケーションをタイムリかつ低コストで提供するため、キーデバイスとなる画像処理LSI、プロトコル処理LSIなどのマルチメディア関連LSIの開発が不可欠である。これらの大規模化・複雑化する多種多様なマルチメディア関連LSIを短期間で開発するため、コアCPUと特定用途向きの専用ハードウェアをオンチップに集積化したエンベディドシステムLSIの開発が盛んである。我々もマルチメディア用のエンベディドシステムLSIを既に開発し[1, 2]、各種マルチメディア通信・蓄積サービス用のMPEG2 CODECシステムに使用している[3, 4]。

エンベディドシステムLSIは、専用ハードウェアによる「高速性」とコアCPU上で動作するファームウェア(内蔵ソフトウェア)による「柔軟性」を有しており、大規模かつ複雑なLSIを短期間で開発するLSI構成法として有効である。しかし、複数の機能部品から構成される専用ハードウェアを制御し、かつ、リアルタイム性を要求されるファームウェアは、極めて複雑なものとなっている。これらのLSI開発においては、ハードウェア開発と共に、ファームウェア開発が全LSI開発期間の大

きな部分を占める。

一方、複数のハードウェア資源を効率良く制御し、リアルタイム性を保証するような複雑なファームウェアを効率良く開発する方法として、リアルタイムOS[5, 6]のマルチタスク機能、スケジューリング機能などの適用が有効である。これらはオンボードで実現されているエンベディドシステムにおいては極めて有効な手段として、システムに組み込まれている[7, 8]。しかし、既存のリアルタイムOSを実装するためには、実行時に数十KB～数百KB以上の使用メモリが必要となり、チップ上に搭載することは実用的ではない。これに対し我々は、使用メモリ量を削減してチップ上に搭載可能な新たなオンチップリアルタイムOS[9, 10]を検討してきた。

本論文では、エンベディドシステムLSIのチップ上に搭載可能なオンチップリアルタイムOSの構成法を提案し、その有効性を評価した。本オンチップリアルタイムOSは、ファームウェアに対する要求条件の分析に基づいて、タスク管理、割込管理、セマフォ管理の最小限の機能を実現している。リアルタイムのプロトコル処理を行うMEPG2多重(MUX)LSIのファームウェアに適用した場合、実行時約2KB程度の使用メモリ量で実装することができる。このため、各種エンベディドシステムLSIの中核として搭載し、複雑なファームウェア開発の効率化を図ることができる。

2．オンチップリアルタイムOSの構成法

ターゲットLSIの概要とそのファームウェア(内蔵ソフトウェア)に対する要求条件を示し、オンチップリアルタイムOSとして必要な機能を抽出した。

2.1 ターゲットLSIの概要

ここでターゲットとするエンベディドシステムLSIは、図1に示すように、コアCPUとメモリや複数の機能部品からなる特定用途向きの専用ハードウェアから構成され、専用ハードウェアによる高速性とファームウェアによる柔軟性を持ち合わせている。このようなLSIをリアルタイム処理を実現するマルチメディアLSIに適用する場合、ファームウェアは、複数のハードウェア資源の制御をリアルタイムに行いながら、CPUで様々な処理を行わなければならい。

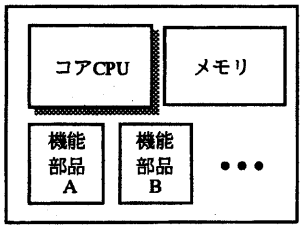

図1　エンベディッドシステムLSIの構成

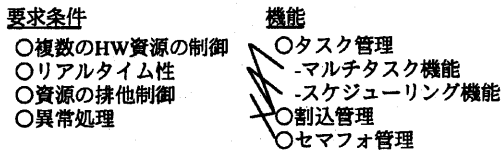

図2　要求条件とリアルタイムOSの機能

2.2 ファームウェアへの要求条件

(1)複数のハードウェア(HW)資源の制御：

ターゲットLSIは、コアCPU上で動作するファームウェアにより全体の制御を行う。従って、複数の機能部品とのインタラクションやデッドラインを考慮した上で設計されなければならない。

(2)リアルタイム性：

ターゲットLSIでは、一定時間内に処理を完了することを保証する必要がある。また、割り込みイベントが発生してから、ファームウェアが割り込みを受け付けるまでの時間が十分短くなければならない。

