

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第4870914号
(P4870914)

(45) 発行日 平成24年2月8日 (2012. 2. 8)

(24) 登録日 平成23年11月25日 (2011. 11. 25)

(51) Int. Cl.	F I
G 0 6 F 9/46 (2006. 01)	G 0 6 F 9/46 4 1 0
G 0 6 F 9/54 (2006. 01)	G 0 6 F 9/46 4 8 0 C

請求項の数 11 (全 46 頁)

(21) 出願番号	特願2004-158420 (P2004-158420)	(73) 特許権者	000005049
(22) 出願日	平成16年5月27日 (2004. 5. 27)		シャープ株式会社
(65) 公開番号	特開2004-362564 (P2004-362564A)		大阪府大阪市阿倍野区長池町2 2 番 2 2 号
(43) 公開日	平成16年12月24日 (2004. 12. 24)	(73) 特許権者	511093317
審査請求日	平成18年9月12日 (2006. 9. 12)		スティーブン フランク
(31) 優先権主張番号	10/449, 732		アメリカ合衆国 マサチューセッツ O 1
(32) 優先日	平成15年5月30日 (2003. 5. 30)		O 6 2, フローレンス, ノース ファ
(33) 優先権主張国	米国 (US)		ームス ロード 6 8 0
前置審査		(74) 代理人	100078282
			弁理士 山本 秀策
		(74) 代理人	100062409
			弁理士 安村 高明
		(74) 代理人	100107489
			弁理士 大塩 竹志

最終頁に続く

(54) 【発明の名称】 デジタルデータプロセッサ

(57) 【特許請求の範囲】

【請求項 1】

1 つ以上のプロセッサと、

該 1 つ以上のプロセッサ上で実行される 1 つ以上の仮想処理装置であって、各仮想処理装置は、1 つ以上の処理またはスレッド（まとめて「スレッド」という）を実行し、各スレッドは、そのスレッドの寿命の間、同じ仮想処理装置および／または同じプロセッサで実行されるように制約されているか、制約されていないかのいずれかである、1 つ以上の仮想処理装置と、

該 1 つ以上のプロセッサと通信接続されているイベント伝達メカニズムとを備え、

該イベント伝達メカニズムは、

該 1 つ以上のプロセッサを形成するロジックとは独立したロジックであって、(i) 1 つ以上のイベントの各々をそれぞれのスレッドに、イベントとスレッドとの対応を示す情報に基づいて関連付け、(i i) スレッドが仮想処理装置および／またはプロセッサに対して透過的に実行できるように、スレッドがいずれの仮想処理装置および／またはプロセッサ上で実行中であるかに関わらず、それぞれの該スレッドに、関連付けられたイベントを通知するロジックを備え、

該スレッド同士は、メモリおよび該イベントを利用して、互いに通信するよう構成されている、デジタルデータプロセッサ。

【請求項 2】

前記イベントは、ハードウェア割り込み、ソフトウェア開始シグナリングイベント（「

ソフトウェアイベント」) およびメモリエventのいずれかを含むというさらなる改善点を有する、請求項1に記載のデジタルデータプロセッサ。

【請求項3】

少なくとも関連付けされた前記イベントを通知されたスレッドは、一時停止状態から実行状態に移行するというさらなる改善点を有する、請求項1に記載のデジタルデータプロセッサ。

【請求項4】

前記イベント伝達メカニズムは、イベント発生を伝えるイベントキューを含むというさらなる改善点を有する、請求項1に記載のデジタルデータプロセッサ。

【請求項5】

前記イベント伝達メカニズムは、イベントをスレッドにマッピングするテーブルを含み、前記イベントとスレッドとの対応を示す情報とは、該テーブルであるというさらなる改善点を有する、請求項1に記載のデジタルデータプロセッサ。

【請求項6】

前記テーブルは、イベントを、スレッドが該イベントの発生に応答するために、該スレッドにマッピングするというさらなる改善点を有する、請求項5に記載のデジタルデータプロセッサ。

【請求項7】

前記イベント伝達メカニズムは、ペンディングメモリ演算に関連のあるイベントと、該ペンディングメモリ演算を生じさせたメモリ参照命令を実行したそれぞれの前記スレッドとの間の関連を確立するテーブルを含み、

前記イベントとスレッドとの対応を示す情報とは、該テーブルであり、

該テーブルは、前記ペンディングメモリ演算のアドレス、状態情報、該メモリ参照命令を実行したスレッドのIDを保持するというさらなる改善点を有する請求項1に記載のデジタルデータプロセッサ。

【請求項8】

前記イベント伝達メカニズムは、処理装置のうちの1つで実行中のシステムスレッドに、仮想処理装置に常駐しないスレッドに関連付けられたイベントの発生を通知し、

該システムスレッドとは、システム特権およびアプリケーション特権のうち該システム特権を有するスレッドであり、

該システムスレッドは、仮想処理装置に常駐しない該スレッドを一時停止状態から実行状態に移行させるというさらなる改善点を有する、請求項1に記載のデジタルデータプロセッサ。

【請求項9】

前記システムスレッドは、前記仮想処理装置で常に実行中ではなく、待機またはアイドル状態（一時停止状態）であるスレッドに関連付けられたイベントの発生を通知された際、該待機またはアイドル状態（一時停止状態）であるスレッドを一時停止状態から実行状態に移行させるというさらなる改善点を有する、請求項8に記載のデジタルデータプロセッサ。

【請求項10】

前記スレッドと通信接続されているイベントキューであって、該イベントキューは、少なくとも1つの該スレッドによって該イベントキューに入れられた少なくともソフトウェア開始シグナリングイベント（「ソフトウェアイベント」）を含む前記イベントを伝える、イベントキューを備え、

前記イベント伝達メカニズムは、該スレッドと、該イベントキューとに通信接続されているイベント伝達メカニズムであって、該イベント伝達メカニズムは、それぞれの該スレッドに、関連付けられた該イベントの発生を通知し、該イベントは、少なくともソフトウェアイベントを含み、

イベントが通知される少なくとも1つのスレッドが、（i）該イベントキューにアクセスして通知を引き起こした発生を決定し、（ii）該イベントの発生を扱うこととによ

10

20

30

40

50

て、ソフトウェアイベントに応答する、請求項 1 に記載のデジタルデータプロセッサ。

【請求項 11】

ソフトウェアイベントを前記イベントキューに入れる命令を実行する 1 つ以上のスレッドが、いずれのスレッドにイベントを通知するかを決定し、決定されたスレッドにイベント伝達メカニズムによりイベントを通知する、請求項 10 に記載のデジタルデータプロセッサ。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、デジタルデータ処理、より具体的には、統一イベント通知およびコンシューマ (consumer) - プロデューサ (producer) メモリ演算を有する デジタルデータプロセッサ に関する。

10

【背景技術】

【0002】

コンピューティングおよびアプリケーションの進化には、大きく分けて 3 つの段階があった。最初に、メインフレームおよびミニコンピュータの段階があった。次に、パーソナルコンピュータの段階が続いた。現在は、埋め込みプロセッサ、または「変装したコンピュータ」の段階である。

【0003】

20

埋め込みプロセッサは、デジタルテレビ、デジタルビデオレコーダ、PDA、携帯電話、ならびに、マルチメディア用途 (MP3 デコードおよび／またはエンコードを含む) と、音声および／またはグラフィカルユーザインターフェースと、インテリジェントエージェントおよび他のバックグラウンドタスクと、ならびに透過的インターネット、ネットワーク、ピア・ツウ・ピア (P2P) もしくは他の情報アクセスとをサポートする他の機器において用いられている。これらのアプリケーションの多くは、複雑な映像、音声または他の信号処理を必要とし、互いに対して同時に、リアルタイムで実行される必要がある。

【0004】

従来技術の埋め込みアプリケーションシステムは、代表的には、(1) ユーザインターフェース処理、高レベルアプリケーション処理、およびオペレーティングシステムを扱う、1 つ以上の汎用プロセッサ、例えば、ARM、MIP または x86 の種類の汎用プロセッサを、(2) 特定のインターフェースにおいて、または、特定のアプリケーション内で、リアルタイム／低待ち時間ベースで、特定のタイプの算術計算を扱う専用の 1 つ以上のデジタル信号プロセッサ (DSP) (メディアプロセッサを含む) と組み合わせる。DSP の代わりに、または DSP に加えて、特殊用途ハードウェアが、しばしば提供される。特殊用途ハードウェアは、例えば DSP が複数のアクティビティを一度に扱うことができないために、または DSP が非常に特化された計算素子のニーズを満たすことができないために DSP がプログラマブルベースで扱うことができない専用のニーズを扱う。

30

【0005】

40

従来技術のシステムにおける問題点は、ハードウェア設計の複雑さと、それに結び付けられた、プログラミングおよび異種の計算素子をインターフェースさせることにおけるソフトウェアの複雑さである。その結果として、しばしば、埋め込み処理サブシステムは、演算能力が足りないが、サイズ、コスト、および／または電力要件が過剰である。他の問題は、ハードウェアおよびソフトウェアの両方が、アプリケーションごとに再設計される必要があることである。さらに、従来技術のシステムは、バランスを良くしない。あるハードウェア素子からの他のハードウェア素子へと転送する能力は無い。

【発明の開示】

【発明が解決しようとする課題】

【0006】

50

本発明の目的は、デジタルデータ処理の改善されたデジタルデータプロセッサを提供することである。

【0007】

本発明のより具体的な目的は、高計算要件、リアルタイムアプリケーション要件、マルチメディア要件、音声およびグラフィカルユーザインターフェース、インテリジェンス、バックグラウンドタスクサポート、インタラクティビティ、ならびに／あるいは透過的インターネット、ネットワークング、および／またはP2Pアクセスサポートを有するアプリケーションをサポートする改善されたデジタルデータプロセッサを提供することである。関連する目的は、このような改善された装置および方法を、これらの要件を1つ以上有し、互いに同時に実行される、サポート複数アプリケーション集合として提供することである。

10

【0008】

本発明のさらなる目的は、現在および将来の機器における、計算、サイズ、電力およびコスト要件を満たす、処理を行う（埋め込み型、または他の型）、改善されたデジタルデータプロセッサを提供することである。このような機器の例には、デジタルテレビ、デジタルビデオレコーダ、ビデオおよび／またはオーディオプレーヤー、PDA、パーソナル・ナレッジ・ナビゲータ、携帯電話などが含まれるが、これらに限定されるものではなく、数例に過ぎない。

20

【0009】

本発明のさらに別の目的は、本質的に並列処理であるアプリケーションを含む、幅広い範囲のアプリケーションをサポートする改善されたデジタルデータプロセッサを提供することである。

【0010】

さらなる目的は、マルチメディアおよびユーザインターフェース駆動アプリケーションをサポートする改善されたデジタルデータプロセッサを提供することである。

30

【0011】

さらに別の目的は、例えば、ピア・ツウ・ピアマルチ処理を含む、1つ以上のレベルで、マルチタスクングおよびマルチ処理を行う、改善されたデジタルデータプロセッサを提供することである。

【0012】

低コスト、低電力、および／またはロバストな市場に出されるまでの時間が短い実装を提供する、デジタルデータプロセッサを提供することである。

【課題を解決するための手段】

40

【0013】

本発明によるデジタルデータプロセッサは、1つ以上のプロセッサと、該1つ以上のプロセッサ上で実行される1つ以上の仮想処理装置であって、各仮想処理装置は、1つ以上の処理またはスレッド（まとめて「スレッド」という）を実行し、各スレッドは、そのスレッドの寿命の間、同じ仮想処理装置および／または同じプロセッサで実行されるように制約されているか、制約されていないかのいずれかである、1つ以上の仮想処理装置と、該1つ以上のプロセッサと通信接続されているイベント伝達メカニズムとを備え、該イベント伝達メカニズムは、該1つ以上のプロセッサを形成するロジックとは独立したロジックであって、（i）1つ以上のイベントの各々をそれぞれのスレッドに、イベントとスレッドとの対応を示す情報に基づいて関連付け、

50

(i i) スレッドが仮想処理装置および／またはプロセッサに対して透過的に実行できるように、スレッドがいずれの仮想処理装置および／またはプロセッサ上で実行中であるかに関わらず、それぞれの該スレッドに、関連付けられたイベントを通知するロジックを備え、該スレッド同士は、メモリおよび該イベントを利用して、互いに通信するよう構成されており、これにより上記目的を達成する。

【0014】

前記イベントは、ハードウェア割り込み、ソフトウェア開始シグナリングイベント（「ソフトウェアイベント」）およびメモリイベントのいずれかを含むというさらなる改善点を有してもよい。

10

【0015】

少なくとも関連付けされた前記イベントを通知されたスレッドは、一時停止状態から実行状態に移行するというさらなる改善点を有してもよい。

【0016】

前記イベント伝達メカニズムは、イベント発生を伝えるイベントキューを含むというさらなる改善点を有してもよい。

【0017】

前記イベント伝達メカニズムは、イベントをスレッドにマッピングするテーブルを含み、前記イベントとスレッドとの対応を示す情報とは、該テーブルであるというさらなる改善点を有してもよい。

20

【0018】

前記テーブルは、イベントを、スレッドが該イベントの発生に応答するために、該スレッドにマッピングするというさらなる改善点を有してもよい。

【0019】

前記イベント伝達メカニズムは、ペンディングメモリ演算に関連のあるイベントと、該ペンディングメモリ演算を生じさせたメモリ参照命令を実行したそれぞれの前記スレッドとの間の関連を確立するテーブルを含み、前記イベントとスレッドとの対応を示す情報とは、該テーブルであり、該テーブルは、前記ペンディングメモリ演算のアドレス、状態情報、該メモリ参照命令を実行したスレッドのIDを保持するというさらなる改善点を有してもよい。

30

【0020】

前記イベント伝達メカニズムは、処理装置のうちの1つで実行中のシステムスレッドに、仮想処理装置に常駐しないスレッドに関連付けられたイベントの発生を通知し、該システムスレッドとは、システム特権およびアプリケーション特権のうち該システム特権を有するスレッドであり、該システムスレッドは、仮想処理装置に常駐しない該スレッドを一時停止状態から実行状態に移行させるというさらなる改善点を有してもよい。

40

【0021】

前記システムスレッドは、前記仮想処理装置で常に実行中ではなく、待機またはアイドル状態（一時停止状態）であるスレッドに関連付けられたイベントの発生を通知された際、該待機またはアイドル状態（一時停止状態）であるスレッドを一時停止状態から実行状態に移行させるというさらなる改善点を有してもよい。

【0024】

前記スレッドと通信接続されているイベントキューであって、該イベントキューは、少

50

なくとも1つの該スレッドによって該イベントキューに入れられた少なくともソフトウェア開始シグナリングイベント（「ソフトウェアイベント」）を含む前記イベントを伝える、イベントキューを備え、前記イベント伝達メカニズムは、該スレッドと、該イベントキューとに通信接続されているイベント伝達メカニズムであって、該イベント伝達メカニズムは、それぞれの該スレッドに、関連付けられた該イベントの発生を通知し、該イベントは、少なくともソフトウェアイベントを含み、イベントが通知される少なくとも1つのスレッドが、（i）該イベントキューにアクセスして通知を引き起こした発生を決定し、（ii）該イベントの発生を扱うこととによって、ソフトウェアイベントに応答してもよい。

10

【0025】

ソフトウェアイベントを前記イベントキューに入れる命令を実行する1つ以上のスレッドが、いずれのスレッドにイベントを通知するかを決定し、決定されたスレッドにイベント伝達メカニズムによりイベントを通知するというさらなる改善点を有してもよい。

【0038】

（要旨）

本発明によって達成される、これらの目的および他の目的は、ある局面において、1つ以上の仮想処理装置を含む仮想プロセッサを提供する。これらは、1つ以上のプロセッサ上で実行され、それぞれは、1つ以上のプロセスまたはスレッド（まとめて、「スレッド」という）を実行する。スレッドは、同じ仮想処理装置上で、それぞれの寿命を通じて実行されるように制約されてもよいが、必ずしもそうである必要はない。イベント伝達メカニズムは、イベントをそれぞれのスレッドに関連付け、そのスレッドがそのときに実行されている仮想処理装置および／またはプロセッサに関わらず、イベントが発生するときにこれらのスレッドに通知する。

20

【0039】

例えば、デジタルLCDテレビにおいて用いられる本発明による埋め込み型仮想プロセッサは、複数の仮想処理装置を実行するプロセッサモジュールを含み、それぞれが、デジタルLCD-TV動作（例えば、MPEGデマルチプレクス、映像デコード、ユーザインターフェース、オペレーティングシステムなど）のそれぞれの局面を扱うスレッドを処理する。イベント伝達メカニズムは、ハードウェア割り込み、ソフトウェアイベント（例えば、割り込みの性質をおびたソフトウェア開始イベント）、およびメモリイベントをそれぞれのスレッドに関連付ける。イベントが発生する場合、イベント伝達メカニズムは、そのときいずれの仮想処理装置を実行しているかに関わらず、イベントを適切なスレッドに伝達する。

