快適な実時間マルチタスク処理を実現するプロセッサ

（研究課題名：実時間マルチタスク処理を支援するプロセッサアーキテクチャ）

「機能と構成」領域　　田中　清史

要　　旨

従来のマルチスレッド型プロセッサは、内蔵するプロセッサコンテキスト数に限りがあるため、メモリアクセスレイテンシの隠蔽能力に限界がありました。この問題を解決するために、本研究ではプロセッサ内にコンテキスト専用のバッファを設け、タスク実行のバックグラウンドでバッファ内コンテキストとプロセッサコンテキストとのスワップを行うことにより、シームレスなタスク切り替えを可能とする拡張マルチスレッドアーキテクチャを提案しました。また、リアルタイムシステムにおいて重要な高速割り込み応答を可能とする機構、さらにキャッシュメモリの再構成により、大規模データに対する高速なアクセスを実現する機構を提案してきました。本研究では、提案する機構を持つプロセッサを実際にLSIとして研究開発・評価しました。

1．研究のねらい

近年LSIの集積度が上がり、従来のプロセッサよりも大規模な回路が1チップに搭載可能となってきました。このことから、オンチップマルチプロセッサ化、システムオンチップ化が次世代のキーテクノロジーとして注目されています。また、シングルプロセッサの機構としても、特別な付加ハードウェアを必要とする各種機構が現実的に可能になってきました。しかし、従来の高速化を達成するための研究では、シングルタスクの高速化に重点が置かれてきました。本研究では、実時間（リアルタイム）汎用マルチタスク環境におけるプロセッサの有効実行性能を向上させるための機構をターゲットとしています。

計算機はマルチタスク環境で使用されていますが、これは各実行タスクが時分割でプロセッサを使用することにより実現されています。1つのタスクが一定時間プロセッサを使用した後、あるいは外部からの割り込みが発生した時、プロセッサコンテキストが入れ換えられ、他のタスクによるプロセッサ使用に移行します。現状のプロセッサにおいてプロセッサコンテキストを入れ換える際、入れ換え前の汎用レジスタおよびプログラムカウンタ、状態

レジスタなどの専用レジスタの内容をメモリに退避し、入れ換え後のタスクのために汎用レジスタおよび専用レジスタの内容をメモリから読み出してセットするといった、メモリベースでの手続きが実行されます。このような方式ではキャッシュミスなどの要因により、コンテキスト切り替えに要する時間が各タスクの有効実行時間に対して無視できない大きさになります。更には、リアルタイム性が要求されるタスクの場合には、コンテキスト切り替えの時間を含んだ最悪実行時間によってそのリアルタイム性を保障するため、このコンテキスト切り替えの処理時間を削減することによりタスク実行に余裕を持たせることが可能となります。また、組み込みシステムをターゲットとした場合、敏速な割り込み応答が要求されるため、割り込み処理への高速なコンテキスト切り替えが重要となります。

本研究では、従来のマルチスレッドアーキテクチャを拡張することによりタスク切り替えの際のレジスタ値やアドレス空間などのプロセッサコンテキストを入れ換える時間をゼロにするアーキテクチャ、およびキャッシュメモリを再構成することによりメモリアクセス時間を削減する方法を提案し、実際にプロセッサの回路を設計し、LSIを開発・評価してきました。

2. 研究方法と成果

2.1 優先度ベースのマルチスレッドアーキテクチャ

近年、インテルのPentium 4、IBMのPOWER5など、マルチスレッド型のプロセッサが出現してきました。マルチスレッドアーキテクチャは複数のスレッド（タスク）の実行コンテキストを内蔵し、それらを瞬時に切り替えて実行することにより、メモリアクセスレイテンシの隠蔽および命令実行ユニットの稼働率を向上させます。マルチスレッドアーキテクチャには、クロックサイクル毎にスレッドを切り替えるcycle-by-cycle multithreading方式、キャッシュミスなど長いレイテンシを発生させる事象で切り替えるblock-multithreading方式、および毎クロックサイクルで複数のスレッドからそれぞれ任意数の命令を実行するsimultaneous multithreading方式があります。本研究で提案するプロセッサは基本的にblock-multithreading方式を利用し、スレッドの優先度を考慮した切り替えを行います。