(3)資源の排他制御：

LSIの低コスト化のため、メモリ量、ゲート規模等のハードウェア制約が厳しい。排他制御によってオンチップ上の限られた資源を共有しなければならない。

(4)異常処理：

LSIが正常に動作できなくなった場合に、異常検出あるいは正常状態に復帰できるように処理しなければならない。

2.3 オンチップリアルタイムOSの構成

ファームウェアに対する要求条件とそれを実現するタスク管理、割込管理、セマフォ管理から構成されるオンチップリアルタイムOSとの関係を図2に示す。各々の機能を以下に示す。

(1)タスク管理：

マルチタスク機能‥マルチタスク機能では、複数のハードウェア資源の制御に対して独立し

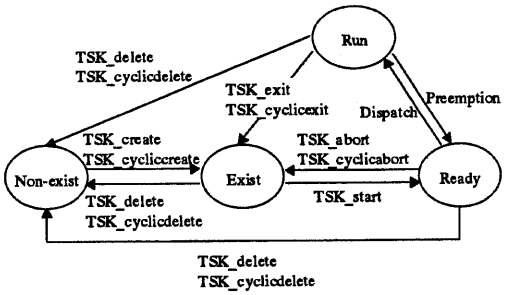

図3　タスク状態遷移図

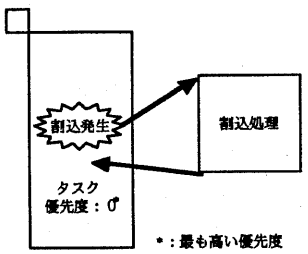

図5　割込処理

たタスクを割り当てることにより、各々のタスクを独立した処理として設計することができ、容易にインプリメントすることができる。

スケジューリング機能…優先度に従うスケジューリング機能は、各処理のリアルタイム性を保証する。各処理タスクでスケジューリングを考慮した設計をする必要がなくなり、プログラムの開発/デバッグ効率が向上する。

(2)割込管理：

割り込みイベントが発生した時に、可能な限り優先して受け付けることにより、ハードウェア機能部品とのインタラクションで発生する処理のデッドラインを容易に守ることができる。また、異常処理も同様に優先して扱うことができる。

(3)セマフォ管理：

排他制御を行うことにより、共有ハードウェア資源を有効に利用することができる。リアルタイムOSの機能を用いて、簡単に排他制御を実現することができる。

3．オンチップリアルタイムOSの設計

タスク管理、割込管理、セマフォ管理から構成されるオンチップリアルタイムOSを設計した。以下に各機能について示す。

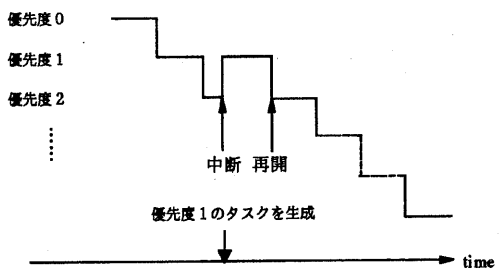

図4　プリエンプション可能なスケジューリング方法

3.1　タスク管理

タスクには、周期的に起動されるタスク(周期タスク)とそれ以外のタスクがある。タスク管理では、タスクの状態遷移を実現するため、タスク/周期タスクを生成(TSK_create/TSK_cyclicreate)、起動(TSK_start/TSK_cyclicstart)、終了(TSK_exit/TSK_cyclicexit)、削除(TSK_delete/TSK_cyclicdelete)するシステムコールを提供する。タスクは、Non-exist、Exist、Ready、Runの4つの状態を持つ。各システムコールによるタスクの状態遷移を図3に示す。

リアルタイム性が要求されるタスクは、初期設定時に生成され、通常動作中に削除されることは少ない。このため、タスク生成とタスク起動のシステムコールを分離し、通常動作でのシステムコールステップ数を減少させた。同様の理由で、タスク削除とタスク終了システムコールを分離した。

タスクスケジューリング方法は、タスク生成時に付与される優先度に従い、優先度の高いタスクから順番に実行するスケジューリング方式を用いる。実行中に優先度の高いタスクが実行待ちになった場合には、実行中のタスクをプリエンプション(中断)し、優先度の高いタスクを実行する。プリエンプション発生時のスケジューリングを図4に示す。