30

【0040】

本発明の関連の局面は、選択されたスレッドが、一時停止状態から実行状態に移行することによってイベント伝達メカニズムからの通知に応答する、上述したような仮想プロセッサを提供する。上記の例を続けると、デジタルLCD-TV埋め込み型仮想プロセッサにおける仮想処理装置上で実行されるユーザインターフェーススレッドは、待機またはアイドルから、イベント伝達メカニズムによって伝達されるユーザキーパッド割り込みに応答して、実行中に移行し得る。

40

【0041】

本発明のさらなる関連の局面は、イベント伝達メカニズムが、仮想処理装置のうちの1つの上で実行されるシステムスレッドに、処理装置に常駐していないスレッドに関連付けられたイベントの発生を通知する、上述したような仮想プロセッサを提供する。システムスレッドは、スレッドを一時停止状態から実行状態に移行することによって、このような通知に応答し得る。

【0042】

本発明のさらなる関連の局面は、少なくとも選択されたアクティブスレッドが、互いに

50

同時に、かつ／または、オペレーティングシステムカーネルの介入なしに、このような通知のそれぞれに応答する、上述したような仮想プロセッサを提供する。

【0043】

本発明のさらなる局面は、イベント伝達メカニズムが、ペンディングメモリ演算と、このような演算の完了を待機する間に一時停止したそれぞれのスレッドとの間の関連を確立するペンディングメモリ演算テーブルを含む、上述したような仮想プロセッサを提供する。イベント伝達メカニズムは、メモリイベントを、全てのペンディングメモリ演算が完了したスレッドにシグナリングする。本発明の関連の局面は、少なくともハードウェア割り込みをスレッドにマッピングするイベントからスレッドへのルックアップテーブルを含む仮想プロセッサを提供する。

10

【0044】

さらに他の局面において、本発明は、1つ以上のスレッドは、ソフトウェアイベントをイベントキューにエンキューする命令を実行する、上述したような仮想プロセッサを提供する。関連の局面によると、命令を実行する1つ以上のスレッドが、いずれのスレッドにイベントが通知されるのかを指定する。

【0045】

本発明の他の局面は、スレッドのうちの少なくとも1つが、現在の命令シーケンスの実行を一時停止し、エラーハンドラを実行することによって、ハードウェア割り込みに応答する、上述したような仮想プロセッサを提供する。関連の局面において、そのスレッドは、エラーハンドラの実行の間、イベント通知を少なくとも一時的にディセーブルすることによってハードウェア割り込みに応答する。さらなる関連の局面において、そのスレッドは、エラーハンドラの実行に続く現在の命令シーケンスを一時停止することによって、ハードウェア割り込みに応答する。

20

【0046】

さらに他の局面において、本発明は、改善されたデータフローベースの同期化を有するデジタルデータプロセッサを提供する。このようなデジタルデータプロセッサは、複数のプロセッサおよび／またはスレッド（再度、まとめて「スレッド」という）と、それらのスレッドによってアクセス可能なメモリとを含む。少なくとも選択されたメモリ位置は、関連付けられた状態を有し、1つ以上のスレッドによるアクセスのために、データ項目を格納することが可能である。状態は、少なくともフル状態（Full）および空（Empty）状態を含む。選択されたスレッドは、選択されたメモリ位置を参照する第1のメモリ命令を実行する。選択された位置が空状態と関連付けられている場合、選択されたスレッドは選択された位置がフル状態と関連付けられるようになるまで一時停止する。

30

【0047】

本発明の関連の局面は、選択された位置がフル状態と関連付けられている場合、第1の命令の実行によって、選択された位置に格納されたデータ項目が選択されたスレッドに読み出され、選択された位置が空状態と関連付けられるようになる、改善されたこのようなデジタルデータプロセッサを提供する。本発明のさらに関連の局面によると、複数の実行中スレッドは、1つ以上の処理装置に常駐し、一時停止されたスレッドは、これらのスレッドに少なくとも一時的に存在しないようにされる。

40

【0048】

本発明は、さらなる局面において、選択されたスレッドまたは他のスレッドが、選択されたメモリ位置を参照する第2のメモリ命令を実行する、上述したようなデジタルデータプロセッサを提供する。選択された位置が空状態と関連付けられている場合、第2のメモリ演算の実行によって、選択されたデータが選択された位置に格納され、選択された位置がフル状態と関連付けられるようになる。

【0049】

さらに他の局面において、本発明は、メモリと、そのメモリにアクセスするスレッドを実行する1つ以上の仮想処理装置とを含む仮想プロセッサを提供する。選択されたスレッドは、メモリ内のある位置に向けられた第1のメモリ命令を実行する。その位置が空状態

50

と関連付けられている場合、命令の実行によって、その位置がフル状態と関連付けられるようになるまでスレッドが一時停止される。

【0050】

本発明の関連の局面は、上述したようなイベント伝達メカニズムをさらに含む上述したような仮想プロセッサを提供する。

【0051】

本発明のさらなる局面は、上述したような仮想プロセッサを1つ以上含む、デジタルLCDテレビ、デジタルビデオレコーダ(DVR)およびサーバ、MP3サーバ、携帯電話、ならびに／あるいは他のデバイスを提供する。本発明の関連の局面は、上述したような改善されたデータフロー同期化を有するプロセッサを含む、このようなデバイスを提供する。

10

【0052】

本発明のさらに他の局面は、上述した仮想プロセッサ、デジタルデータプロセッサおよびデバイスの演算を並列処理する方法を提供する。

【0053】

本発明のこれらの局面および他の局面は、図面および以下の説明に明らかである。

【発明の効果】

【0054】

本発明は、デジタルデータ処理の改善されたデジタルデータプロセッサを提供することができる。

20

【0055】

本発明はまた、高計算要件、リアルタイムアプリケーション要件、マルチメディア要件、音声およびグラフィカルユーザインターフェース、インテリジェンス、バックグラウンドタスクサポート、インタラクティビティ、ならびに／あるいは透過的インターネット、ネットワークング、および／またはP2Pアクセスサポートを有するアプリケーションをサポートする改善されたデジタルデータプロセッサを提供することができる。

【0056】

本発明はまた、現在および将来の機器における、計算、サイズ、電力およびコスト要件を満たす、処理を行う(埋め込み型、または他の型)、改善されたデジタルデータプロセッサを提供することができる。

30

【発明を実施するための最良の形態】

【0057】

本発明のより完全な理解は、図面を参照しながら達成され得る。

【0058】

(詳細な説明)

図1は、本発明のある実施例に従って構成され、動作するプロセッサモジュール5を示す。プロセッサモジュール5は、本明細書および添付の図面を通じて、場合によっては、「SEP」と呼ばれる。モジュールは、汎用プロセッサ、例えば、PC、ワークステーション、またはメインフレームコンピュータの基礎を提供し得るが、例示した実施形態は、埋め込みプロセッサとして利用されている。

40

【0059】

モジュール5は、単独で、または、1つ以上のこのようなモジュールと組み合わせて用いられ得るが、特に、計算要件が性質的に並行であり、複数の同時に実行されるアプリケーションおよび／または命令レベル並列処理の利点を受けるデバイスまたはシステムを含み得る。このようなデバイスまたはシステムには、リアルタイム要件を有するデバイスまたはシステム、マルチメディアアプリケーションが実行されるデバイスまたはシステム、ならびに／あるいは、イメージ、信号、グラフィックス、および／またはネットワーク処理のような高計算要件を有するデバイスまたはシステムが含まれ得る。モジュールはまた

50

、1つのプラットフォーム上の複数のアプリケーションの集積、例えば、アプリケーションの同時使用があるような場合にも適する。モジュールは、モジュールが埋め込まれているか、または、他の方法で組み込まれたデバイスおよび／またはシステムにわたって、また、ネットワーク（有線、無線、もしくはその他）、またはデバイスおよび／またはシステムを接続する他の媒体にわたってシームレスアプリケーション実行を提供する。さらに、モジュールは、ピア・ツウ・ピア（P2P）アプリケーション、また、ユーザインタラクティブ性を有するアプリケーションに適する。上記の説明は、モジュール5が適するアプリケーションおよび環境の広範囲にわたるリストであるとは意図されず、一例に過ぎない。

【0060】

モジュール5が埋め込まれ得るデバイスおよびシステムの例には、特に、デジタルLCD-TV、例えば、図24に示すタイプのデジタルLCD-TVが含まれる。デジタルLCD-TVにおいて、モジュール5は、システムオンチップ（SOC）構成に組み込まれる。（当然、モジュールは、シングルチップ上に組み込まれる必要はなく、複数チップ、1つ以上の回路基板、1つ以上の別々に実装されたデバイス、および／またはこれらの組合せに組み込まれ得ることが理解される。）さらなる例には、デジタルビデオレコーダ（DVR）およびサーバ、MP3サーバ、携帯電話、静止画および動画カメラを集積するアプリケーション、ゲームプラットフォーム、ユニバーサルネットワークディスプレイ（例えば、デジタルLCD-TV、ネットワークにつながれた情報／インターネット機器、ならびに、汎用アプリケーションプラットフォーム）、G3携帯電話、個人用情報端末、な

【0061】

モジュール5は、スレッド処理装置（TPU）10～20と、レベル1（L1）命令およびデータキャッシュ22および24と、レベル2（L2）キャッシュ26と、パイプライン制御28と、実行（または機能ユニット）30～38、すなわち、整数処理ユニット、浮動小数点処理ユニット、比較ユニット、メモリユニット、およびブランチユニットとを含む。ユニット10～38は、図に示すように接続され、以下により詳細に説明される。

【0062】

概要として、TPU10～20は、プロセッサモジュール5内で物理的に実装される仮想処理装置であり、任意の所与の瞬間で、1つ（以上）のプロセス（単数または複数）および／またはスレッド（単数または複数）（まとめて「スレッド」と呼ばれる）に結合され、処理する。TPUは、汎用レジスタ、プリディケートレジスタ、制御レジスタにおいて表される、それぞれのスレッドごとの状態を有する。TPUは、ランチ（launch）およびパイプライン制御等のハードウェアを共有し、このハードウェアは、各サイクルにおいて、任意の組み合わせのスレッドから5つまでの命令をランチする。図面に示されているように、TPUは、さらに実行ユニット30～38を共有し、この実行ユニット30～38は、独立して、何れのスレッドから来たかを知る必要なく、ランチした命令を実行する。

【0063】

さらなる概要として、例示したL2キャッシュ26は、スレッド処理装置10～20の全てによって共有されており、モジュール5が組み込まれるチップの内部（ローカル）と外部との両方で、命令およびデータを記憶装置に格納する。例示したL1命令およびデータキャッシュ22および24も、TPU10～20によって共有されており、上記のチップにとってローカルな記憶装置に基づく。（当然、他の実施形態において、レベル1およびレベル2キャッシュは、異なるように構成され得る。例えば、モジュール5に対して完全にローカルであってもよいし、完全に外部であってもよいし、または他の様態であってもよい。）

モジュール5の設計は、スケラブルである。2つ以上のモジュール5が、SOC設計または他の構成において「連動」されてもよく、そのことによって、アクティブスレッド

の数および処理能力全体が増大する。モジュール5によって用いられ、本明細書中で説明されるスレッドモデルに起因して、結果として増加したTPUは、ソフトウェアに対して透過性を有している。例示したモジュール5は、6個のTPUを有するが、他の実施形態はより多数（当然、より少ない数であってもよい）のTPUを有し得る。しかし、さらなる機能ユニットが提供され得る。例えば、サイクルごとにランチされる命令の数は、5個から10～15個、または、それより多い数まで増加され得る。L1およびL2キャッシュ構成についての以下の記載から明らかなように、これらの数も、スケールされ得る。

【0064】

例示したモジュール5は、Linuxをアプリケーションソフトウェア環境として用いる。マルチスレッドとの関連で、これは、リアルタイムおよび非リアルタイム用途が1つのプラットフォーム上で実行することを可能にする。これはまた、生産機能性を増大させるためのオープンソースソフトウェアおよびアプリケーションの活用を可能にする。さらに、これは、各種のプロバイダからのアプリケーションの実行を可能にする。

【0065】

（マルチスレッド）

上述したように、TPU10～20は、仮想処理装置であり、1つのプロセッサモジュール5内に物理的に実装される。TPU10～20は、それぞれ、所与の瞬間に、1つ（またはそれ以上）のスレッドに結びつけられ、処理する。スレッドは、幅広い用途を具現化し得る。デジタルLCD-TVにおいて有用な例には、例えば、MP EG2信号デマルチプレクス、MP EG2映像デコード、MP EG音声デコード、デジタルTVインターフェース演算、オペレーティングシステム実行（例えば、Linux）が含まれる。当然、これらの用途および／または他の用途が、デジタルLCD-TV、およびモジュール5が具現化され得る各種の他のデバイスおよびシステムにおいて有用であり得る。

【0066】

TPUによって実行されるスレッドは独立しているが、メモリおよびイベントを通じて通信し得る。プロセッサモジュール5のそれぞれのサイクルの間、命令は、実行または機能ユニット30～38を利用するために必要な数のアクティブ実行中スレッドによってランチされる。例示的な実施形態において、この点で、ラウンドロビンプロトコルは、それぞれのスレッドに「公正さ」を保証するために課せられる（しかし、他の実施形態においては、優先順位または他のプロトコルが、代わりにまたはそれに加えて用いられ得る）。1つ以上のシステムスレッドは、（例えば、アプリケーションを起動する、スレッドアクティブ化を容易にするなどのため）TPU上で実行され得るが、アクティブスレッドを実行するためにオペレーティングシステム介入は必要とされない。

【0067】

複数アクティブスレッド（仮想プロセッサ）をサポートする、基礎をなす理論的根拠は、以下の通りである。

【0068】

（機能的性質）

プロセッサごとの複数アクティブスレッドは、1つのマルチスレッドプロセッサが複数のアプリケーション、メディア、信号処理およびネットワークプロセッサに取って代わることを可能にする。また、複数アクティブスレッドは、アプリケーション、イメージ、信号処理およびネットワーク化に対応する複数スレッドが、低待ち時間および高性能で、動作し、同時に相互運用することを可能にする。コンテキストスイッチングおよびインターフェースオーバーヘッドは最小限にされる。MP4デコードのような1つのイメージ処理アプリケーション内でさえ、スレッドは、例えば、フレームnが作成されている間にフレームn+1のためのデータを準備するため、パイプライン方式で容易に同時に演算することができる。

【0069】

（パフォーマンス）

プロセッサごとの複数アクティブスレッドは、機能ユニットをより良好に利用し、メモ

10

20

30

40

50

りおよび他のイベント待ち時間を許容することによって、個々のプロセッサのパフォーマンスを高める。同時に実行するスレッドを4つまでサポートするため、パフォーマンスが2倍向上することも珍しくない。電力消費およびダイサイズの増大は、無視できる程度であり、電力単位あたりのパフォーマンスおよび価格パフォーマンス比は向上する。プロセッサごとの複数アクティブスレッドはまた、ブランチおよびキャッシュミスに起因するパフォーマンスの劣化を、このようなイベントの間、他のスレッドを実行することによって、低減する。さらに、プロセッサごとの複数アクティブスレッドは、殆どのコンテキストスイッチオーバーヘッドをなくし、リアルタイムアクティビティの待ち時間を減らす。さらに、プロセッサごとの複数アクティブスレッドは、一般的な、高性能イベントモデルをサポートする。

10

【0070】

(実装)

プロセッサごとの複数アクティブスレッドは、パイプラインおよび設計全体の簡略化につながる。他のスレッドが実行され得るので、複雑なブランチ予測の必要はない。これは、複数プロセッサチップに対して、1つのプロセッサチップのコストを下げることに つながり、他の複雑さが消去される場合、コストを下げる。これは、さらに、電力単位あたりのパフォーマンスを向上させる。

【0071】

図2は、従来のスーパースカラープロセッサによるスレッド処理を、例示したプロセッサモジュール5のスレッド処理と対比させた図である。2Aを参照すると、スーパースカ 20
ラープロセッサにおいて、(斜め線によって示される) 1つの実行中スレッドからの命令は、実行されているコード内の実際の並列処理および依存状態に基づいて、利用可能な実行装置上で実行されるように動的にスケジューされる。これは、平均的に、ほとんどの実行装置が、各サイクルの間、利用されることができないことを意味する。実行装置の数が増加するにつれて、利用率は、典型的には下がる。また、実行装置は、メモリシステムおよびブランチ予測ミス/待機の間、アイドル状態である。