2.2 プロセッサコンテキストバッファ

大規模データを対象とするアプリケーションの普及により、メモリアクセスの待ち時間はプログラムの全実行時間の75％にもおよぶとされています。このことはスレッド実行が頻繁

にキャッシュミスを発生させることを意味します。したがって、数個のスレッドコンテキストを内蔵していても、メモリアクセスの全ての待ち時間を隠蔽することは困難です。一つのスレッドコンテキストを格納する領域を本研究ではプロセッサコンテキストセット（PCS）と呼びますが、これを多数用意することはハードウェアのサイズおよびコストの面から困難であり、また実行ユニットに接続するための選択器（セレクタ）の規模が大きくなるため動作速度が低下する要因となります。そこで本研究では、小規模のハードウェアで仮想的に切り替え対象のスレッドコンテキスト数を増加させる“プロセッサコンテキストバッファ（PCB）”を提案しました。

図1に4つのPCSとPCBの構成例を示します。図では、セレクタによって選択されているPCS1のタスクが実行中（アクティブ）です。このタスク実行がキャッシュミスを起こすと、他のPCSの実行に切り替えられます。実行中にキャッシュミスを起こしたPCSは、図中のPCS2のように、PCB内のタスクと入れ替えられます。この入れ替え（スワップ）はタスク実行のバックグウンで行われます。このスワップにより、常に実行継続可能なタスクをPCSに存在させ、アクティブタスク実行のキャッシュミス発生時に切り替え先が存在するようにします。

本研究では、マルチスレッドの切り替え先の選択、およびPCB内のスワップ対象タスクの選択ともに優先度を使用します。すなわち、キャッシュミスが発生した場合、他のPCSタスクのうち最も優先度の高いタスクを持つPCSに実行を切り替えます。また、キャッシュミス

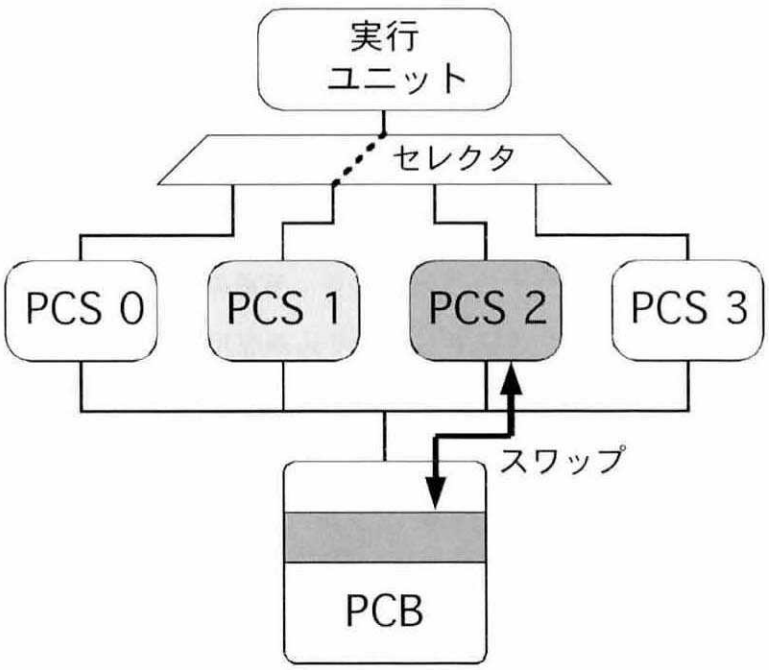

図1　プロセッサコンテキストバッファ（PCB）

を起こしたタスクのPCSは、PCB内のタスクのうち最も優先度の高いものとスワップされます。さらに、PCB内の継続可能タスクのうち、その優先度がPCSのタスクの優先度よりも高い場合、それらはスワップされます。すなわち、常に実行継続可能なタスクのうち優先度の高いものをPCSに存在させます。

PCBのエントリ数は有限であるため、オペレーティングシステムが管理するタスクの全て、あるいはリアルタイムアプリケーション内の分割されたタスクの全てをPCSあるいはPCBに存在させることは不可能です。そこで、メモリ内にコンテキストストレージバッファ（CSB）領域を儲け、そこに残りのタスクを配置します。PCB内タスクとCSB内タスクの間で入れ替えを行いますが、これはソフトウェアの指示によります。すなわち、タスクは実行中に特別なレジスタにスワップの相手タスクを指定して自タスクの実行を中断します。中断されたタスクはPCSからPCBへ移動し、その後CSB内の相手タスクと入れ替えられます。この入れ替えはアクティブタスク実行のバックグラウンドで行われます。