3.2　割込管理

リアルタイム性の厳しい処理を行ったり、異常処理を行うために割込管理機能を実現する。割込管理では、割込ハンドラ定義(INT_define)、割込ハンドラ定義解除(INT_cancel)、割込終了(INT_exit)のシステムコールを提供する。割込で処理を行う場合は、予め、割込ハンドラとして処理を登録しておく。図5に示すように、割込発生時には、最

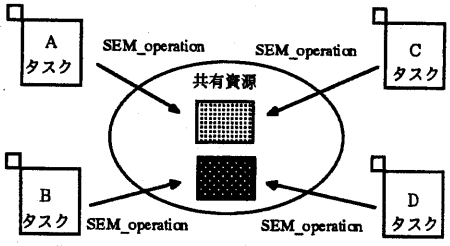

図6　セマフォ制御

表1　システムコール一覧

項目	システムコール名	機能
初期設定	INI_kernel	カーネル初期設定
タスク管理	TSK_create	タスク生成
	TSK_start	タスク起動
	TSK_exit	タスク終了
	TSK_delete	タスク削除
	TSK_cycliccreate	周期タスク生成
	TSK_cyclicstart	周期タスク起動
	TSK_cyclicexit	周期タスク終了
	TSK_cyclicdelete	周期タスク削除
	TSK_abort	他タスク強制終了
	TSK_cyclicabort	他周期タスク強制終了
	TSK_idget	自タスクid情報取得
割込管理	INT_define	割込ハンドラ定義
	INT_cancel	割込ハンドラ定義解除
	INT_exit	割込終了
セマフォ管理	SEM_create	セマフォ生成
	SEM_operation	セマフォ操作
	SEM_ddete	セマフォ削除

も高い優先度を持つタスクの処理中であっても、登録している割込ハンドラが直ちに起動される。

3.3　セマフォ管理

セマフォ管理により、資源の排他制御を実現する。セマフォ管理では、セマフォ生成(SEM_create)、セマフォ操作(SEM_operation)、セマフォ削除(SEM_delete)のシステムコールを提供する。セマフォの資源数の操作は、すべてセマフォ管理のシステムコールで行う。また、セマフォの資源数は、セマフォ生成時に指定する。各タスクは、セマフォ操作システムコールを発行することにより、資源の確保・解放ができる。図6に示すように、セマフォ制御で、共通に使用できる資源を排他制御して、タスク間で資源を共有することができる。

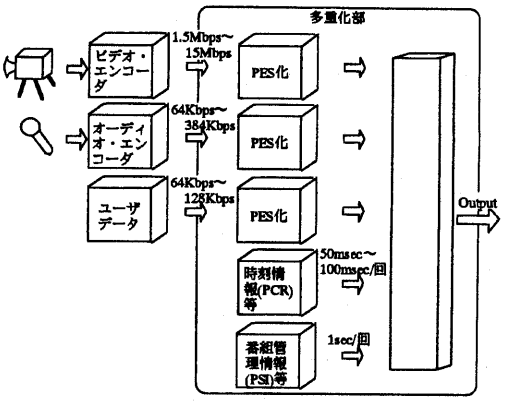

図7　MPEG2多重処理

3.4　システムコール

本オンチップリアルタイムOSは、初期設定/タスク管理/割込管理/セマフォ管理を１８種類のシステムコールで実現している。システムコール一覧を表１に示す。これらを用いることにより、リアルタイム性を保証するような複雑なファームウェアを効率良く開発/デバッグすることができる。

4．オンチップリアルタイムOSの実装と評価

リアルタイム通信のプロトコル処理を行うMPEG2多重(MUX)LSIのファームウェアに適用し、本オンチップリアルタイムOSの動作確認と性能評価を行う。

4.1　MPEG2 MUX LSIへの適用

4.1.1　LSIの概要

MUX LSIは、図7に示すように、エンコードされたVideo、Audioデータ やUserデータをパケット化(PES化)し、PCR(時刻情報)、PSI(番組管理情報)データと多重化して１つのストリームを作成する。各々の処理のデッドラインは様々であるが、PCR処理に関するデッドラインが最も厳しく、50msec～100msecに最低１回の処理が必要である。MUX LSIファームウェアでは、複数のデッドライン要求を満足しなければならない。

4.1.2　ファームウェア設計

既存のMUX LSIのファームウェア[2]を、本オンチップリアルタイムOSと複数のタスクで新たに構成した。MUX LSIのファームウェアは、Video、Audio、User、PCR、PSIなどの処理単位で、独立したタスクに分割できる。オンチップリアルタイ