【0072】

対照的に、2Bを参照すると、モジュール5において、複数スレッド(それぞれ異なる線パターンによって示される)からの命令は、同時に実行される。各サイクルで、モジュール5は、複数スレッドからの命令をスケジューリングして、利用可能な実行ユニットリ 30
ソースを最適に利用する。従って、実行装置利用およびパフォーマンス全体はより高く、同時にソフトウェア対して透過性を有している。

【0073】

(イベントおよびスレッド)

例示した実施形態において、イベントには、ハードウェア(またはデバイス)イベント、例えば、割り込みと、デバイスイベントと同等であるが、ソフトウェア命令によって開始されるソフトウェアイベントと、メモリイベント、例えば、キャッシュミスの完了またはメモリプロデューサーコンシューマ(フルー空)取引とがある。ハードウェア割り込みは、通常は、アイドルスレッド(例えば、ターゲットスレッドまたはターゲットグループ内のスレッド)によって扱われるデバイスイベントに変換される。ソフトウェアイベント 40
は、例えば、1つのスレッドが他のスレッドを直接ウェイク(wake)することを可能にするために、用いられ得る。

【0074】

各イベントは、アクティブスレッドに結合される。特定のスレッド結合がない場合、例示的な実施形態において、常にアクティブであるデフォルトシステムスレッドに結合する。そのスレッドは、その後、イベントを必要に応じて処理する。これには、仮想プロセッサ上の新たなスレッドのスケジューリングが含まれる。特定のスレッド結合がある場合、(イベント伝達メカニズムに関連して以下に説明される) ハードウェアまたはソフトウェアイベントの伝達の際、ターゲットスレッドは、アイドル状態から実行中に移行される。ターゲットスレッドが既にアクティブであり、実行中である場合、イベントは、処理のた 50

め、デフォルトシステムスレッドに向けられる。

【0075】

例示した実施形態において、スレッドは、キャッシュミスおよび同期化待機を含むメモリシステムストール（短期遮断）と、ブランチミス予測（非常に短期の遮断）と、明示的なイベントの待機（ソフトウェアまたはハードウェアのいずれかで生成される）と、およびシステムスレッド明示的ブロックアプリケーションスレッドとに起因する非実行状態（ブロック）になり得る。

【0076】

本発明の好ましい実施形態において、イベントは、物理プロセッサモジュール5および効率的な動的分散実行環境の基礎を提供するネットワークにわたって動作する。従って、例えば、デジタルLCD-TVあるいは他のデバイスまたはシステムにおいて実行されるモジュール5は、スレッドを実行し得、ネットワーク（無線、有線、またはその他）を介して動的に移行されるメモリ、あるいは、サーバまたは他の（遠隔）デバイスからの他のメディアを利用する。スレッドおよびメモリベースのイベントは、例えば、スレッドが、本明細書の原理に従って動作する任意のモジュール5上で透過的に実行し得ることを保証する。これは、例えば、モバイルデバイスが、ネットワーク接続された他のデバイスの電力の活用することを可能にする。これは、また、ピア・ツウ・ピアおよび複数スレッドアプリケーションの、遠隔のネットワーク接続されたデバイス上での透過的に実行を可能にする。利点には、パフォーマンスの向上、機能性の向上および電力消費の低下が含まれる。

【0077】

スレッドは、2つの特権レベル、すなわち、システムおよびアプリケーションで実行される。システムスレッドは、全ての状態のそのスレッドおよびプロセッサ内の他のスレッド全てにアクセスし得る。アプリケーションスレッドは、それ自体に対応する非特権状態にしかアクセスできない。デフォルトで、スレッド0は、システム特権で実行される。他のスレッドは、システム特権スレッドによって作成される場合に、システム特権に構成され得る。

【0078】

図3を参照すると、例示した実施形態において、スレッド状態は以下の通りである。

【0079】

（アイドル（またはノンアクティブ））

スレッドコンテキストはTPUにロードされ、スレッドは、命令を実行していない。アイドルスレッドは、例えば、ハードウェアまたはソフトウェアイベントが発生するときに、実行に移行する。

【0080】

（待機（またはアクティブ、待機））

スレッドコンテキストはTPUにロードされるが、現在命令を実行しない。待機スレッドは、メモリ命令が進むことを可能にし得る、発生を待っているイベント、例えば、キャッシュ動作が完了するときに、実行に移行する。

【0081】

（実行（またはアクティブ、実行））

スレッドコンテキストはTPUにロードされ、現在命令を実行している。スレッドは、例えば、そうではない場合には空／充填（Fill）（プロデューサーコンシューマメモリ）命令が完了しないので、メモリ命令がキャッシュ動作、例えばキャッシュミスを完了することを待機する必要があるときに、待機に移行する。スレッドは、イベント命令が実行されるときに、アイドルに移行する。

【0082】

TPUのそれぞれに関連付けられたスレッド有効ビット（またはフラグあるいは他のインジケータ）は、スレッドのTPUへのソフトウェアローディングおよびアンローディングのために、任意のスレッド状態を妨害することなく、スレッド実行をディセーブルする

10

20

30

40

50

【0083】

プロセッサモジュール5は、実行する命令の利用可能性に基づいて、アクティブスレッドにわたってバランスを良くする。モジュールはまた、均一にフルにされたスレッドそれぞれについて、命令キューを保持しようと試みる。従って、最もアクティブであり続けるスレッドが、ほとんどの命令を実行する。

【0084】

(イベント)

図4は、本発明の1実施例によるシステムにおけるイベント伝達メカニズムを示す。イベントがスレッドにシグナリングされ、スレッドは（現在実行状態にある場合）実行を一時停止し、デフォルトイベントハンドラを、例えば、仮想アドレス0x0において実行することによって認識する。

【0085】

例示的な実施形態において、特定のスレッドにシグナリングされ得る5つの異なるイベントタイプがある。

【0086】

【表1】

イベント	説明	スレッド伝達
スレッド待機タイムアウト	待機命令がスレッド _n によって実行されてからのタイムアウト値が経過	スレッド _n
スレッド例外	実行中の命令が例外をシグナリング	スレッド _n
HWイベント	ハードウェアデバイスによって生成されるイベント（例えば、割り込み）	スレッドルックアップに対するイベントによって規定されるスレッド _r
SWイベント	swイベント命令によってシグナリングされるイベント（例えば、sw割り込み）	命令がスレッドを指定する そのスレッドがアクティブではない場合、待機またはアイドルがデフォルトシステムスレッドに伝達される
メモリイベント	スレッド _n についての全てのペンディング状態のメモリ演算が完了	スレッド _n

例示したイベントキュー40は、ハードウェアデバイスによって提示されたイベントおよびソフトウェアベースのイベント命令（例えば、ソフトウェア「割り込み」）を、仮想イベント番号（VTN）およびイベント番号を含むタプルの形式で、段階的に行う。

【0087】

【表2】

63	32	31	16	15	4	3	2	1	0
		threadnum		eventnum				how	priv

【0088】

【表3】

ビット	フィールド	説明
0	priv	イベントが、 0. システム特権 1. アプリケーション特権 でシグナリングされる特権
1	how	スレッドがアイドル状態にない場合、イベントがどのようにシグナリングされるか指定する。スレッドがアイドル状態にある場合このフィールドは無視され、イベントは直接シグナリングされる 0. アイドル状態のスレッドを待機。キュー内のこのイベント以降の全てのイベントも待機する。 1. スレッドを直ちにトラップ
15:4	eventnum	このイベントのロジカル番号を指定する。このフィールドの値は、システム例外ステータスまたはアプリケーション例外ステータスレジスタの詳細フィールドにおいて取り込まれる
31:16	threadnum	このイベントがシグナリングされる論理スレッド数を指定する

当然、ハードウェアデバイスによって提示されるイベントおよびソフトウェア命令は、

他の形式で、かつ／または他の情報を含んで提示され得る。

【0089】

次に、イベントタプルは、受け取った順序で、イベントからスレッドへのルックアップテーブル（イベントテーブルまたはスレッドルックアップテーブルとも呼ばれる）42に送られる。イベントからスレッドへのルックアップテーブル42は、いずれのT P Uが指示されたスレッドを現在扱っているかを判定する。その後、イベント番号からなる「T P Uイベント」の形態でイベントが、イベントからスレッドへの伝達メカニズム44を介してT P U（および、それぞれのスレッド）に提示される。特定のイベントを扱うスレッドが依然として示されていない場合、対応するイベントは、T P Uのうちの1つにおいてアクティブであるデフォルトシステムスレッドに送られる。

10

【0090】

イベントキュー40は、ハードウェア、ソフトウェア、および／またはその組合せにおいて実施され得る。モジュール5によって代表される、埋め込み型システムオンチップ（S o C）実装において、キューは、必要なキュー機能を提供する、一連のゲートおよび専用バッファとして実装される。別の実施形態において、キューは、ソフトウェア（またはハードウェア）リンクリスト、アレイなどにおいて実装される。

【0091】

テーブル42は、ハードデバイスまたはイベント命令によって提示されるイベント番号（例えば、ハードウェア割り込み）と、イベントをシグナリングすることが好ましいスレッドとの間に確立される。可能な場合としては、以下の場合がある。

20

【0092】

イベント番号にエントリがない：デフォルトシステムスレッドにシグナリングする
スレッドに提示：スレッドが実行、アクティブ、またはアイドルである場合特定のスレッド数にシグナリングし、そうではない場合、指定されたシステムスレッドにシグナリングする

テーブル42は、アップデートされたイベントのスレッドへのマッピングを保持する、専用の、またはそうではない1つの格納領域であり得る。テーブルはまた、分散型、またはそうではない複数の格納領域を構成し得る。それにも関わらず、テーブル42は、ハードウェア、ソフトウェア、および／またはこれらの組合せに実装され得る。モジュール5によって代表される埋め込み型システムオンチップ（S o C）実装において、テーブルは、アップデートされたイベントのスレッドへのマッピングを保持する、専用格納領域上の「ハードウェア」ルックアップを行うゲートによって実装される。このテーブルはまた、例えば、スレッドがT P U 10～20に新たにロードされ、かつ／または、インアクティブにされ、T P U 10～20からアンロードされるにつれて、マッピングをアップデートするシステムレベル特権スレッドによって、ソフトウェアアクセス可能である。いくつかの実施形態において、テーブル42は、マッピングを保持する格納領域のソフトウェアベースルックアップによって実装される。

30

【0093】

イベントのスレッドへの伝達メカニズム44はまた、ハードウェア、ソフトウェア、および／またはこれらの組合せにおいて、実装され得る。モジュール5によって代表される埋め込み型S o C実装において、シグナリングされるイベントをT P Uキューにルーティングするメカニズム44はゲート（およびラッチ）によって実装される。T P Uキュー自体は、一連のゲート、および、伝達されたイベントを待ち行列に並べるための専用バッファ46～48として実装される。上述したように、別の実施形態において、メカニズム44は、必要な機能性を提供するソフトウェア（または他のハードウェア構造）において実装され、同様に、キュー46～48は、ソフトウェア（またはハードウェア）にリンクされたリスト、アレイなどにおいて実装される。

40

【0094】

例示的な実施形態において、ハードウェアおよびソフトウェアイベント（すなわち、ソフトウェア開始シグナリングイベント、または「ソフトウェア割り込み」）を処理するプ

50

ロシージャの概略は、以下の通りである。

【0095】

1. イベントは現在実行しているアクティブスレッドであるTPUにシグナリングされる

2. そのTPUは、アクティブスレッドの実行を一時停止する。例外ステータス、例外IPおよび例外メモリアドレス制御レジスタは、イベントのタイプに基づいて、イベントに対応する情報を示すように設定される。全てのスレッド状態が有効である。

【0096】

3. TPUは、イベントシグナリングが対応するスレッド装置についてディセーブルされた状態で、仮想アドレス0x0において、デフォルトイベントハンドラのシステム特権で実行を開始する。Gプレジスタ0～3はスクラッチレジスタを含み、プリディケートレジスタ0～1がイベントハンドラによってスクラッチレジスタとして利用され、システム特権である。慣習によって、GP[0]は、イベント処理スタックポインタである。

【0097】

4. イベントハンドラは、対応するスレッド実行装置のための再入および再イネーブルイベントをシグナリングできるように、十分な状態を保護する。

【0098】

5. その後、イベントハンドラはイベントを処理する。これは、イベントをSWベースキューに提示するだけでもよいし、何らかの他の行動を取ることであってもよい。

【0099】

6. その後、イベントハンドラは、状態を回復し、元のスレッドの実行に戻る。

【0100】

メモリ関連イベントは、ある程度までしか異なるように扱われない。ペンディング（メモリ）イベントテーブル（PET）50は、スレッドを実行から待機に移行するメモリ演算のためのエントリを（メモリ参照命令から）保持する。イベントからスレッドへのルックアップテーブル42と同様に実装され得るテーブル50は、参照を開始したペンディングメモリ演算、状態情報およびスレッドIDを保持する。PETにおけるエントリに対応して、メモリ演算が完了し、任意の他のペンディング動作がそのスレッドについてのPETにない場合、PETイベントは、対応するスレッドにシグナリングされる。

【0101】

例示した実施形態によるメモリイベント処理の概略は、以下の通りである。

【0102】

1. イベントは、現在アクティブスレッドを実行している装置への信号である

2. スレッドがアクティブ待機状態であり、イベントがメモリイベントである場合、スレッドは、アクティブ実行に移行し、現在のIPにおいて実行を継続する。そうではない場合、このイベントは無視される。

【0103】

図面にさらに示されるように、例示した実施形態において、スレッド待機はタイムアウトし、スレッド例外は、スレッドに直接シグナリングされ、イベントからスレッドへの伝達メカニズム44を介して送られない。

【0104】

（トラップ）

マルチスレッドおよびイベントの目的は、スレッドの通常のプログラム実行が、妨害されないようにすることである。発生するイベントおよび割り込みは、イベントを待機していた適切なスレッドによって扱われる。これが可能でない場合もあり、通常処理が割り込まれる必要がある。SEPは、この目的のため、トラップメカニズムをサポートする。イベントタイプに基づくアクションのリストは、システム例外ステータスレジスタにおいて列挙されるトラップの完全なリストと共に、以下に示される。

【0105】

10

20

30

40

【表 4】

イベントタイプ	スレッド状態	特権レベル	アクション
アプリケーションイベント	アイドル	アプリケーションまたはシステム	実行状態に移行することによってイベントを認識、アプリケーション特権
システムイベント	アイドル	アプリケーションまたはシステム	イベントを認識するシステムトラップ、実行状態に移行
アプリケーションイベント（アイドルを待機）	待機または実行	アプリケーションまたはシステム	イベントはアイドルになるまで待ち状態のまま
アプリケーションイベント（アイドルでない場合トラップ）	待機または実行	アプリケーションまたはシステム	イベントを認識するためのアプリケーショントラップ
システムイベント	待機または実行	アプリケーション	イベントを認識するためのシステムトラップ
システムイベント（アイドルを待機）	待機または実行	システム	イベントはアイドルになるまで待ち状態のまま
システムイベント（アイドルでない場合トラップ）	待機または実行	システム	イベントを認識するためのシステムトラップ
アプリケーショントラップ	いずれでもよい	アプリケーション	アプリケーショントラップ
アプリケーショントラップ	いずれでもよい	システム	システムトラップ
システムトラップ	いずれでもよい	アプリケーション	システムトラップ、システム特権レベル
システムトラップ	いずれでもよい	システム	システムトラップ

例示したプロセッサモジュール 5 は、トラップが発生する場合、以下のアクションをとる。

【0106】

1. 実行される次の命令を指定する I P（命令ポインタ）は例外 I Pレジスタにロードされる
2. 特権レベルは、例外 I Pレジスタのビット 0 に格納される
3. 例外タイプは、例外状態レジスタに格納される
4. 実行がメモリユニット命令に関連する場合、例外に対応するメモリアドレスは、例外メモリアドレスレジスタにロードされる
5. 現在の特権レベルはシステムに設定される
6. I P（命令ポインタ）はクリアされる（ゼロ）
7. 実行は I P 0 で開始する
（仮想メモリおよびメモリスistem）

例示したプロセッサモジュール 5 は、64ビット仮想アドレス（VA）空間と、64ビットシステムアドレス（SA）（標準的な物理アドレスと異なる特徴を有する）と、VAまたはSAがまばらに充填された仮想アドレスからシステムアドレスへの変換のセグメントモデルとを有する仮想メモリおよびメモリスistemアーキテクチャを利用する。

