(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3805305号
(P3805305)
(45)発行日　平成18年8月2日(2006.8.2)　(24)登録日　平成18年5月19日(2006.5.19)

(51)Int.Cl.		F I		
G06F 9/30	(2006.01)	G06F	9/30	330A
G06F 9/38	(2006.01)	G06F	9/38	310E
G06F 9/46	(2006.01)	G06F	9/46	410
G06F 9/48	(2006.01)	G06F	9/46	452B

請求項の数 30　(全 20 頁)

(21)出願番号	特願2002-566777 (P2002-566777)
(86)(22)出願日	平成14年2月19日 (2002.2.19)
(65)公表番号	特表2004-532444 (P2004-532444A)
(43)公表日	平成16年10月21日 (2004.10.21)
(86)国際出願番号	PCT/GB2002/000742
(87)国際公開番号	WO2002/067116
(87)国際公開日	平成14年8月29日 (2002.8.29)
審査請求日	平成16年9月30日 (2004.9.30)
(31)優先権主張番号	0104045.0
(32)優先日	平成13年2月19日 (2001.2.19)
(33)優先権主張国	英国(GB)

(73)特許権者　501176037
イマジネイション　テクノロジーズ　リミテッド
イギリス　ハートフォードシャー　ダブリューディー4　8エルゼット　キングス　ラングリー　ホーム　パーク　エステイト　(番地なし)
(74)代理人　100059959
弁理士　中村　稔
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100065189
弁理士　宍戸　嘉一

最終頁に続く

(54)【発明の名称】マルチスレッドプロセッサ上の優先順位及び命令速度の制御

(57)【特許請求の範囲】

【請求項１】

プロセッサによって実行される命令スレッドに対して命令の発行速度を制御する方法であって、

命令スレッドに対して命令が実行される速度を記憶する段階と、

前記記憶された速度に応答して命令を実行させるために要求を発行する段階と、

命令実行に応答して命令要求が発行される速度を低減する段階と、

命令実行がない場合に命令が実行される速度を増加させる段階と、

を含むことを特徴とする方法。

【請求項２】

命令の意図した実行速度と実際の実行速度との差の値を累積する段階を含み、

前記命令を実行させるために要求を発行する段階は、実行の前記実際の速度と前記意図した速度との間の不足量に依存する、

ことを特徴とする請求項１に記載の方法。

【請求項３】

累積過剰（accumulated excess）に応答して命令の実行を停止する段階を含むことを特徴とする請求項２に記載の方法。

【請求項４】

前記プロセッサは、複数の命令スレッドを処理するマルチスレッドプロセッサであることを特徴とする請求項１から請求項３のいずれか１項に記載の方法。

【請求項５】
各スレッドに優先順位を割り当て、各スレッドの優先順位によって各スレッド上の命令を実行する段階を含むことを特徴とする請求項４に記載の方法。
【請求項６】
前記スレッドに優先順位を割り当てる段階は、
各スレッドに対するランク順位を確立するのに必要な複数のメトリックを準備する段階と、
各メトリックを一組のビットに割り当てる段階と、
前記一組のビットを各スレッドに対する複合メトリックに組み立て、重要度が最も高いメトリックが前記複合メトリックの最上位のビットに割り当てられ、重要度が最も低いメトリックが最下位のビットに割り当てられるようにする段階と、
を含むことを特徴とする請求項５に記載の方法。
【請求項７】
前記スレッドの実行に対するリアルタイムの締切をモニタし、前記締切が過ぎるまでに残された時間によって前記スレッドの優先順位を調節する段階を含むことを特徴とする請求項５に記載の方法。
【請求項８】
プロセッサによって実行される命令スレッドに対して命令の発行速度を制御するための装置であって、
命令スレッドに対して命令が実行される速度を記憶する手段と、
前記記憶された速度に応答して命令を実行させるために要求を発行する手段と、
実行に応答して要求が発行される速度を低減する手段と、
命令実行がない場合に命令が実行される速度を増加させる手段と、
を含むことを特徴とする装置。
【請求項９】
意図した実行速度と実際の実行速度との間の値の差を累積する手段、
を含み、
前記命令を実行させるために要求を発行する手段は、前記実際及び意図した実行速度間の不足量に依存する、
ことを特徴とする請求項８に記載の装置。
【請求項１０】
前記実際及び意図した実行速度間の累積過剰に応答して実行を停止する手段を含むことを特徴とする請求項８に記載の装置。
【請求項１１】
前記プロセッサは、複数の命令スレッドを処理するマルチスレッドプロセッサであることを特徴とする請求項８、請求項９、又は、請求項１０に記載の装置。
【請求項１２】
各スレッドに優先順位を割り当てる手段を含み、
前記要求を発行する手段は、各スレッドの優先順位によって発行する、
ことを特徴とする請求項１１に記載の装置。
【請求項１３】
前記スレッドに優先順位を割り当てる手段は、
各スレッドに対するランク順位を確立するのに必要な複数のメトリックを準備する手段と、
各メトリックを一組のビットに割り当てる手段と、
前記一組のビットを各スレッドに対する複合メトリックに組み立て、重要度の最も高いメトリックが前記複合メトリックの最上位のビットに割り当てられ、重要度の最も低いメトリックが最下位のビットに割り当てられるようにする手段と、
を含むことを特徴とする請求項１２に記載の装置。
【請求項１４】