２．３　高速割り込み応答機構

割り込みが発生した際には、一般に割り込みハンドラは実行中のタスクのコンテキストをメモリに保存し、割り込み処理終了時に保存したコンテキストをメモリから読み出すことによって元のタスク実行に復帰します。このメモリベースの保存・復帰はキャッシュミスの原因となり、特に割り込みに対する応答時間が重要であるリアルタイムシステムでは問題となります。本研究ではマルチスレッドアーキテクチャを応用し、割り込み専用PCSを用意することにより上記のオーバヘッドを削除します。すなわち、あるタスクの実行中に割り込み要求が発生した場合、そのタスクのPCSの内容はそのままにしておき、割り込み専用PCSに切り替えて割り込み処理を行います。同様に割り込み処理終了時には、単に元のPCSに切り替えることによって元のタスク実行を再開します。

複数の割り込み専用PCSを用意することにより、多重の割り込みに対して高速に応答可能となります。例えば、割り込みが発生して割り込み専用PCSに切り替えて割り込み処理を開始しますが、その処理中に優先度のより高い更なる割り込みが発生した場合、別の割り込み専用PCSに切り替えてその割り込み処理を行うことが可能です。

２．４　再構成可能キャッシュメモリ

従来のキャッシュメモリは、アプリケーションの参照するデータの時間的・空間的局所性にその効率が依存していました。今日の大規模数値計算やデータベース、メディアプロセッ

シングなどではデータ参照に局所性が無いものが多く、高速なキャッシュメモリの有効利用が困難になっています。またマルチスレッド型プロセッサにおいてスレッド同士でキャッシュを共有する場合、スレッド間でキャッシュ内の使用領域の競合が頻発し、キャッシュミスが増加、すなわちキャッシュ効率が低下する傾向があります。そこで本研究では、キャッシュメモリ管理法を動的に再構成することにより、データ参照効率を向上させることを提案しました。

優先度ベースパーティショニング

スレッド間でキャッシュを共有するため、単一スレッド実行型プロセッサよりもキャッシュ参照の効率が低下します。本研究では各スレッドに優先度を付加しているため、なるべく優先度の高いスレッドのキャッシュミスを削減することが重要です。そこでセットアソシアティブ方式のキャッシュをパーティションに分割し、スレッド毎に使用するパーティションを限定する方式を提案しました。例えば、4ウェイ・セットアソシアティブキャッシュを再構成することにより、4つのパーティションに分割します。例えば優先度の高いスレッドに3つのパーティション、その他に1パーティションを割り当てることにより、優先度の高いスレッドのキャッシュヒット率をなるべく維持することが可能となります。動的再構成によるパーティション分割、およびスレッドへのパーティションの割り当てはオペレーティングシステムが行います。

FIFOバッファ

数値計算やデータベース処理では、メモリ内に一定間隔に配置されたデータ列を連続して参照する場合がありますが、このような参照ではキャッシュに読み込んだ全データのうち、実際に参照するデータは断片的に存在することになるため非効率です。そこで本研究では、動的再構成により一つのパーティションをFIFOバッファとして使用可能とする方式を提案しました。図2に通常のキャッシュの一つのウェイからFIFOバッファパーティションへの再構成を示します。このFIFOバッファは、従来のキャッシュのようにブロック単位の管理・使用ではなく、図のように読出しカウンタと書込みカウンタによって指されるデータ単位での使用となるため、キャッシュメモリの断片的使用を回避します。また、外部のDMA機構と協調動作させることで、ブロックサイズを超えたプリフェッチ効果によりプログラム実行の高速化が期待できます。