【0107】

オフチップメモリがDDR DRAMまたは他の形式のダイナミックメモリを利用し得るにも関わらず、TPU10～20によってアクセスされる全てのメモリは、キャッシュとして効率的に管理される。図1に戻ると、例示した実施形態において、メモリスistemは、2つのロジックレベルからなる。レベル1キャッシュは、最適な待ち時間および帯域幅のため、別個のデータキャッシュ24と命令キャッシュ22とに分割される。例示したレベル2キャッシュ26は、オンチップ部分およびレベル2拡張と呼ばれるオフチップ部分からなる。全体として、レベル2キャッシュは、個々のSEPプロセッサ（単数または複数）5用のメモリであり、複数SEPプロセッサ5が用いられる実装において、分散型の「全キャッシュ」メモリスistemに寄与する。当然、これらの複数のプロセッサは、同じメモリスistem、チップまたはバスを物理的に共有してもよく、例えば、ネットワークまたはその他を介して接続されてもよい。

【0108】

図5に、例示されたシステムにおいて用いられるVAからSAへの変換を示す。この変

10

20

30

40

50

換は、（例示した実施形態において）セグメントが可変サイズ、例えば、 $2^{24} - 2^{28}$ バイトであってもよい、セグメントベースで扱われる。SAは、メモリシステムにキャッシュされる。メモリシステムにあるSAは、キャッシュ $22/24$ 、 26 のレベルのうちの1つにエントリを有する。任意のキャッシュ（およびメモリシステム）にないSAは、事実上、メモリシステムにない。従って、メモリシステムは、ページ（およびサブページ）粒状度で、ソフトウェアおよびOSにとって自然な様式で、プロセッサ上のページテーブルのオーバーヘッドなしにまばらに充填される。

【0109】

上記の特徴に加えて、例示した実施形態の仮想メモリおよびメモリシステムアーキテクチャは、以下のさらなる特徴を有する。分散型共用型メモリ（DSM）、ファイル（DSF）、オブジェクト（DSO）およびピア・トゥ・ピア（DSP2P）の直接サポート、スケーラブルキャッシュおよびメモリシステムアーキテクチャと、スレッド間で共有され得るセグメントと、高速レベル1キャッシュとである。これは、何故なら、完全な仮想から物理アドレスへの変換または仮想キャッシュの複雑さがいない状態で、ルックアップがタグアクセスと並列であるからである。

【0110】

（仮想メモリの概要）

例示的なシステムにおける仮想アドレスは、メモリ参照およびブランチ命令によって構成される64ビットアドレスである。仮想アドレスは、セグメントあたりのベースで、全てのシステムメモリおよびIOデバイスへのアクセスに用いられるシステムアドレスに変換される。各セグメントのサイズは、 $2^{24} \sim 2^{48}$ バイトの間で変化する。より具体的には、図5を参照すると、仮想アドレス50は、図に示した状態で、セグメントテーブル52におけるエントリに一致させるために用いられる。一致されたエントリ54は、図に示す仮想アドレスのコンポーネントと組合せられた場合、対応するシステムアドレスを指定する。さらに、一致されたエントリ54は、対応するセグメントサイズおよび特権を指定する。次に、そのシステムアドレスは、システムメモリにマッピングする。システムメモリは、例示した実施形態において、まばらに埋められた 2^{64} バイトを含む。例示した実施形態は、システム特権で、スレッドによってディセーブルされるアドレス変換を可能にする。この場合、セグメントテーブルがバイパスされ、全てのアドレスは、低い32ビットへと切り捨てられる。

【0111】

例示したセグメントテーブル52は、スレッドあたり16～32エントリ（TPU）を含む。テーブルは、ハードウェア、ソフトウェア、および／またはこれらの組合せで実装され得る。モジュール5によって代表される、埋め込み型システムオンチップ（SOC）実装において、テーブルは、メモリにおける別々のエントリが各スレッド（例えば、スレッドあたり別のテーブル）に設けられる状態で、ハードウェアに実装される。セグメントは、同じシステムアドレスを指す各スレッドにつき、別々のエントリを設定することによって、2つ以上のスレッドの間で共有され得る。他のハードウェアまたはソフトウェア構造が、この目的のため、代わりに、またはこれに加えて用いられ得る。

【0112】

（キャッシュメモリシステム概要）

上述したように、命令およびデータ帯域幅を最大限にするため、レベル1キャッシュは、別々のレベル1命令キャッシュ22およびレベル1データキャッシュ24として構成される。

【0113】

図6を参照すると、オンチップL2キャッシュ26aは、タグおよびデータ部分からなる。例示した実施形態において、オンチップL2キャッシュ26aは、サイズが0.5～1Mバイトであり、128ブロックを有し、16方向に結合可能である。各ブロックは、拡張L2タグを格納するために64kバイトが提供された状態で、128バイトデータまたは16拡張L2タグを格納する。タグ内のタグモードビットは、拡張L2キャッシュに

ついて、データ部が16タグからなることを示す。

【0114】

拡張L2キャッシュ26bは、上述したように、DDR DRAMベースであるが、他のメモリアイプが用いられてもよい。例示した実施形態において、拡張L2キャッシュ26bは、サイズが1ギガバイトまでであり、256方向に結合可能であり、16kバイトページおよび128バイトサブページを有する。0.5mバイトのL2キャッシュ26aおよび1gバイトの拡張L2キャッシュ26bの構成について、L2拡張を完全に記述するために12%のオンチップL2キャッシュのみが必要である。より大きいオンチップL2またはより小さいL2拡張サイズのため、割合はより低くなる。L2キャッシュ（オンチップおよび拡張）を集めて、分散型SEPメモリシステムを作成する。

10

【0115】

例示した実施形態において、L1命令キャッシュ22およびL1データキャッシュ24は、8方向に結合可能であり、32kバイトおよび128バイトブロックを有する。図に示すように、レベル1キャッシュの両方は、レベル2キャッシュの適切なサブセットである。レベル2キャッシュは、オンチップおよびオフチップ拡張L2キャッシュからなる。

【0116】

図7は、L2キャッシュ26aにおいてタグルックアップを行って、L2キャッシュアドレス78に一致するデータブロック70を識別するために、例示した実施形態において用いられるL2キャッシュ26aおよびロジックを表す。例示した実施形態において、そのロジックは、16個のキャッシュタグアレイグループ72a~72pと、対応するタグ比較要素74a~74pと、対応するデータアレイグループ76a~76pとを含む。これらは、図示されているように、L2キャッシュアドレス78をグループタグアレイ72a~72pと比較し、これも図示されているように、示されたデータデータアレイグループ76a~76pによって識別されるデータブロック70を選択する。

20

【0117】

キャッシュタグアレイグループ72a~72p、タグ比較要素74a~74p、および対応するデータアレイグループ76a~76pは、ハードウェア、ソフトウェア、および/またはこれらの組み合わせに実装され得る。モジュール5によって代表される埋め込み型SoC実装において、これらは図19に示すように実装される。図19は、32x256シングルポートメモリセルにおいて実施されるキャッシュタグアレイグループ72a~72pと、128x256シングルポートメモリセルにおいて実施されるデータアレイグループ76a~76pとを示し、これらは全て図示されているように現在状態マシン制御ロジック190に接続されている。この素子は、図示されているように、本明細書と矛盾しない状態でL2キャッシュ装置26aの動作を容易にする状態マシン192と、L1命令およびデータキャッシュ22および24からのリクエストをバッファリングするリクエストキュー194とに接続されている。

30

【0118】

ロジック素子190は、さらに、L2キャッシュのオフチップ部分26bとのインターフェースを提供するDDR DRAM制御インターフェース26cに接続されている。ロジック素子190は、同様に、液晶ディスプレイ（LCD）、音声出力インターフェース、映像入力インターフェース、映像出力インターフェース、ネットワークインターフェース（無線、有線、またはその他）、記憶装置インターフェース、周辺機器インターフェース（例えば、USB、USB2）、バスインターフェース（PCI、ATA）を含むが、これらに限定されないAMBA適合コンポーネントに接続されている。DDR DRAM制御インターフェース26cおよびAMBAインターフェース26dは、同様に、図示されるように、L2データキャッシュバス198によってL1命令およびデータキャッシュへのインターフェース196に接続されている。

40

【0119】

図8は、同様に、L2拡張キャッシュ26bにおいてタグルックアップを行って、指定されたアドレス78に一致するデータブロック80を識別するために、例示した実施形態

50

において用いられるロジックを表す。例示した実施形態において、そのロジックは、データアレイグループ82a～82pと、タグ比較要素84a～84pと、タグラッチ86とを含む。これらは、図示されているように、L2キャッシュアドレス78をデータアレイグループ82a～82pと比較し、これも図示されているように、アドレス78の対応する部分に一致するグループのうちの1つからのタグを選択するように、示されているように接続されている。一致するタグからの物理的なページ数は、図示されているように、アドレス78のインデックス部と組合せられて、オフチップメモリ26bにおけるデータブロック80を識別する。

【0120】

データアレイグループ82a～82pおよびタグ比較要素84a～84pは、ハードウェア、ソフトウェア、および／またはこれらの組合せに実装され得る。モジュール5によって代表される、埋め込み型SoC実装では、これらは、必要なルックアップおよびタグ比較機能を提供する、ゲートおよび専用メモリにおいて実装される。他のハードウェアまたはソフトウェア構造が、この目的のため、代わりに、または追加して用いられ得る。

【0121】

以下は、例示した実施形態におけるL2およびL2Eキャッシュ動作を示す擬似コードである。

【0122】

L2Eタグルックアップ、ヒットする場合、L1キャッシュにデータで応答する、それ以外では、L2Eタグのルックアップ、ヒットする場合、

タグをL2に割り当てる；

L2Eデータにアクセスし、対応するL2エントリに格納する；

L1キャッシュにデータで応答する；

それ以外では、拡張L2Eタグのルックアップ

L2Eタグを割り当てる；

タグをL2に割り当てる；

L2Eデータにアクセス、対応するL2エントリに格納し、L1キャッシュにデータで応答する

(スレッド処理装置の状態)

図9を参照すると、例示した実施形態は、6個までのアクティブスレッドをサポートする6個のTPUを有する。それぞれのTPU10～20は、図9に示すように、汎用レジスタ、プリディケートレジスタおよび制御レジスタを含む。システムおよびアプリケーション特権レベルにおけるスレッドは、同一状態を含むが、いくつかのスレッド状態情報は、凡例およびそれぞれの線パターンによって示すように、システム特権レベルにおいてのみ可視である。レジスタに加えて、それぞれのTPUは、さらにペンディングメモリイベントテーブル、イベントキューおよびイベントからスレッドへのルックアップテーブルを含むが、これらは何れも図9には示されない。

【0123】

実施形態によっては、汎用レジスタの数は48（以下）～128（以上）個であってもよく、例示した実施形態は128個有する。プリディケートレジスタの数は24（以下）～64（以上）であってもよく、例示した実施形態は32個有する。アクティブスレッドの数は6（以下）～256（以上）であり、例示した実施形態は8個有する。ペンディングメモリイベントテーブルのエントリは、16（以下）～512（以上）個であり、例示した実施形態は16個有する。スレッドあたりのペンディングメモリイベントの数は、少なくとも2つであることが好ましい（しかし、より少ない可能性がある）。イベントキューは、（256（以上、または以下））である。イベントからスレッドへのルックアップテーブルのエントリは16（以下）～256（以上）個であり、例示した実施形態は、32個有する。

【0124】

(汎用レジスタ)

例示した実施形態において、各スレッドは、実装によって、128個までの汎用レジスタを有する。汎用レジスタ3～0（GP[3:0]）は、システム特権レベルにおいて可視であり、イベント処理の初期の段階において、イベントスタックポインタおよび動作しているレジスタに利用され得る。

【0125】

（プレディケーションレジスタ）

プリディケートレジスタは、汎用SEPプリディケートメカニズムの一部である。それぞれの命令の実行は、参照プリディケートレジスタの値に基づいて、条件付きである。

【0126】

SEPは、64個までの1ビットプリディケートレジスタをスレッド状態の一部として提供する。それぞれのプリディケートレジスタは、プリディケートと呼ばれるものを保持する。プリディケートは、比較命令の実行の結果に基づいて、1（真）に設定されるか、または0（偽）にリセットされる。プリディケートレジスタ3～1（PR[3:1]）はシステム特権レベルで可視であり、イベント処理の初期の段階の間、動作するプリディケートに利用され得る。プリディケートレジスタ0は、読み出し専用であり、常に1、すなわち、真と読み出される。実行を無条件にすることは命令によって行われる。

10

【0127】

（制御レジスタ）

（スレッド状態レジスタ）

【0128】

20

【表5】

63	20	19	18	17	16	15	8	6	5	4	3	2	1	0
		daddr	iaddr	align	endian	mod			Thread state	priv	tenable	atrapen	strapen	

【0129】

【表 6】

ビット	フィールド	説明	特権	単位	設計使用
0	strapen	システムトラップイネーブル。リセット時クリア。システムトラップのシグナリングは、このビットおよび atrapen を、再入のときにソフトウェアによって再度設定されるまでリセットする。 0 - システムトラップがディセーブルされる 1 - イベントがイネーブルされる	system_rw	スレッド	ブランチ
1	atrapen	アプリケーショントラップイネーブル。リセット時クリア。アプリケーショントラップのシグナリングは、このビットを、再入のときにソフトウェアによって再度設定されるまでリセットする。アプリケーショントラップは、特権レベルもアプリケーションレベルのときに、アプリケーションレベルとマークされたイベントによって引き起こされる。 0 - (イベントがスレッドに伝達されるときに) イベントがディセーブルされる 1 - イベントイネーブルされる	app_rw	スレッド	
2	lenable	スレッドイネーブル。スレッド 0 についてリセット時クリアが設定され、他の全てのスレッドについてクリアされる。 0 - スレッド演算がディセーブルされる 1 - スレッド演算がイネーブルされる	System_rw	スレッド	ブランチ
3	priv	特権レベル。リセット時クリア。 0 - システム特権 1 - アプリケーション特権	System_rw App_r	スレッド	ブランチ
5:4	state	スレッド状態。リセット時、スレッド 0 について「実行」に設定され、他の全てのスレッドについて「アイドル」に設定される。 0 - アイドル 1 - 予約済み 2 - 待機 3 - 実行	System_rw	スレッド	ブランチ
16:8	mod[7:0]	GPレジスタが変更される。リセット時にクリアされる ビット 8 レジスタ 0 ~ 15 ビット 9 レジスタ 16 ~ 31 ビット 10 レジスタ 32 ~ 47 ビット 11 レジスタ 48 ~ 63 ビット 12 レジスタ 64 ~ 79 ビット 13 レジスタ 80 ~ 95 ビット 14 レジスタ 96 ~ 111 ビット 15 レジスタ 112 ~ 127	App_rw	スレッド	パイプ
16	endian	エンディアンモード。リセット時クリア 0 - リトルエンディアン 1 - ビッグエンディアン	System_rw App_r	プロセッサ	メモリ
17	align	アライメントチェック。クリアの場合、並べられていないメモリ参照が許容される。設定される場合、並べられていないメモリ参照は全て、並べられていないデータのフォールトとなる。リセット時クリア。	System_rw App_r	プロセッサ	メモリ
18	iaddr	命令アドレス変換イネーブル。リセット時クリア。 0 - ディセーブルされる 1 - イネーブルされる	System_rw App_r	プロセッサ	ブランチ
19	daddr	データアドレス変換イネーブル。リセット時クリア。 0 - ディセーブルされる 1 - イネーブルされる	System_rw App_r	プロセッサ	メモリ

(I Dレジスタ)

【 0 1 3 0 】

【表 7】

63	32	39	32	31	16	15	8	7	0
				thread_id		id		type	

【 0 1 3 1 】

【表 8】

ビット	フィールド	説明	特権	単位
7:0	type	プロセッサタイプおよび改定 [7 : 0]	read only	プロセッサ
15:8	id	プロセッサ ID [7 : 0] - 仮想プロセッサ番号	read only	スレッド
31:16	thread_id	スレッド ID [15 : 0]	System_rw App_ro	スレッド

(命令ポインタレジスタ)

【0 1 3 2】

【表 9】

63	4	3	1	0
Doubleword			mask	0

10

次に実行される命令の 64 ビット仮想アドレスを指定する。

【0 1 3 3】

【表 10】

ビット	フィールド	説明	特権	単位
63:4	Doubleword	命令ダブルワードのアドレス	app	スレッド
3:2	mask	命令ダブルワード内のいずれの命令が実行されずに残っているかを示す。 ・ビット 1 - 第 1 の命令ダブルワードビット [4 0 : 0 0] ・ビット 2 - 第 2 の命令ダブルワードビット [8 1 : 4 1] ・ビット 3 - 第 3 の命令ダブルワードビット [1 2 2 : 8 1]	app	スレッド
0		常にゼロと読まれる	app	スレッド