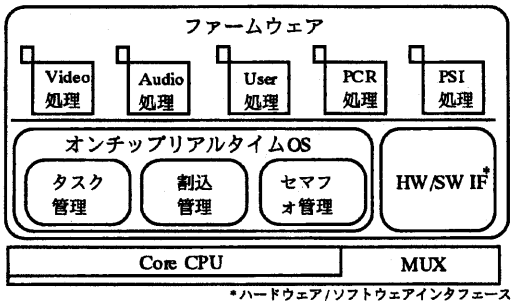

図8　MPEG2 MUX LSI ファームウェア構成

表2　タスクの優先度とプログラム規模

タスク	優先度	プログラム規模 (kline)
初期化タスク	9	0.50
PCR初期化タスク	3	0.05
PSI初期化タスク	3	0.10
Video初期化タスク	3	0.10
PCRタスク	1	0.25
PSIタスク	2	1.28
Video監視タスク	8	0.25
Video header処理タスク	3	0.97
ヘッダファイル等	-	5.00
計		8.50

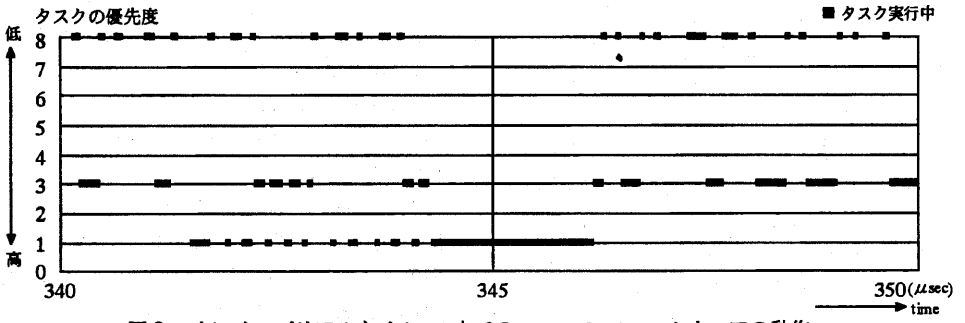

図9　オンチップリアルタイムOS上でのMUX LSI ファームウェアの動作

ムOS上でのMUX LSIファームウェアの構成を図8に示す。ここでは、評価用として、Video、PCR、PSIの処理を作成した。時刻管理を行うPCR処理は、最も速い応答性が必要であるため、PCRタスクに最も高い優先度を与えた。各タスクの優先度とプログラム規模を表2に示す。各タスクは、依存関係がなく独立な処理として記述できるため、ファームウェア設計が容易であった。

割込ハンドラを含めすべての処理をC言語で記述することができるため、開発/デバッグ効率が向上した。

4.1.3　実験環境と動作確認

UNIX上で動作するMUX LSI仮想環境[11]を用いて、本オンチップリアルタイムOSを用いて開発したMUX LSIのファームウェアを動作させた。本仮想環境は、命令レベルでファームウェアを実行することができる。また、MUX LSIハードウェアの機能を仮想的にUNIX上で実現している。

伝送レート、ビデオ符号化レートを6Mbit/s、5Mbit/s、PCR周期、PSI周期を100ms、1sに設定し、仮想環境上で動作を確認した。図9に本オンチップリアルタイムOS上のタスクの動作状態を示す。最も優先度の高いPCRタスクが、Video監視タスクなどより低い優先度のタスクをプリエンプションして、優先的に実行されている。また、出力のビットストリームを解析した結果、PCR/PSI周期が守られ、Videoデータが所望の割合で出力されていることを確認した。

4.2　メモリ量の評価

本オンチップリアルタイムOSが必要とする動的な使用メモリ量は、アプリケーションのタスク数に依存する。表3に本オンチップリアルタイムOSのデータ領域とスタック領域に使用したメモリ量を示す。また、MUX LSIアプリケーションのデータ領域とスタック領域についても表3に示す。本オンチップリアルタイムOS部分の動的な使用メモリ量が約2KB程度で実現でき、MUXファームウェア部分が約1.4KBで実現できた。本オンチップリアルタイムOSと複数のタスクで構成したファームウェアで、4KB以下のメモリ量で実現することができた。