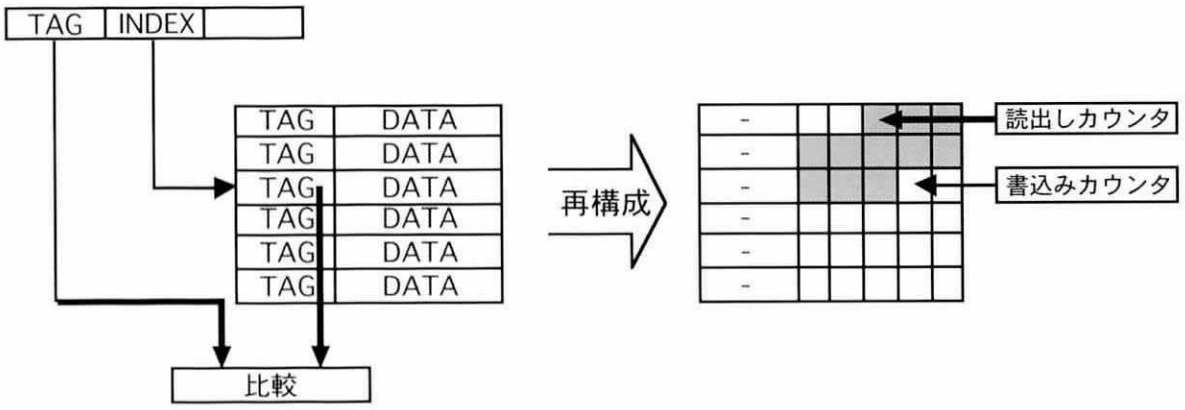

図2　再構成によるFIFOバッファの提供

2.5　プロトタイププロセッサ：*PRESTOR-1*

本研究では、提案する各種機構を持つプロセッサを実際に設計し、LSIとして研究開発しました。本プロセッサを*PRESTOR-1*（PRESTO RISC-1）と呼びます。開発したLSIチップの写真を図3に示します。また、開発したLSIを搭載する評価システムを作成しました（図4）。*PRESTOR-1*の仕様は表1のようになっています。

表1　*PRESTOR-1*の仕様

パッケージング	HQFP240プラッチックパッケージ
テクノロジー	0.18μm　5Metal ゲートアレイ
ダイサイズ	5.0mm × 10.0mm
内部動作電圧／入出力信号レベル	1.8V ／ 3.3V
命令セット	SPARC Architecture Version 8　整数命令互換
拡張命令	レジスタセット間データ移動命令
	割込みレベル制御命令
	キャッシュ制御命令
	バイトオーダー変換命令
キャッシュメモリ容量（命令／データ）	16KB ／ 16KB
再構成FIFOバッファ容量	4 KB
TLBサイズ（命令／データ）	256エントリ／ 256エントリ
PCS数（通常実行用／割り込み専用）	4 ／ 4
PCBエントリ数	4
命令実行パイプライン段数	10段
外部割込み要求信号	15チャネル