20

(システム例外ステータスレジスタ)

【0 1 3 4】

【表 11】

63	4	15	4	3	0
			detail		etype

30

【0 1 3 5】

【表 1 2】

ビット	フィールド	説明	特権	単位
3:0	etype	例外タイプ 0. なし 1. イベント 2. タイマイイベント 3. システムコール 4. シングルステップ 5. 保護フォルト 6. 保護フォルト、システムコール 7. メモリ参照フォルト 8. SWイベント 9. HWイベント 10. その他	read only	スレッド
15:4	detail	フォルトの詳細ー以下の例外タイプに有効 ・メモリ参照フォルト詳細 (タイプ5) 1. なし 2. 充填されることを待機中 3. 空になることを待機中 4. キャッシュミスの完了を待機中 5. メモリ参照エラー ・イベント (タイプ1) - 12ビットイベント番号を指定する		

(アプリケーション例外ステータスレジスタ)

【0 1 3 6】

【表 1 3】

63	4	15	4	3	0
		detail		etype	

【0 1 3 7】

【表 1 4】

ビット	フィールド	説明	特権	単位
3:0	etype	例外タイプ 0. なし 1. イベント 2. タイマイイベント 3. システムコール 4. シングルステップ 5. 保護フォルト 6. 保護フォルト、システムコール 7. メモリ参照フォルト 8. SWイベント 9. HWイベント 10. その他	read only	スレッド
15:4	detail	保護フォルトの詳細ー以下の例外タイプに有効 ・イベント (タイプ1) - 12ビットイベント番号を指定する		

(システム例外 I P)

【0 1 3 8】

【表 1 5】

63	4	3	1	0
	Doubleword	mask	priv	

システム特権へのシグナリングされた例外に対応する命令のアドレス。ビット [0] は例外のときの特権レベルである。

【0 1 3 9】

【表 1 6】

ビット	フィールド	説明	特権	単位
63:4	Doubleword	例外をシグナリングする命令ダブルワードのアドレス	system	スレッド
3:1	mask	命令ダブルワード内のいずれの命令が実行されずに残っているかを示す。 ・ビット1－第1の命令ダブルワードビット [4 0 : 0 0] ・ビット2－第2の命令ダブルワードビット [8 1 : 4 1] ・ビット3－第3の命令ダブルワードビット [1 2 2 : 8 1]	system	スレッド
0	priv	例外の場合のスレッドの特権レベル	system	スレッド

10

シグナリングされた例外に対応する命令のアドレス。ビット [0] は例外のときの特権レベルである。

【0 1 4 0】

(アプリケーション例外 I P)

【0 1 4 1】

【表 1 7】

63	4	3	1	0
Doubleword		mask		

20

アプリケーション特権へのシグナリングされた例外に対応する命令のアドレス。

【0 1 4 2】

【表 1 8】

ビット	フィールド	説明	特権	単位
63:4	Doubleword	例外をシグナリングされた命令ダブルワードのアドレス	system	スレッド
3:1	mask	命令ダブルワード内のいずれの命令が実行されずに残っているかを示す。 ・ビット1－第1の命令ダブルワードビット [4 0 : 0 0] ・ビット2－第2の命令ダブルワードビット [8 1 : 4 1] ・ビット3－第3の命令ダブルワードビット [1 2 2 : 8 1]	system	スレッド

30

シグナリングされた例外に対応する命令のアドレス。ビット [0] は例外のときの特権レベルである。

【0 1 4 3】

(例外メモリアドレス)

【0 1 4 4】

【表 1 9】

63	0
Address	

40

例外をシグナリングされるメモリ参照のアドレス。メモリフォルトの場合のみ有効である。例外ステータスレジスタが、メモリ参照フォルト、充填になることを待機、または空になることを待機していることを示す場合にのみ、ペンディングメモリ演算のアドレスを保持する。

【0 1 4 5】

(命令セグメントテーブルポインタ (I S T P)、データセグメントテーブルポインタ

50

(DSTP))

【0146】

【表20】

63	32	31	6	5	1	0
		reserved		ste number		field

I STEおよびI STEレジスタによって利用されて、読み出されるか、または書き込まれるsteおよびフィールドを指定する

10

【0147】

【表21】

ビット	フィールド	説明	特権	単位
0	field	セグメントテーブルエントリの低(0)または高(1)部分を指定する。	system	スレッド
5:1	ste number	STEデータレジスタに読み込まれるSTE番号を指定する。	system	スレッド

(命令セグメントテーブルエントリ(I STE)、データセグメントテーブルエントリ(DSTE))

【0148】

【表22】

20

63	0
data	

読み出しのとき、I STEによって指定されるSTEが宛先汎用レジスタに入れられる。書き込みのとき、I STEまたはDSTEによって指定されるSTEが汎用ソースレジスタから書き込まれる。セグメントテーブルエントリのフォーマットは、変換テーブル構成およびエントリ記述というタイトルの第6章に記載される。

30

【0149】

(命令またはデータレベル1キャッシュタグポインタ(ICTP、DCTP))

【0150】

【表23】

63	32	15	14	13	7	6	2	1	0
		reserved		index		bank			

ICTEまたはDCTEによって読み出されるか、書き込まれる命令キャッシュタグエントリを指定する。

40

【0151】

【表24】

ビット	フィールド	説明	特権	単位
6:2	bank	レベル1キャッシュタグエントリから読み出されるバンクを指定する。第1のインプリメンテーションは、0x0~fの有効バンクを有する	system	スレッド
13:7	index	レベル1キャッシュタグエントリから読み出されたバンク内のインデックスアドレスを指定する。	system	スレッド

(命令またはデータレベル1キャッシュタグエントリ(ICTE、DCTE))

【0152】

50

【表 2 5】

63	0
data	

読み出しのとき、ICTPまたはDCTPによって指定されるキャッシュタグが宛先汎用レジスタに入れられる。書き込みのとき、ICTPまたはDCTPによって指定されるキャッシュタグが汎用ソースレジスタから書き込まれる。キャッシュタグエントリのフォーマットは、変換テーブル構成およびエントリ記述というタイトルの第6章に記載される。

【0153】

(イベントキュー制御レジスタ)

【0154】

【表 2 6】

63	32	31	19	18	17	2	1	0
		address		offset	event			reg_op

イベントキュー制御レジスタ(EQCR)は、イベントキューへの標準および診断アクセスをイネーブルする。レジスタを用いるシーケンスは、レジスタ読み出しが後に続くレジスタ書き込みである。reg_opフィールドの内容は、書き込みおよび次の読み出しのための動作を指定する。実際のレジスタ変更または読み出しは、書き込みによってトリガされる。

【0155】

【表 2 7】

ビット	フィールド	説明	特権	単位
1:0	reg_op	その回の書き込みおよび次の読み出しのためのレジスタ演算を指定する。レジスタ読み出しについて有効。 0-読み出し 1-書き込み 2-キューにプッシュ 3-キューからポップ	system	プロセッサ
17:2	event	書き込みとプッシュについて、キューに書き込まれるかプッシュされるイベント番号を指定する。読み出しおよびポップについて、演算は、キューから読み出されるかポップされるイベント番号を含む	system	プロセッサ
18	empty	キューが現在の演算の前に空であったか否かを示す	system	プロセッサ
31:19	address	読み出しおよび書き込みキュー動作についてアドレスを指定する。アドレスフィールドは、プッシュおよびポップ動作については何も行わない	system	プロセッサ

(イベントスレッドルックアップテーブル制御)

【0156】

【表 2 8】

63	9	48	41	40	33	32	17	16	1	0
		address		thread		mask		event		reg_op

イベントからスレッドへのルックアップテーブルは、ハードウェアデバイスまたはイベント命令によって提示されるイベント番号と、イベントをシグナリングすることが好ましいスレッドとの間のマッピングを確立する。テーブル内の各エントリは、ビットマスクを有するイベント番号、およびイベントがマッピングされる対応するスレッドを指定する。

【0157】

【表29】

ビット	フィールド	説明	特権	単位
0	reg_op	その回の書き込みおよび次の読み出しのためのレジスタ演算を指定する。レジスタ読み出しについて有効。 0－読み出し 1－書き込み	system	プロセッサ
16:1	event[15:0]	書き込みについて、キューに書き込まれるイベント番号を指定する。読み出し動作について、動作は、指定されたテーブルアドレスにイベント番号を含む	system	プロセッサ
32:17	mask[15:0]	対応するイベントビットが有効であるか否かを指定する。 0－有効 1－何も行わない	system	プロセッサ
40:33	thread			
48:41	address	読み出しおよび書き込み動作についてテーブルアドレスを指定する。アドレスフィールドは、プッシュおよびポップ動作については何も行わない	system	プロセッサ

10

(タイマおよびパフォーマンスモニタ)

例示した実施形態において、全てのタイマおよびパフォーマンスモニタレジスタは、アプリケーション特権においてアクセス可能である。

【0158】

(クロック)

20

【0159】

【表30】

63	0
clock	

【0160】

【表31】

ビット	フィールド	説明	特権	単位
63:0	clock	プロセッサがリセットされてからのクロックサイクル数	app	プロセッサ

30

(実行される命令)

【0161】

【表32】

63	32	31	0
count			

40

【0162】

【表33】

ビット	フィールド	説明	特権	単位
31:0	count	実行される命令の数の飽和カウント。読み出し時にクリアされる。全ての1の値は、カウントがオーバーフローしたことを示す。	app	スレッド

(スレッド実行クロック)

【0163】

【表 3 4】

63	32	31	0
active			

【0 1 6 4】

【表 3 5】

ビット	フィールド	説明	特権	単位
31:0	active	スレッドがアクティブ実行中状態にある場合のサイクル数の飽和カウント。読み出し時にクリアされる。全ての1の値は、カウントがオーバーフローしたことを示す	app	スレッド

10

(待機タイムアウトカウンタ)

【0 1 6 5】

【表 3 6】

63	32	31	0
timeout			

20

【0 1 6 6】

【表 3 7】

ビット	フィールド	説明	特権	単位
31:0	timeout	タイムアウトイベントがスレッドにシグナリングされるまでに残ったサイクル数のカウント。クロックサイクルごとに1ずつ減る。	app	スレッド

(仮想プロセッサおよびスレッドID)

例示した実施形態において、各アクティブスレッドは、仮想プロセッサに対応し、8ビットアクティブスレッド数（アクティブスレッド [7 : 0]）によって指定される。モジュール5は、スレッドの高速ローディング（アクティブ化）およびアンローディング（ディアクティブ化）を可能にするように、16ビットのスレッドID（スレッド [15 : 0]）をサポートする。他の実施形態は、異なるサイズのスレッドIDをサポートし得る。

30

【0 1 6 7】

(スレッドー命令フェッチの抽象化)

上述したように、モジュール5のTPU10～20は、L1命令キャッシュ22と、これらのTPUからサイクルごとに5個までの命令をランチするパイプライン制御ハードウェアとを共有する。図10は、モジュール5によって利用される、機能ユニット30～38で実行するための命令をフェッチおよびディスパッチするメカニズムを抽象化した図である。

40

【0 1 6 8】

図に示すように、各サイクルの間、命令は、L1キャッシュ22からフェッチされ、TPU10～20にそれぞれ関連付けられた命令キュー10a～20aに入れられる。これは、サイクルのフェッチ段階と呼ばれる。例示した実施形態において、スレッドキュー10a～20aを等しいレベルに保つという保持するという全体的な目標で、3～6個の命令が1つのスレッドのそれぞれについてフェッチされる。他の実施形態において、異なる数の命令がフェッチされ、かつ／または、キューの相対的な充填について異なるターゲットが設定される。また、フェッチ段階の間、モジュール5（および、具体的には、例えば、上記のイベントを扱うメカニズム）は、イベントおよび待機から実行へのスレッドに対応する移行を認識する。

50

【0169】

フェッチおよび実行／回収段階と並列に実行するディスパッチ段階の間、1つ以上の実行中スレッドからの命令は、そのサイクルについてのリソースの最適の利用を考慮に入れるラウンドロビンプロトコルに基づいて、機能ユニット30～38にディスパッチされる。これらの命令は、スレッドに任意の組合せからのものであり得る。コンパイラは、例えば、命令セットにおいて提供される「ストップ」フラグ、すなわち、1つのサイクル内でランチされ得るスレッド内の命令のグループ間の境界の利用を指定する。他の実施形態において、他のプロトコルが利用され得る。例えば、ある特定のスレッドを優先させるプロトコル、リソース利用を無視するプロトコルなどである。

【0170】

フェッチおよびディスパッチ段階と並列に実行する実行および回収段階の間、複数の命令が1つ以上のスレッドから同時に実行される。上述したように、例示した実施形態において、各サイクルの間、例えば、整数、浮動小数点、ブランチ、比較、およびメモリ機能ユニット30～38によって、5個までの命令がランチされ、実行される。他の実施形態において、例えば、機能ユニットの数およびタイプに依存して、かつ、TPUの数に依存して、より多い命令またはより少ない命令がランチされてもよい。

【0171】

命令は、実行後に完了していれば回収される。結果が書き込まれ、命令は命令キューからクリアされる。

【0172】

他方で、命令がブロックされる場合、対応するスレッドは、実行から待機に移行する。対応するスレッドについてブロックされた命令およびその後続く全ての命令は、その後、ブロックを引き起こした条件が解決されたときに再開される。図11は、これを容易にするために例示した実施形態において用いられる3ポインタキュー管理メカニズムを示す。

【0173】

その図を参照すると、命令キューおよび3ポイントのセットがTPU10～20のそれぞれについて維持される。ここでは、このようなキューのうちの1つ110およびポイントのセット112～116のみが示されている。キュー110は、関連するTPU、より具体的には、そのTPUにおいて現在アクティブなスレッドについて、フェッチされ、実行中であり、回収された（または無効にされた）命令を保持する。命令がフェッチされるにつれて、挿入（またはフェッチ）ポインタ112によって示されるキューの上部に挿入される。実行の次の命令は、抽出（または発行）ポインタ114によって識別される。コミットポインタ116は、実行がコミットされた最後の命令を識別する。命令がブロックされるか、他の理由で中断される場合、コミットポインタ116は、実行パイプラインにおいて、コミットポインタと抽出ポインタとの間の命令を破棄するためにロールバックされる。反対に、ブランチが取られる場合キュー全体はフラッシュされ、ポインタがリセットされる。

【0174】

キュー110は円形に示されているが、他の構成も用いられ得ることが理解される。図11に示す待ち行列に並べるメカニズムは、例えば、図12に示すように実装され得る。命令は、デュアルポートメモリ120、または直列のレジスタ（図示せず）に格納される。それぞれの新たにフェッチされた命令が格納される書き込みアドレスは、（例えば、パイプラインコントロールによって発行される）フェッチコマンドに応答するフェッチポインタロジック122によって供給されて、メモリ120のための後続のアドレスが発生する。発行された命令は、他のポート、ここでは、下部に示されているポートから取られる。各命令が取られる読み出しアドレスは、発行／コミットポインタロジック124によって供給される。そのロジックは、（例えば、パイプラインコントロールによって発行される）コミットおよび発行コマンドに応答して、後続のアドレスを発生させ、かつ／または、必要に応じてリセットする。

10

20

30

40

50

【0175】

(プロセッサモジュール実装)

図13は、特に、TPU10～20を実装するロジックを含む、図1のモジュール5のSoC実装を表す。図1に示すように図13の実装は、L1およびL2キャッシュ22～26を含む。L1およびL2キャッシュ22～26は、上述したように構成され、動作する。同様に、実装は、整数ユニット、浮動小数点ユニット、および比較装置をそれぞれ含む機能ユニット30～34を含む。さらなる機能ユニットが、代わりに、またはそれに加えて設けられてもよい。TPU10～20を実装するためのロジックは、パイプラインコントロール130、ブランチユニット38、メモリユニット36、レジスタファイル136、およびロードー格納バッファ138を含む。図13に示すコンポーネントは、図に示すように、制御および情報転送のため相互接続される。破線は、主な制御を示し、細い実線は、プリディケート値制御を示し、より太い実線は、64ビットデータバスを識別し、さらに太い線は、128ビットデータバスを示す。図13が本発明によるプロセッサモジュール5のある実装を表し、他の実装も実現され得ることが理解される。

10

【0176】

(パイプラインコントロールユニット)