4.3　オンチップリアルタイムOS適用の効果

本オンチップリアルタイムOS適用時の効果を以下に示す。

(1)複数のハードウェア資源の制御：

マルチタスク機能を用いて、複数のハードウェ

表3　動的な使用メモリ量

項目	使用メモリ量(KB)
オンチップRTOS[1]データ領域	0.16*タスク数(8)+0.5
オンチップRTOS[1]スタック領域	0.3
アプリケーションデータ領域	1.0
アプリケーションスタック領域	0.4
計	3.48

[1]リアルタイムOS

ア資源を制御することができる。PCR処理/Video処理の処理単位で、独立にタスクを設計することが可能である。各々のタスクは、オンチップリアルタイムOSが提供するマルチタスク機能により、他のタスクとの依存性を考慮せずに設計できる。

(2)リアルタイム性：

オンチップリアルタイムOSでスケジューリングを行い、リアルタイム性を保証している。このため、PCR処理、Video処理等で、互いにデッドラインを考慮してファームウェアを設計する必要はない。また、割込管理により、割込の即応性にも優れている。

(3)資源の排他制御：

オンチップリアルタイムOSで提供されるシステムコールを用いて、容易にタスク間で資源を共有することができる。

(4)異常処理：

割込管理のサポートにより、即座に異常を検出し、適切な異常処理を行うことができる。

以上の効果により、ファームウェア設計が簡易になり、ファームウェア開発の短TAT化を図ることができる。さらに、タスクとして、ファームウェアを複数の機能毎に部品化を図ることにより、機能の変更や追加を容易に行うことができるばかりでなく、再利用も容易である。

集積回路技術の進展により、オンチップシステムの大規模・複雑化は今後も進展すると考えられるため、エンベディドシステムLSIにオンチップリアルタイムOSを搭載することは、複雑なファームウェア開発の効率化を図るために有効である。

5．おわりに

エンベディドシステムLSIのファームウェア開発の短TAT化を図るため、チップ上に搭載可能なオンチップリアルタイムOSを開発した。本オンチップリアルタイムOSを用いて、MPEG2 MUX LSIのファームウェアを開発し、動作確認と性能評価を行った。その結果、実行時約2KBの使用メモリ量で実現することができた。また、複雑なファームウェア設計が容易となり、ファームウェア開発/デバッグの短TAT化が図れた。今後、本オンチップリアルタイムOSをエンベディドシステムLSIの中核として、各種マルチメディアアプリケーションに適用して行く。

[参考文献]

[1]M. Inamori, J. Naganuma, and M. Endo, A Memory-based Architecture for MPEG2 System Protocol LSIs, ED&TC 96, Mar. 1996.

[2]J. Naganuma and M. Endo., Polling-based Real-time Software for MPEG2 System Protcol LSIs, ASP-DAC97, Jan. 1997.

[3]N. Terada, H. Ishii, T. Tachi, Y. Okumura, and H. Kotera, A MPEG2-Based Digital CATV and VOD System using ATM-PON Architecture, Proc. of IEEE MULTIMEDIA'96 Conference, pp. 522--531, 1996.

[4]Y. Tashiro, T. Izuoka, K. Yanaka, Y. Ito, N. Ono, Y. Yashima, H. Yamauchi, and H. Kotera, MPEG2 Video and Audio CODEC Board Set for a Personal Computer, Proc. of IEEE Global Telecommunications Conference, Vol. 1, pp.483--487, 1995.

[5]J.L.Perterson、オペレーティングシステムの概念、培風館、1987.

[6]Sakamura, K., ed., μITRON3.0 Specification, TRON Association.

[7]中本、高田、田丸、組み込みシステム技術の現状と動向、情報学会誌 Vol. 38 No.10、Oct. 1997.

[8]G.Castelli and G.Ragazzini , EOS: A Real-Time Operating System Adapts to Application Architectures, IEEE Micro, Vol. 15, No. 5, Oct. 1995.

[9]岩崎、長沼、遠藤、オンチップリアルタイムOS構成法の検討 - MPEG2多重/分離LSIへの適用 -、1996年電子情報通信学会ソサエティ大会.

[10]岩崎、長沼、遠藤、オンチップリアルタイムOSの設計評価、情報処理学会第55回全国大会、Sep. 1997

[11]遠藤、岩崎、長沼、仮想LSI環境によるリアルタイムOSとMPEG2 MUXファームウェアの設計、1997年電子情報通信学会ソサエティ大会.