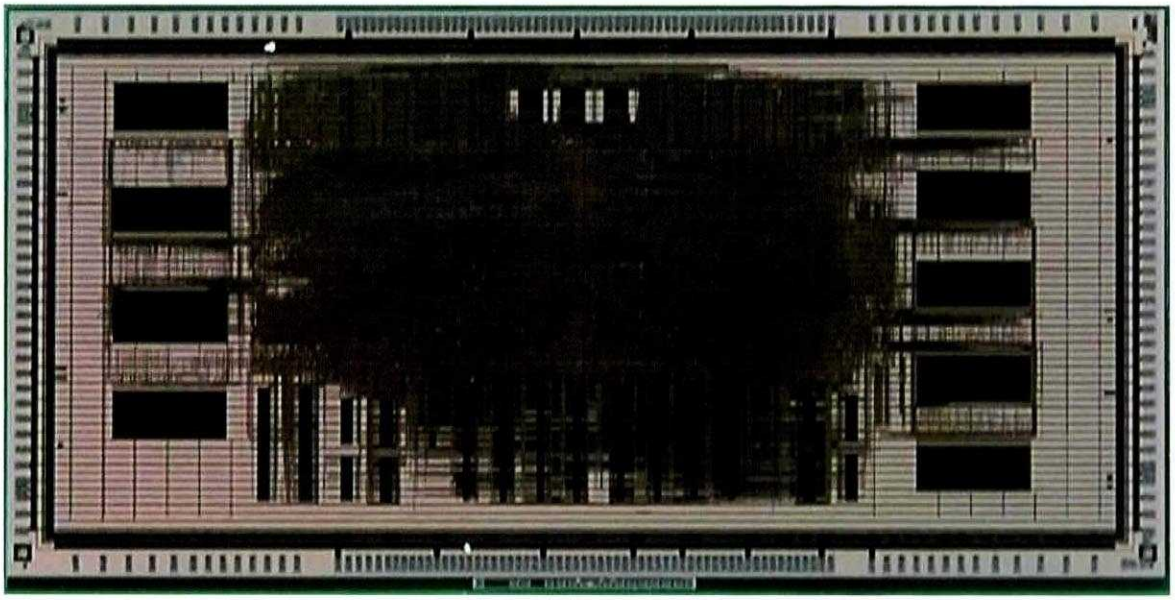

図３　*PRESTOR-1*チップ写真

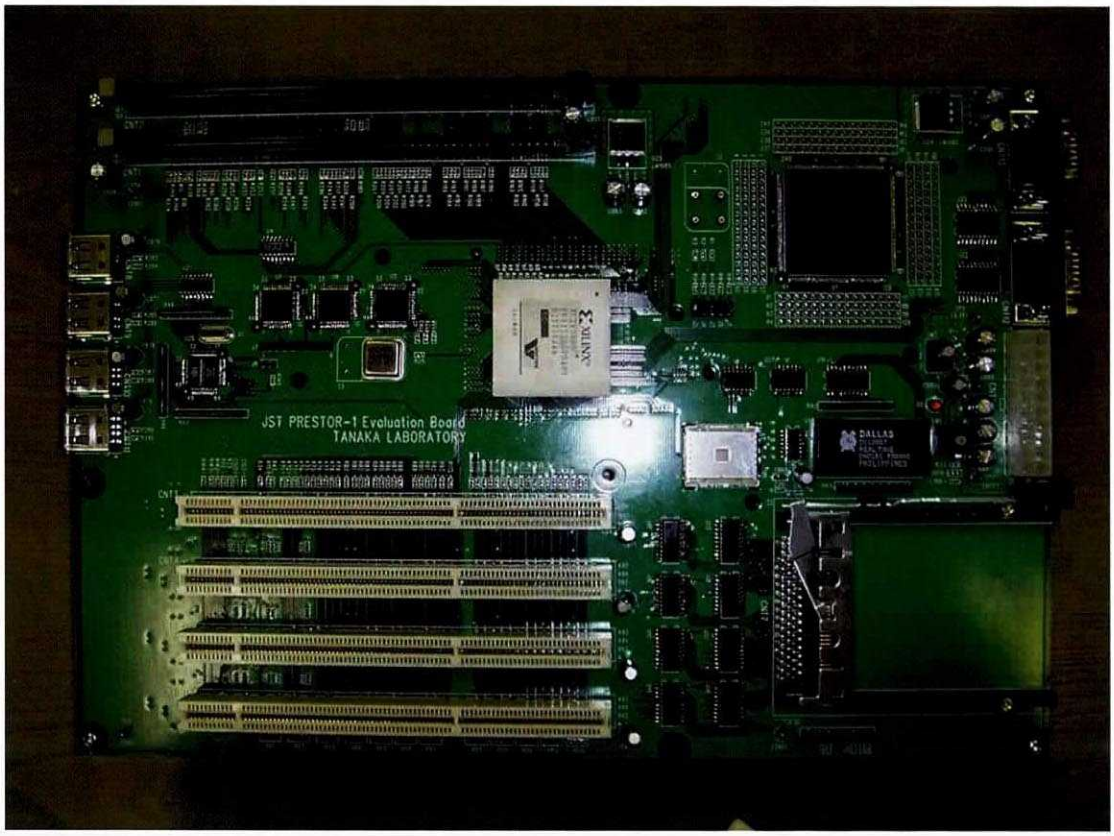

図４　評価基板

評価システムの仕様を表２に示します。

表２　評価基板の仕様

プリント基板サイズ	210mm×300mm
CPU	*PRESTOR-1*
メモリコントローラ	Xilinx社 FPGA（XC2V3000）
DIMMソケット（SDRAM）	2
PCIスロット（64ビット／66MHz）	4
USB2.0ポート	4
シリアルポート	2
PCカードスロット（PCMCIA）	1
リアルタイムクロック	MAXIM　DS12887