例示した実施形態において、パイプラインコントロール130は、図11～12を参照しながら上述したスレッドごとのキューを含む。1スレッドあたり、12、15または18個の命令があるとパラメータ化され得る。コントロール130は、それらのキューからラウンドロビンベースで（これは他の基準でも行われることも注記しておく）命令を取り上げる。コントロール130は、レジスタファイル136（命令のソースおよび宛先レジスタを提供するリソースである）、および機能ユニット30～38へのアクセスのシーケンスを制御する。パイプラインコントロール130は、スレッドごとのキューからの基本的な命令クラスをデコードし、命令を機能ユニット30～38にディスパッチする。上述したように、1つ以上のスレッドからの複数の命令が、同じサイクル内の機能ユニットによって、実行のためスケジューリングされ得る。コントロール130は、さらに、ブランチユニット38がスレッドごとの構成キューを空にしたときにブランチユニット38にシグナリングする役割も果たし、例えば、サイクルごとのベースで消費を低減することが可能な場合機能ユニットをアイドルにする役割も果たす。

20

【0177】

図14は、パイプラインコントロールユニット130のブロック図である。このユニットは、図に示されているように互いに（かつ、モジュール5の他のコンポーネントに）接続されている、スレッドクラスキューのためのコントロールロジック140と、スレッドクラス（スレッドあたりの）キュー142自体と、命令ディスパッチ144と、ロングワードデコードユニット146と、機能ユニットキュー148a～148eとを含む。スレッドクラス（スレッドごとの）キューは、図11～12を参照しながら上述したように、構成され、動作する。スレッドクラスキュー制御ロジック140は、それらのキュー142の入力側を制御し、従って、図11～12を参照しながら上述した挿入ポインタ機能性を提供する。制御ロジック140はまた、ユニットキュー148a～148eの入力側を制御する役割も果たし、命令フェッチングを制御するためにブランチユニット38とインターフェースする役割も果たす。この後者の点において、ロジック140は、上述した状態で（例えば、ほとんどの命令を回収するTPUを補償するように）、命令フェッチングのバランスを取る役割を果たす。

30

40

【0178】

命令ディスパッチ144は、各サイクルで、スレッドクラスキューのそれぞれにおいて利用可能な命令のスケジュールを評価し、判定する。上述したように、例示した実施形態において、キューは、命令をより素早く回収するキューを考慮に入れながら、ラウンドロビンベースで扱われる。命令ディスパッチ144はまた、スレッドクラスキュー142の出力側を制御する。この点において、命令ディスパッチ144は、図11～12を参照しながら上述した抽出およびコミットポインタを管理する。これには、コミットポインタ巻

50

き戻し命令が回収されたときのアップデートおよび（例えば、スレッドスイッチまたは例外のため）命令が中断される場合のそのポインタのロールバックが含まれる。

【0179】

ロングワードデコードユニット146は、L1命令キャッシュ22から入来する命令ロングワードをデコードする。例示した実施形態において、それぞれのこのようなロングワードは、命令にデコードされる。これは、1つまたは2つのロングワードのデコードのためにパラメータ化され、ロングワードは、それぞれ、3つおよび6つの命令にデコードされる。デコードユニット146はまた、各命令の命令クラスをデコードする役割を果たす。

【0180】

ユニットキュー148a~148eは、機能ユニット30~38によって実行される実際の命令を待ち行列に並べる。各キューは、スレッド単位のベースで構成され、そのクラスキューと矛盾しないように保たれる。ユニットキューは、上述したように、スレッドクラスキューコントロール140と、制御の目的のために命令ディスパッチ144に接続されている。キュー148a~148eからの命令は、機能ユニット30~38への途中である、対応するパイプライン150a~150bに転送される。命令はまた、レジスタファイルパイプライン152に送られる。

【0181】

図15は、個々のユニットキュー、例えば、148aのブロック図である。これは、TPUにつき、1つの命令キュー154a~154eを含む。これらは、制御の目的で、スレッドクラスキューコントロール140（tcqueue_ctlと表示される）と、命令ディスパッチ144（idispatchと表示される）とに接続される。これらはまた、図示されているように、命令入力のためロングワードデコードユニット146（lwdecodeと表示される）に接続され、スレッド選択ユニット156に接続される。スレッド選択ユニットは、図示されているように命令ディスパッチ144によって提供される制御信号に基づいて、スレッド選択を制御する。ユニット156からの出力は、対応するパイプライン150a~150eと、レジスタファイルパイプライン152とにルーティングされる。

【0182】

再度図14を参照すると、整数ユニットパイプライン150aおよび浮動小数点ユニットパイプライン150bは、それぞれの機能ユニットに適した命令フィールドをデコードする。各パイプラインはまた、機能ユニットへのコマンドのタイミングを調節する。さらに、各パイプライン150aおよび150bは、ブランディングまたは中断に基づいて、それぞれのパイプラインを閉める（squash）。さらに、各パイプラインは、サイクルの間用いられていないときに、それぞれの機能ユニットに電力低下信号を印加する。例示した比較ユニットパイプライン150c、ブランチユニットパイプライン150d、およびメモリユニットパイプライン150eは、それぞれの機能ユニット、すなわち、比較ユニット34、ブランチユニット38、およびメモリユニット36と同様の機能性を提供する。レジスタファイルパイプライン150はまた、レジスタファイル136と同様の機能性を提供する。

【0183】

再度図13を参照すると、例示したブランチユニット38は、命令アドレス生成およびアドレス変換と、命令フェッチングとを行う。さらに、ブランチユニット38は、スレッド処理装置10~20の状態を維持する。図16は、ブランチユニット38のブロック図である。ブランチユニット38は、図に示されているように互いに（かつモジュール5の他のコンポーネントと）接続されている、制御ロジック160、スレッド状態記憶装置162a~162e、スレッドセクタ164、アドレス加算器166、セグメント変換内容アドレス可能メモリ（CAM）168を含む。

【0184】

制御ロジックデバイス160は、パイプラインコントロール130からのコマンド信号

10

20

30

40

50

に基づいて、ユニット 38 を駆動する。制御ロジックデバイス 160 はまた、図示されているように、命令キャッシュ 22 および L2 キャッシュ 26 受信通知を入力として受け取る。ロジック 160 は、図示されているように、スレッドスイッチをパイプラインコントロール 130 に出力し、コマンドを命令キャッシュ 22 および L2 キャッシュに出力する。スレッド状態記憶装置 162a~162e は、それぞれの TPU 10~20 について、スレッド状態を格納する。TPU のそれぞれについて、スレッド状態記憶装置 162a~162e は、図 3 に示し、上述された、汎用レジスタと、プリディケートレジスタと、制御レジスタとを維持する。

【0185】

スレッド状態記憶装置から取得されたアドレス情報は、図示されているように、スレッドセクタにルーティングされる。スレッドセクタは、そこからスレッドアドレスを選択し、アドレス計算が、コントロール 160 からの制御信号（図示されている）に基づいて行われる。アドレス加算器 166 は、図示されているように、選択されたアドレスを増分させるか、または、スレッドセクタ 164 の出力およびレジスタファイル（レジスタソースと呼ばれる）によって供給されるアドレッシング情報に基づいて、ブランチアドレス計算を行う。さらに、アドレス加算器 166 は、ブランチ結果を出力する。新たに計算されるアドレスは、セグメント変換メモリ 168 にルーティングされ、セグメント変換メモリ 168 は、図 5 を参照しながら上述したように動作し、次の命令フェッチと共に用いられる変換された命令キャッシュアドレスを生成する。

【0186】

（機能ユニット）

再度図 13 を参照すると、メモリユニット 36 は、データキャッシュ 24 アドレス生成およびアドレス変換を含む、メモリ参照命令実行を行う。さらに、ユニット 36 は、上述したように、ペンディング（メモリ）イベントテーブル（PET）50 を維持する。図 17 は、メモリユニット 36 のブロック図である。メモリユニット 36 は、図に示されているように互いに（かつ、モジュール 5 の他のコンポーネントに）接続されている、制御ロジック 170、アドレス加算器 172、およびセグメント変換内容アドレス可能メモリ（CAM）174 を含む。

【0187】

制御ロジック 170 は、パイプラインコントロール 130 からのコマンド信号に基づいて、ユニット 36 を駆動させる。制御ロジック 170 はまた、図示されているように、データキャッシュ 22 状態および L2 キャッシュ 26 受信確認を入力として受け取る。ロジック 170 は、図示されているように、スレッドスイッチをパイプラインコントロール 130 およびブランチユニット 38 に出力し、コマンドをデータキャッシュ 24 および L2 キャッシュに出力する。アドレス加算器 172 は、レジスタファイル 136 から提供されたアドレッシング情報を増分させるか、または必要なアドレス計算を行う。新たに計算されるアドレスは、セグメント変換メモリ 174 にルーティングされ、セグメント変換メモリ 174 は、図 5 を参照しながら上述したように動作し、データアクセスとともに用いられる変換された命令キャッシュアドレスを生成する。図面に示されていないが、ユニットはまた、上述したように、PET を含む。

【0188】

図 18 は、L1 命令キャッシュ 22 または L2 データキャッシュ 24 のいずれかを実装するキャッシュユニットのブロック図である。ユニットは、データアレイとして機能する、16 個の 128 x 256 バイトシングルスポートメモリセルを、タグアレイとして機能する、16 個の対応する 32 x 56 バイトデュアルポートメモリセル 182a~182p とともに含む。制御ロジック 184 および 186 は、図示されているように、メモリセルと、L1 キャッシュコントロールおよび L2 キャッシュコントロールとに接続されている。

【0189】

再度図 13 を参照すると、レジスタファイル 136 は、機能ユニット 30~38 によって実行されている命令によってアクセスされる、全てのソースおよび宛先レジスタのリソ

10

20

30

40

50

ースとして機能する。レジスタファイルは、図 20 に示すように実装される。図 20 に示されているように、遅延および書き込みオーバーヘッドを低減するため、ユニット 136 は、機能ユニット 30～38 ごとに、別のレジスタファイルインスタンスに分解される。例示した実施形態において、各インスタンスは、48 個の 64 ビットレジスタを TPU のそれぞれについて提供する。他の実施形態は、TPU に割り当てられたレジスタの数に依存して、TPU の数およびレジスタのサイズが変動し得る。

【0190】

各インスタンス 200a～200e は、各インスタンスの上部に入る矢印によって示されるように、5 個の書き込みポートを有し、それを介して、機能ユニット 30～38 は同時に出力データを書き込むことができる（そのことにより、インスタンスが矛盾しないデータを維持することを確実にする）。それぞれは、各インスタンスの底部から出ている矢印によって示されているように、異なる数の読み出しポートを有し、それを介して、それぞれの機能ユニットがデータを取得する。従って、図に示されているように、整数ユニット 30 と、浮動小数点ユニット 32 と、メモリユニットとに関連付けられたインスタンスは全て、3 つの読み出しポートを有し、比較ユニット 34 に関連付けられたインスタンスは 2 つの読み出しポートを有し、ブランチユニット 38 に関連付けられたインスタンスは 1 つのポートを有する。

【0191】

レジスタファイルインスタンス 200～200e は、各サイクルで、1 つのスレッドにつき全てのポートが読み出されることによって最適化され得る。さらに、格納ビットは、ポートアクセスのためのワイヤの下でフォールド (fold) され得る。

【0192】

図 21 および 22 は、それぞれ、整数ユニット 30 および比較ユニット 34 のブロック図である。図 23A および 23B は、それぞれ、浮動小数点ユニット 32 および浮動小数点ユニット 32 で用いられている乗算-加算統合ユニットのブロック図である。これらのユニットの構成および動作は、図面に提供されているコンポーネント、インターコネクションおよびラベルから明らかである。

【0193】

(コンシューマープロデューサメモリ)

従来技術によるマルチプロセッサにおいて、同期化オーバーヘッドおよび（例えば、イメージ処理のための複数ステップについて）スレッドまたはプロセッサの間のデータベース処理フローを実装するためのプログラミングの困難さが非常に高い。プロセッサモジュール 5 は、これが容易に行われることを可能にするメモリ命令を提供し、データが利用可能になることをスレッドが待機することを可能にし、データが利用可能であることを他のスレッドが示すときに、透過的に起動させる。このようなソフトウェア透過性コンシューマープロデューサメモリ演算は、効率的なデータ志向コンシューマープロデューサプログラミングスタイルを有する、より高いパフォーマンスの微細なスレッドレベル並列処理を可能にする。

【0194】

例示した実施形態は、データを選択されたメモリ位置にロードし、状態をその位置、すなわち、「充填」状態に関連付けるように、データプロデューサであるスレッドによって用いられる「フル」メモリ命令を提供する。その命令が実行されるときに位置が既にその状態である場合、例外がシグナリングされる。

【0195】

この実施形態はまた、選択された位置からデータを取得するデータコンシューマによって用いられる「空」情報を提供する。位置がフル状態に関連付けられる場合、データはそこから（指定されたレジスタに）読み出され、命令は、位置が「空」状態と関連付けられるようにする。反対に、空状態が実行されるときに位置が完全状態には関連付けられない場合、命令は、命令を実行するスレッドを、一時的にアイドル（または、他の実施形態には、アクティブ、非実行）状態に移行させ、関連付けられた後で、アクティブ、実行状態

に戻して、空命令を完了まで実行する。空命令を用いることは、スレッドが、そのデータが、低いオーバーヘッドおよびソフトウェアに透過的な状態で、利用可能であるときに、実行することを可能にする。

【0196】

例示した実施形態において、ペンディング（メモリ）イベントテーブル（PET）50が、充填および空動作の対象であるメモリ位置についてのステータス情報を格納する。これには、それらの位置のアドレス、それぞれのフルまたは空状態、その位置のデータの「コンシューマ」のアイデンティティ、すなわち、空命令を実行して、位置が充填されることを待機しているスレッドが含まれる。これにはまた、データのプロデューサのアイデンティティが含まれる。これは、例えば、例外の原因をシグナリングし、追跡すること（例えば、同じアドレスについて、空命令が介在することなく後続の充填命令が実行される場所）において有用である。

10

【0197】

それぞれの位置についてのデータは、PET 50に格納されないが、充填および／または空命令の対象でないデータと同様に、キャッシュおよび／またはメモリシステム自体に維持される。他の実施形態において、ステータス情報は、メモリシステムにおいて、例えば、属する位置に沿って、かつ／または、別のテーブル、リンクされたリストなどに格納される。

【0198】

従って、例えば、空命令が所与のメモリ位置上で実行される場合、PETは、同じ位置が現在フル状態であることを示すエントリを有するか否かを決定するためにチェックされる。有する場合、エントリは、空に変更され、読み出しが行われ、データがメモリ位置から命令によって指定されたレジスタに移動される。

20

【0199】

反対に、空命令が実行されるときに、所与のメモリ位置について、PETにはエントリがない場合（または、このようなエントリのいずれかが、位置が現在からであると示す場合）、エントリが、PETにおいて、所与の位置が空であることを示し、空命令を実行するスレッドが充填命令によってその位置に後で格納される所与のデータのコンシューマであることを示すように、作成（または更新）される。

【0200】

充填命令が後で（おそらくは他のスレッドによって）実行される場合、PETは、同じ位置が現在空状態であることを示すエントリを有するか否かを決定するためにチェックされる。このようなエントリが見つかり、その状態はフルに変更され、イベント伝達メカニズム44（図4）は、通知をそのエントリにおいて識別されたコンシューマースレッドにルーティングするために用いられる。そのスレッドがTPUにおいてアクティブ、待機状態である場合、通知はそのTPUに送られ、TPUは、アクティブ、実行状態を入力し、空命令を、今回は（メモリ位置がフル状態なので）完了するために再実行する。そのスレッドがアイドル状態である場合、通知は（いずれのTPUで現在実行しているかに関わらず）システムスレッドに送られ、実行、アクティブ状態において、スレッドがTPUにロードされ、空命令が再実行されることできるようにする。

30

40

【0201】

例示した実施形態において、コンシューマ／プロデューサのようなメモリ演算のためのPETのこの使用は、選択されたメモリ命令、例えば、充填および空についてのみ行われ、より従来のロードおよび格納メモリ命令については行われない。従って、例えば、ロード命令が現在から命令の対象であるメモリ位置について、実行されている場合でも、命令が再実行されることができるよう、空命令を実行するスレッドに通知がされない。他の実施形態は、この点において異なり得る。

【0202】

図24Aは、3つの独立したスレッド230、232、および234を表す。スレッド230、232、および234の同期化、およびその間のデータ転送は、本発明による充

50

填および空命令によって容易にされ得る。例えば、MPEG2 デマルチプレクススレッド 230 は、例えば、チューナ、ストリーミングソース、その他のMPEG2 ソース236 から取得したMPEG2 信号をデマルチプレクスする役割を果たす。この例を続けるため、MPEG2 デマルチプレクススレッド230 は、TPU10 上でアクティブ、実行状態であると仮定される。スレッド232 は、映像デコードステップ1 スレッドであり、デマルチプレクスされたMPEG2 信号からの映像信号のデコードの第1の段階を行う。TPU12 上で、アクティブ、実行状態であると仮定される。スレッド234 は、映像復号ステップ2 スレッドであり、LCD インターフェース238 または他のデバイスを介する出力について、デマルチプレクスされたMPEG2 信号からの映像信号のデコードの第2の段階を行う。TPU14 上でアクティブ、実行状態であると仮定される。

10

【0203】

ソース236 からのデータストリーミングをリアルタイムで収容するため、スレッド230～234 のそれぞれは、アップストリームソースによって提供されるデータを継続的に処理し、他のスレッドと並列して行う。図24Bは、同期化を確実にし、スレッド間のデータ転送を容易にするような状態で、これを容易にする充填および空命令の使用を示す。

【0204】

図面を参照すると、矢印240a～240gは、スレッド間、特に、あるスレッドによって書き込まれた（充填された）データ位置と、他のスレッドによって読み出された（空にされた）データ位置との間の充填依存関係を示す。従って、スレッド230は、アドレスA0に向けられたデータを処理し、スレッド232は、その位置がターゲットの空命令を実行し、スレッド234は、アドレスB0（スレッド232が最終的に充填させる）がターゲットの空命令を実行する。空命令の結果として、スレッド232は、位置A0の充填が完了することを待機する間、待機状態（例えば、アクティブ、非実行、またはアイドル）になり、位置B0の充填の完了を待機する間、スレッド234は待機状態になる。

20

【0205】

スレッド230がA0の充填を完了すると、スレッド232の空にする動作も完了し、完全命令を介してB0に向けられた結果を用いて、そのスレッドがA0からのデータを処理することが可能になる。スレッド234は、待機状態を維持し、依然として、その充填の完了を待機する。その間、スレッド230は、アドレスA1に向けられた処理データを開始し、スレッド232は、空命令を実行し、A1の充填の完了を待機する間、待機状態にする。

30

【0206】

スレッド232がB0の充填デマンドを実行するとき、スレッド234を空にすることを完了することによって、TV視聴者へのディスプレイのためのLCD インターフェース（図示せず）によってそこから読み出される、C0に向けられた結果を用いて、そのスレッドがB0からのデータを処理することを可能になる。3つのスレッド230、232、234は、処理を継続し、充填および空命令を図に示したような状態で、MPEG2 ストリーム全体の処理が完了するまで、実行する。

【0207】

充填および空命令のさらなる理解は、その命令フォーマットを検討することによって達成され得る。

40

【0208】

（空）

フォーマット：ps EMPTY. cache. threads dreg, breg, ireg {, stop}

説明：空は、メモリシステムに有効アドレスの状態を調べるように命令する。状態がフルである場合、空命令は、状態を空にし、値をdregにロードするように変化させる。状態が既に空である場合、命令は、その命令がフルにされるまで、スレッドフィールドに指定された待機挙動で待機する。

50

【0209】

(オペランドおよびフィールド)

ps 命令が実行されるか否かを指定するプリディケートソースレジスタ。真の場合命令は実行され、偽の場合命令は実行されない（他の作用はなし）

stop 0 命令グループがこの命令によって詳述されていないことを示す

1 命令グループがこの命令によって詳述されていることを示す

thread 0 無条件、スレッドスイッチなし

1 無条件スレッドスイッチ

2 ストール（スレッドのブロック実行）時条件付きスレッドスイッチ

3 予約済み

10

scache 0 再使用キャッシュヒントがある **tbd**

1 再使用キャッシュヒントがある読み出し／書き込み

2 再使用キャッシュヒントがない **tbd**

3 再使用キャッシュヒントがない読み出し／書き込み

im 0 アドレス計算のためインデックスレジスタ (**ireg**) を指定

1 アドレス計算のため **disp** を指定

ireg 命令のインデックスレジスタを指定

breg 命令のベースレジスタを指定

disp メモリ参照命令のため2での補数置換定数（8ビット）を指定する

dreg 命令の宛先レジスタを指定する

20

(充填)

フォーマット: **ps FILL. cache. threads slreg, breg, ireg {, stop}**

説明: レジスタ **slreg** は、有効アドレスにおいて、メモリ内のワードに書き込まれる。有効アドレスは、**breg**（ベースレジスタ）と、**ireg**（インデックスレジスタ）または **im**（イミディエートメモリ）フィールドに基づいて **disp**（置換）のいずれかを加えることによって計算される。有効アドレスの状態はフルに変更される。状態が既にフルである場合、例外がシグナリングされる。

【0210】

(オペランドおよびフィールド)

30

ps 命令が実行されるか否かを指定するプリディケートソースレジスタ。真の場合命令は実行され、偽の場合命令は実行されない（他の作用はなし）

stop 0 命令グループがこの命令によって詳述されていないことを示す

1 命令グループがこの命令によって詳述されていることを示す

thread 0 無条件、スレッドスイッチなし

1 無条件スレッドスイッチ

2 ストール（スレッドのブロック実行）時条件付きスレッドスイッチ

3 予約済み

scache 0 再使用キャッシュヒントがある **tbd**

1 再使用キャッシュヒントがある読み出し／書き込み

2 再使用キャッシュヒントがない **tbd**

3 再使用キャッシュヒントがない読み出し／書き込み

40

im 0 アドレス計算のためインデックスレジスタ (**ireg**) を指定

1 アドレス計算のため **disp** を指定

ireg 命令のインデックスレジスタを指定

breg 命令のベースレジスタを指定

disp メモリ参照命令のため2での補数置換定数（8ビット）を指定する

slreg 命令の第1オペランドを含むレジスタを指定

(ソフトウェアイベント)

ハードウェアおよびソフトウェアイベントのより完全な理解が、命令フォーマットの検

50

討によって達成され得る。

【0211】

(イベント)

フォーマット: `ps EVENTs1 reg {, stop}`

説明: `EVENT` 命令は実行中スレッドについてイベントキューをポーリングする。イベントが存在する場合、命令は、例外ステータスレジスタにロードされるイベントステータスで完了する。そのイベントキューにイベントがない場合、スレッドはアイドル状態に移行する。

【0212】

(オペランドおよびフィールド)

`ps` 命令が実行されるか否かを指定するプリディケートソースレジスタ。真の場合命令は実行され、偽の場合命令は実行されない（他の作用はなし）

`stop 0` 命令グループがこの命令によって詳述されていないことを示す

1 命令グループがこの命令によって詳述されていることを示す

`s1 reg` 命令の第1ソースオペランドを含むレジスタを指定

(SWイベント)

フォーマット: `ps. SWEVENTs1 reg {, stop}`

説明: `SWEvent` 命令は、イベントをスレッドによって扱われるイベントキューにエンキューする。イベントフォーマットについて `xxx` を参照されたい。

オペランドおよびフィールド

`ps` 命令が実行されるか否かを指定するプリディケートソースレジスタ。真の場合命令は実行され、偽の場合命令は実行されない（他の作用はなし）

`stop 0` 命令グループがこの命令によって詳述されていないことを示す

1 命令グループがこの命令によって詳述されていることを示す

`s1 reg` 命令の第1ソースオペランドを含むレジスタを指定

(CtlFld)

フォーマット: `ps CtlFld. ti cfield, {, stop}`

説明: コントロールフィールド命令は、`cfield` で指定されたコントロールフィールドを変更する。コントロールレジスタ内の他のフィールドは、変更されない

(オペランドおよびフィールド)

`ps` 命令が実行されるか否かを指定するプリディケートソースレジスタ。真の場合命令は実行され、偽の場合命令は実行されない（他の作用はなし）

`stop 0` 命令グループがこの命令によって詳述されていないことを示す

1 命令グループがこの命令によって詳述されていることを示す

`ti 0` このスレッドの制御レジスタへのアクセスを指定

1 `ID. t_indirect field` (スレッド不正) (特権) によって指定されたスレッドの制御レジスタへのアクセスを指定

`cfield`

`cfield [4:0]` コントロールフィールド 特権

【0213】

10

20

30

40

【表 3 8】

000	スレッド状態	アプリケーション
nn	値	
00	アイドル	
01	予約済み	
10	待機	
11	実行中	
0010S	システムトラップイネーブル	システム
0011S	アプリケーショントラップイネーブル	アプリケーション
0100S	スレッドイネーブル	システム
0101S	特権レベル	システム
0110S	変更されたレジスタ	アプリケーション
0111S	命令アドレス変換イネーブル	システム
1000S	データアドレス変換イネーブル	システム
1001S	アライメントチェック	システム
1010S	エンディアンモード	システム
1011S	予約済み	
11**S		

S = 0 クリア、S = 1 設定

(プロセッサモジュール5を組み込んだデバイス)

図25は、SoCフォーマットで具現化された本発明による、デジタルLCD-TVサブシステム242のブロック図である。サブシステム242は、上述したようにプロセッサモジュール5を含み、例えば、上述した、MPEG2信号デマルチプレクス、MPEG2映像デコード、MPEG音声デコード、デジタルTVユーザインターフェース操作、およびオペレーティングシステム実行（例えば、Linux）を提供する実行スレッドを同時に実行するように動作する。モジュール5は、これも上述したように、L2キャッシュ26のオフチップ部分を含むDDR DRAMフラッシュメモリに接続されている。このモジュールは、AMBA AHBバス244へのインターフェース（図示せず）を含む。このモジュールは、AMBA AHBバス244を介して、デジタルLCD-TVの他のコンポーネントへのインターフェース、すなわち、映像入力インターフェース、映像出力インターフェース、音声出力インターフェース、およびLCDインターフェースを提供する「知的財産」または「IP」246と通信する。当然、他のIPが、追加で、またはこれらの代わりに設けられ、AHBバス5を介して、またはその他の方法でモジュール5に接続されてもよい。例えば、図面において、例示したモジュール5は、オプションのIP、例えば、DMAエンジン248、高速I/Oデバイスコントローラ250、および低速デバイスコントローラ252（APBブリッジ254を介して）と通信し、それを介して、デジタルLCD-TVはソース信号を取得し、かつ／または、制御される。

【0214】

図26は、同様にSoCフォーマットで具現化された本発明による、デジタルLCD-TVまたは他のアプリケーションサブシステム256のブロック図である。例示したサブシステムは、図25に246で示される特定のIPの代わりに、APBおよびAHB/APBブリッジと、APBマクロ258とを有する状態で描かれている点以外は、上述したように構成されている。アプリケーションの必要性に依存して、素子258は、上記の実装と同様に、または他の様態で、映像入力インターフェース、映像出力インターフェース、音声出力インターフェースおよびLCDインターフェースを含み得る。

【0215】

例示したサブシステムは、複数のモジュール5、例えば、1～20個（またはこれ以上）のこのようなモジュールをさらに含む。これらのモジュールは、モジュール5によって利用されるオフチップL2キャッシュ26bとインターフェースし、好ましくは、その一部を形成するインターコネクタを介して接続される。このインターコネクタは、モジュール5によって共有され、より具体的には、L2キャッシュ26によって共有されるシフトレジスタバスを含むリングインターコネクタ（RI）の形態であり得る。あるいは、インターコネクタは、他の形態の、モジュール5の組み合わせられたメモリシステム内でのデータの高速移動を容易にする、独占権下にあってもよいし、そうでなくてもよいインター

コネクトであり得る。いずれにせよ、L2 キャッシュは、任意の1つのモジュール5のためのL2 キャッシュがその個々のプロセッサのための唯一のメモリシステムになるのではなく、プロセッサモジュール5の全てについての分散型キャッシュメモリシステム全体にも寄与するように、接続されることが好ましい。当然、上述したように、モジュール5は、同じメモリシステム、チップまたはバスを物理的に共有する必要はなく、ネットワークを介して、または他の状態で接続され得る。

【0216】

以上のように、本発明の好ましい実施形態を用いて本発明を例示してきたが、本発明は、この実施形態に限定して解釈されるべきものではない。本発明は、特許請求の範囲によってのみその範囲が解釈されるべきであることが理解される。当業者は、本発明の具体的な好ましい実施形態の記載から、本発明の記載および技術常識に基づいて等価な範囲を実施することができることが理解される。本明細書において引用した特許、特許出願および文献は、その内容自体が具体的に本明細書に記載されているのと同様にその内容が本明細書に対する参考として援用されるべきであることが理解される。

【0217】

(要約)

本発明は、ある局面において、1つ以上の仮想処理装置を含む仮想プロセッサを提供する。これらの仮想処理装置は、1つ以上のプロセッサ上で実行され、それぞれの仮想処理装置は、1つ以上のプロセスまたはスレッド（まとめて、「スレッド」という）を実行する。スレッドは、同じ仮想処理装置上で、それぞれの寿命を通じて実行されるように制約されてもよいが、必ずしもそうである必要はない。イベント伝達メカニズムは、イベントをそれぞれのスレッドに関連付け、そのスレッドがそのときに実行されている仮想処理装置および／またはプロセッサに関わらず、イベントが発生するときにこれらのスレッドに通知する。本発明は、他の局面において、改善されたデータフローベースの同期化を有する仮想および／またはデジタルデータプロセッサを提供する。このようなプロセッサ内で実行されているプロセスまたはスレッド（再度、まとめて「スレッド」という）は、例えば、他のスレッドによって生成されるデータの利用可能性をスレッドが待機し、他のスレッドがデータを（例えば、「充填」または他のメモリーコンシューマ命令を実行することによって）利用可能にするとときに透過的に起動させることを可能にする、メモリー命令（例えば、「空」または他のメモリーコンシューマ命令）を実行し得る。

【図面の簡単な説明】

【0218】

【図1】図1は、本発明のある実施例に従って構成され、動作するプロセッサモジュールを表す図である。

【図2】図2は、従来のスーパースカラープロセッサによる処理と本発明のある実施例に従って構成され、動作するプロセッサモジュールによる処理とを対比させた図である。

【図3】図3は、本発明のある実施例に従って構成され、動作するプロセッサ内の仮想処理装置（またはスレッド処理装置（TPU））において実行されているスレッドの可能な状態を示す図である。

【図4】図4は、本発明のある実施例に従って構成され、動作するプロセッサモジュールにおけるイベント伝達メカニズムを表す図である。

【図5】図5は、本発明のある実施例に従って構成され、動作するシステムにおける仮想アドレスからシステムアドレスへの変換のメカニズムを示す図である。

【図6】図6は、本発明のある実施例に従って構成され、動作するシステムにおけるレベル1およびレベル2キャッシュの構成を表す図である。

【図7】図7は、本発明のある実施例に従って構成され、動作するシステムにおいてタグルックアップを行うために用いられるL2キャッシュおよびロジックを表す図である。

【図8】図8は、本発明のある実施例に従って構成され、動作するシステムにおいて、L2拡張キャッシュ内でタグルックアップを行うために用いられるロジックを表す図である。

【図 9】図 9 は、本発明のある実施例に従って構成され、動作するシステムにおいて、各スレッド処理装置（TPU）について維持される、汎用レジスタ、プリディケートレジスタ、スレッド状態レジスタまたは制御レジスタを表す図である。

【図 10】図 10 は、本発明のある実施例に従って構成され、動作するシステムにおいて、スレッドによって実行される命令をフェッチし、ディスパッチするメカニズムを表す図である。

【図 11】図 11 は、本発明のある実施例に従って構成され、動作するシステムにおいて用いられるキュー管理メカニズムを示す図である。

【図 12】図 12 は、本発明のある実施例に従って構成され、動作するシステムにおいて用いられるキュー管理メカニズムを示す図である。

10

【図 13】図 13 は、本発明のある実施例に従って構成され、動作する、処理ユニットを実装するロジックを含む、図 1 のプロセッサモジュールのシステム・オン・チップ（SOC）実装を表す図である。

【図 14】図 14 は、本発明のある実施例に従って構成され、動作するシステムにおけるパイプライン制御ユニットのブロック図である。

【図 15】図 15 は、本発明のある実施例に従って構成され、動作するシステムにおける個々のユニットキューのブロック図である。

【図 16】図 16 は、本発明のある実施例に従って構成され、動作するシステムにおけるブランチユニットのブロック図である。

【図 17】図 17 は、本発明のある実施例に従って構成され、動作するシステムにおけるメモリユニットのブロック図である。

20

【図 18】図 18 は、本発明のある実施例に従って構成され、動作するシステム L1 命令キャッシュまたは L1 データキャッシュのいずれかを実装するキャッシュユニットのブロック図である。