２．６　基本性能評価

高速割り込み応答機構の評価

ハードウェア記述言語VHDLで記述された*PRESTOR-1*のRTLシミュレーションにより、高速割り込み応答機構の評価を行いました。シミュレーションでは、それぞれ優先度を割り当てた６つの割り込みを発生させ、最初の割り込み発生から、最後の割り込み処理の終了までのサイクル数を得ました。６つの割り込みのうち、３つがそれぞれ割り込み専用PCSでの実行、３つが一つの割り込み専用PCSを共用しました。結果を表３に示します。

表３　シミュレーション結果（割り込み処理）

方　　式	サイクル数
通常方式	1,775
レジスタウィンドウ方式	949
割り込み専用PCS方式	856

表３において、通常方式は一つのPCSで全ての割り込み処理を行うものです。これは、割り込み発生／処理終了で毎回コンテキストの入れ替えを行います。レジスタウィンドウ方式は、SPARCアーキテクチャにおけるレジスタウィンドウを使用するもので、割り込み発生／終了時には32本の汎用レジスタのうち、グローバルレジスタ８本、オーバーラップレジスタ８本を退避／復帰する必要があります。この結果から、割り込み専用PCSを使用することにより、一回の割り込みあたり230サイクルを削減可能であることがわかりました。こ

れは主に、コンテキストの退避／復帰のためのストア／ロード命令実行のオーバヘッドとキャッシュミス削減の効果です。

再構成可能キャッシュメモリによるFIFOバッファの評価

続いて、再構成可能キャッシュメモリが提供するFIFOバッファの使用に関するRTLシミュレーション結果を示します。右に示す配列内の一定間隔要素の総和を計算するプログラムを実行した結果を表4に示します。

```
#define  STRIDE  8
main () {
  int  i, sum = 0;
  int  vector[N];

  for (i = 0;  i < N;  i += STRIDE) {
    sum += vector[i];
  }
}
```

表中で、Normalが通常キャッシュ、FIFOが再構成後のFIFOバッファを使用した結果です。繰り返し数Nが大きい場合、FIFOバッファへのプリフェッチ効果が大きくなり、実行時間の削減率が向上しました。以上の結果から、*PRESTOR-1*が提供する高速割り込み応答機構、再構成可能キャッシュメモリによるFIFOバッファの有効性が示されました。

表4　シミュレーション結果（STRIDE＝8）

N	キャッシュ	サイクル数	削減率（%）
128	Normal	4,781	75.8
	FIFO	1,156	
512	Normal	18,701	81.5
	FIFO	3,460	
2048	Normal	74,361	83.0
	FIFO	12,676	

3．今後の展望

本研究では、マルチタスク環境で特に大規模データをターゲットとしたリアルタイムアプリケーションの実行を高速化する方式を研究してきました。今後は開発したLSIを搭載した評価システムを使用し、基本性能の評価、およびマルチメディア系リアルタイムアプリケーションの評価を行います。また、今後の注目すべき計算環境としての並列計算機や計算機クラスタシステムにおいて、データ通信時間が大きいためにプロセッサの浪費時間が大きいことに着目し、これに提案するプロセッサアーキテクチャを適用することで可能な限り浪費時

間を削減し、汎用リアルタイム環境を実現する研究を行う予定です。

4．成果リスト

1．K. Khalid and K. Tanaka, Implementation of FIFO Buffer Using Cache Memory, 情報処理学会　計算機アーキテクチャ研究会, Vol.2002, No.112, 2002年.

2．K. Tanaka, Fast Context Switching by Hierarchical Task Allocation and Reconfigurable Caches, Proc. of IWIA 2003, IEEE Computer Society Press, pp.20-29, 2003.

3．K. Khalid and K.Tanaka, Evaluation of Cache Memory as FIFO Buffer, 情報処理学会　計算機アーキテクチャ研究会, Vol.2003, No.27, 2003年.

4．K. Tanaka and T. Fukawa, Highly Functional Memory Architecture for Large-Scale Data Applications, Proc. of IWIA 2004, IEEE Computer Society Press, pp.109-118, 2004年.

5．今井俊晴, 田中清史, 高速フィルタリングを支援する高機能メモリコントローラ, 情報処理学会　計算機アーキテクチャ研究会, Vol.2004, No.123, 2004年.

6．K. Tanaka, *PRESTOR-1*: A Processor Extending Multithreaded Architecture, IWIA2005.