【図 19】図 19 は、本発明のある実施例に従って構成され、動作するシステムにおける、図 7 の L2 キャッシュおよびロジックの実装を表す図である。

【図 20】図 20 は、本発明のある実施例に従って構成され、動作するシステムにおけるレジスタファイルの実装を表す図である。

【図 21】図 21 は、本発明のある実施例に従って構成され、動作するシステムにおける整数ユニットおよび比較ユニットのブロック図である。

30

【図 22】図 22 は、本発明のある実施例に従って構成され、動作するシステムにおける整数ユニットおよび比較ユニットのブロック図である。

【図 23 A】図 23 A は、本発明のある実施例に従って構成され、動作するシステムにおける浮動小数点ユニットのブロック図である。

【図 23 B】図 23 B は、本発明のある実施例に従って構成され、動作するシステムにおける浮動小数点ユニットのブロック図である。

【図 24 A】図 24 A は、本発明のある実施例に従って構成され、動作するシステムにおけるコンシューマおよびプロデューサメモリ命令の使用を示す図である。

【図 24 B】図 24 B は、本発明のある実施例に従って構成され、動作するシステムにおけるコンシューマおよびプロデューサメモリ命令の使用を示す図である。

40

【図 25】図 25 は、本発明のある実施例に従って構成され、動作するシステムにおけるデジタル LCD-TV サブシステムのブロック図である。

【図 26】図 26 は、本発明のある実施例に従って構成され、動作するシステムにおけるデジタル LCD-TV サブシステムまたは他のアプリケーションサブシステムのブロック図である。

【符号の説明】

【0219】

5 モジュール

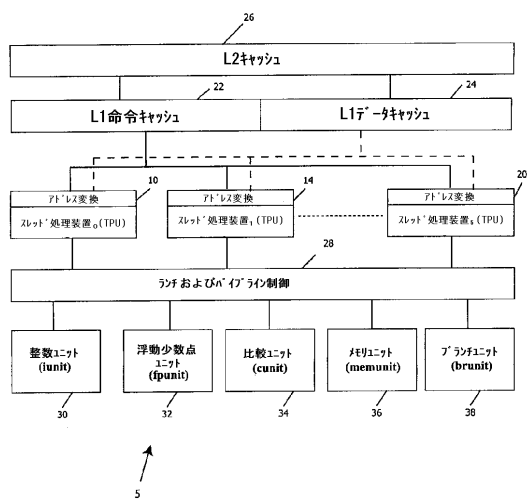
10～20 スレッド処理装置（TPU）

22 L1 命令キャッシュ

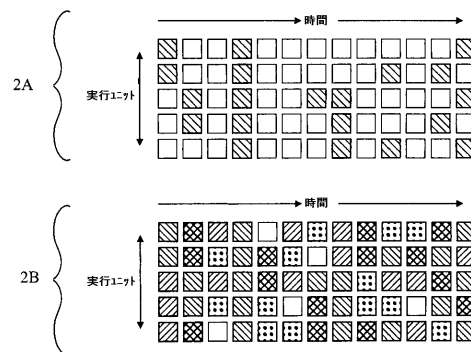
50

- 24 L1データキャッシュ
- 26 L2キャッシュ
- 28 パイプライン制御
- 30 整数ユニット
- 32 浮動小数点ユニット
- 34 比較ユニット
- 36 メモリユニット
- 38 ブランチユニット

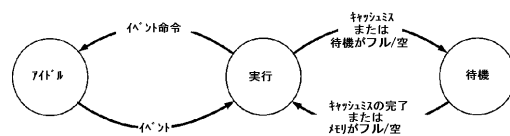
【図1】



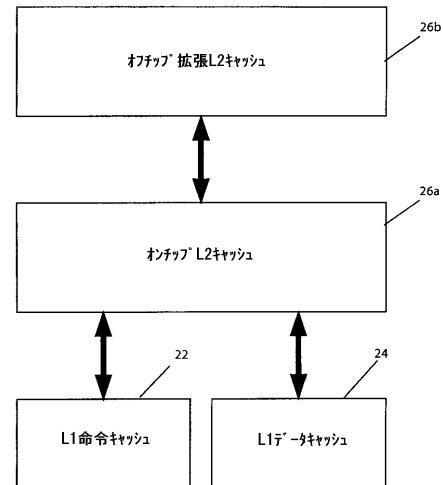
【図2】



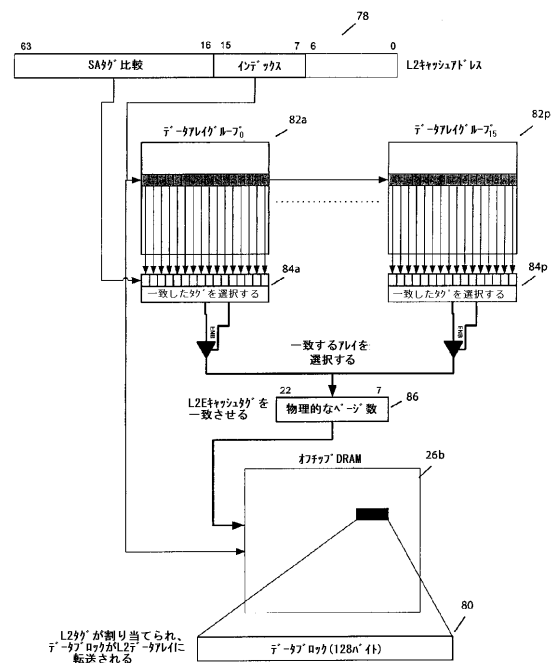
【図3】



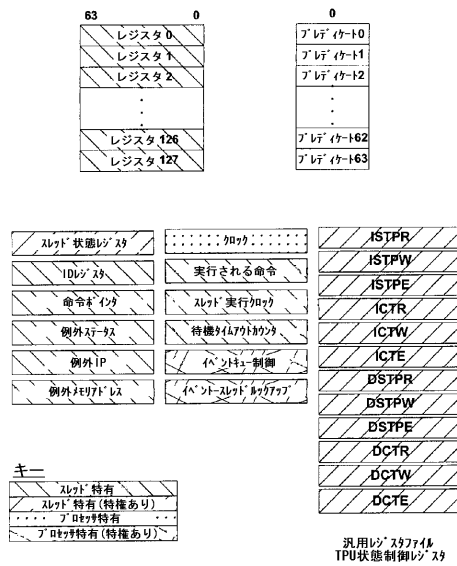
【図 6】



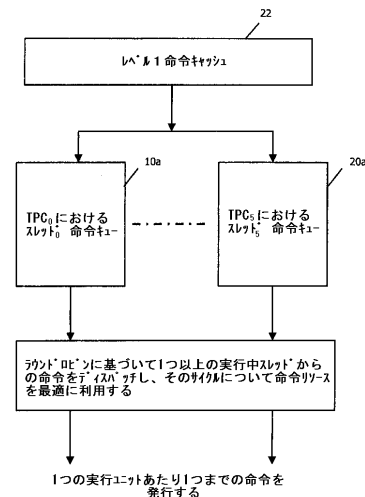
【图 8】



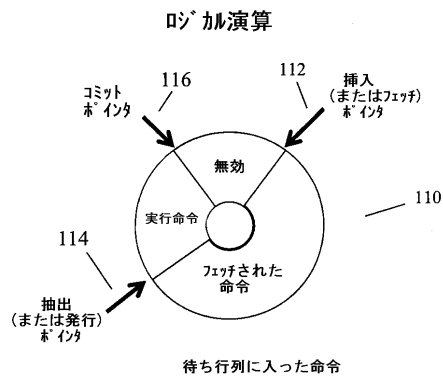
【図 9】



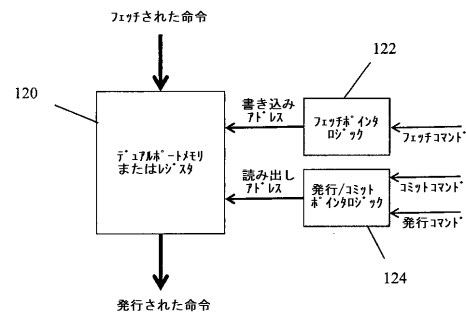
【図 10】



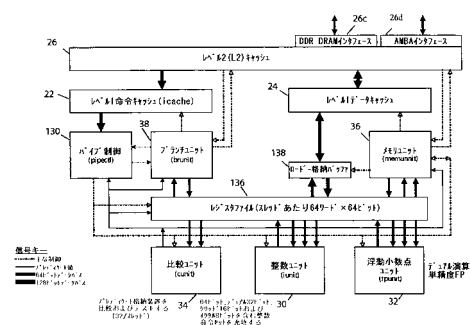
【図 1 1】



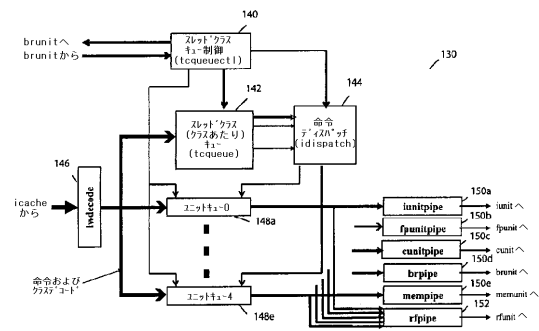
【图 1 2】



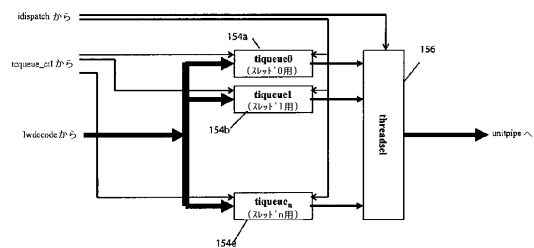
【图 1 3】



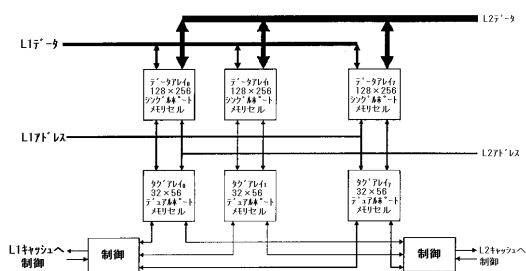
【図 14】



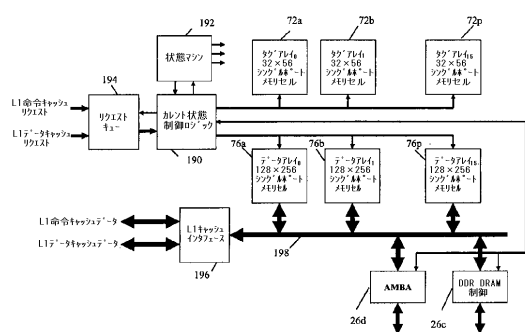
【図 15】



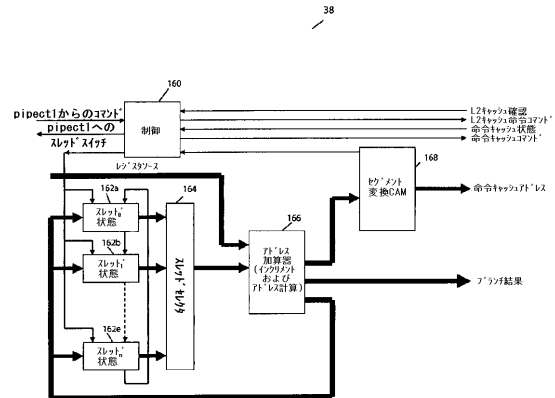
【図 18】



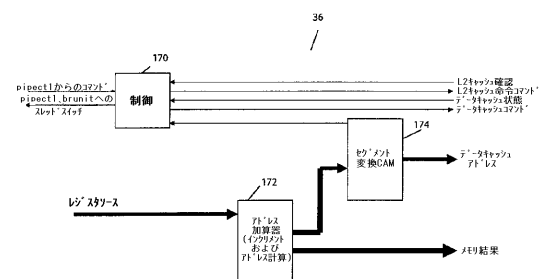
【図 19】



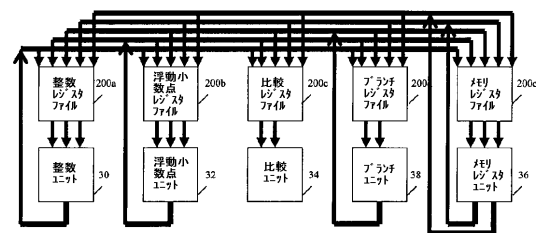
【図 16】



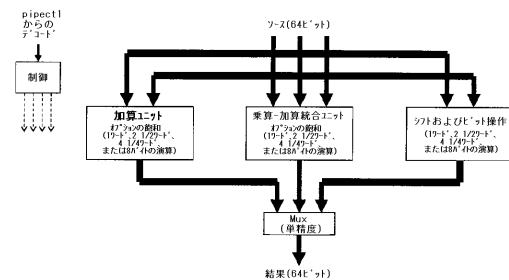
【图 17】



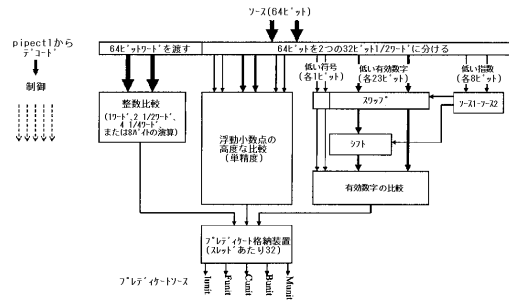
【図 20】



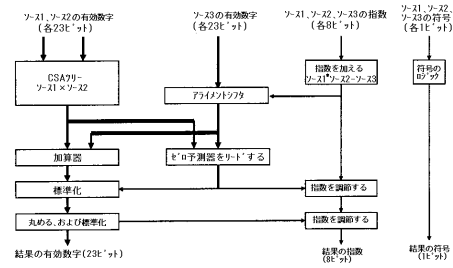
【図 2 1】



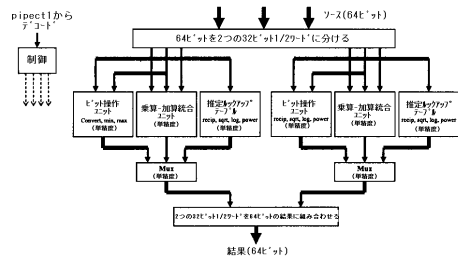
【図 2 2】



【図 2 3 B】



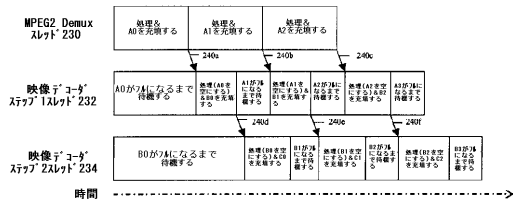
【図 2 3 A】



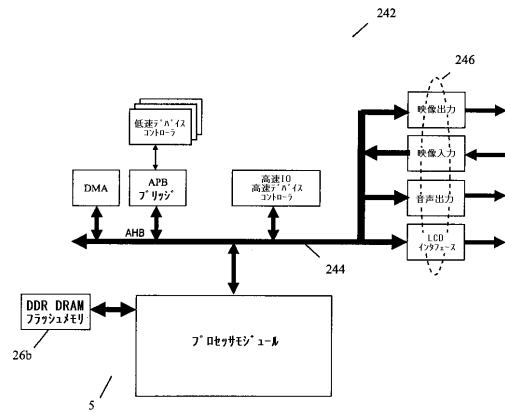
【図 2 4 A】



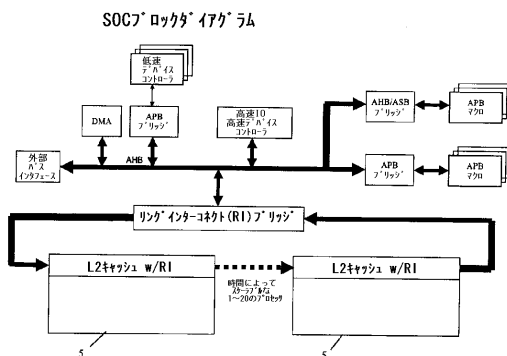
【図 2 4 B】



【図 2 5】



【図 2 6】



フロントページの続き

- (72)発明者 フランク スティーブン
アメリカ合衆国 マサチューセッツ 01027, イーストハンプトン, プレザント ストリート 116, アpartment ナンバー1
- (72)発明者 今井 繁規
奈良県奈良市丸山一丁目1079-147
- (72)発明者 米田 照正
奈良県磯城郡田原本町薬王寺200-18

審査官 井上 宏一

- (56)参考文献 特表2002-544620 (JP, A)
特開2000-076087 (JP, A)
特開平03-208131 (JP, A)
特開2001-051860 (JP, A)
特開平03-040035 (JP, A)
特表2003-516570 (JP, A)

- (58)調査した分野(Int.Cl., DB名)
G06F 9/46 - 9/54