前記スレッドの実行に対するリアルタイムの締切をモニタする手段と、
前記締切が過ぎるまでに残された時間によって前記スレッドの優先順位を調節する手段と、
を含むことを特徴とする請求項１２に記載の装置。
【請求項１５】
マルチスレッドプロセッサ上で実行される複数の命令スレッドにランク順位を割り当てる方法であって、
各スレッドに対するランク順位を確立するのに必要な複数のメトリックを準備する段階と、
各メトリックを一組のビットに割り当てる段階と、
前記一組のビットを複合メトリックに組み立て、重要度の最も高いメトリックが前記複合メトリックの最上位のビットに割り当てられ、重要度の最も低いメトリックが前記複合メトリックの最下位のビットに割り当てられるようにする段階と、
前記複合メトリックに対し、それらの値によってランク順位を割り当てる段階と、
を含むことを特徴とする方法。
【請求項１６】
前記複合メトリックに対してランク順位を割り当てる段階は、
一対の複合メトリックを比較する段階と、
下位に位置付けされた複合メトリックが上位に位置付けされた複合メトリックよりも大きいか又は等しい値を有する場合に、それらの順位を交換する段階と、
を含むことを特徴とする請求項１５に記載の方法。
【請求項１７】
前記一対の複合メトリックを比較する段階は、最下位に順位付けられた複合メトリックの対から最上位に順位付けられた対に向けて進められることを特徴とする請求項１６に記載の方法。
【請求項１８】
前記最上位に順位付けられた対の比較の後に、処理が前記最下位に順位付けられた対に戻ることを特徴とする請求項１７に記載の方法。
【請求項１９】
マルチスレッドプロセッサ上で実行される複数の命令スレッドにランク順位を割り当てるための装置であって、
スレッドに対するランク順位を確立するのに必要な複数のメトリックを準備する手段と、
各メトリックを一組のビットに割り当てる手段と、
前記一組のビットを複合メトリックに組み立て、重要度の最も高いメトリックが最上位のビットに割り当てられ、重要度の最も低いメトリックが最下位のビットに割り当てられるようにする手段と、
前記複合メトリックに対し、それらの値によってランク順位を割り当てる手段と、
を含むことを特徴とする装置。
【請求項２０】
前記複合メトリックにランク順位を割り当てる手段は、
一対の複合メトリックを比較する手段と、
下位に位置付けされた複合メトリックが上位に位置付けされた複合メトリックよりも大きいか又は等しい値を有する場合に、前記一対の複合メトリックのランク順位を交換する手段と、
を含むことを特徴とする請求項１９に記載の装置。
【請求項２１】
前記複合メトリックを比較する手段は、最下位に順位付けられた対から最上位に順位付けられた対まで進められることを特徴とする請求項２０に記載の装置。
【請求項２２】

前記比較手段は、前記最上位に順位付けられた対の比較の後に、前記最下位に順位付けられた対に戻ることを特徴とする請求項２１に記載の装置。
【請求項２３】
マルチスレッドプロセッサ上で実行されるスレッドの命令発行速度を制御する方法であって、
各スレッドが命令を実行するべき平均速度を記憶する段階と、
スレッドに対して未発行の命令の数を表す値をモニタする段階と、
各スレッドに対する前記記憶された平均速度によって前記値を増加させる段階と、
命令実行に応答して前記値を減少させる段階と、
を含むことを特徴とする方法。
【請求項２４】
スレッドに対して未発行の命令の数を表す前記値を累積する段階を含むことを特徴とする請求項２３に記載の方法。
【請求項２５】
スレッドに対して累積することができる命令の数に最大値を設定する段階を含むことを特徴とする請求項２４に記載の方法。
【請求項２６】
未発行の命令を累積して蓄積するスレッドには、前記命令がより高速で実行される、前記累積値によって制限されたバースト実行時間を割り当てることができることを特徴とする請求項２４又は請求項２５に記載の方法。
【請求項２７】
マルチスレッドプロセッサ上で実行されるスレッドの命令発行速度を制御するための装置であって、
各スレッドが命令を実行するべき平均速度を記憶する手段と、
スレッドに対して利用可能であるが未発行の命令の数を表す値をモニタする手段と、
各スレッドに対して前記記憶された平均速度によって前記値を増加させる手段と、
スレッド上の命令実行に応答して前記値を減少させる手段と、
利用可能であるが未発行の命令の数が規定値よりも低くなった場合に、スレッド上の命令の実行を防止する手段と、
を含むことを特徴とする装置。
【請求項２８】
スレッドに対して利用可能であるが未発行の命令の数を表す前記値の中に、平均命令発行速度を累積する手段を含むことを特徴とする請求項２７に記載の装置。
【請求項２９】
スレッドに対して利用可能であるが未発行の命令の数に最大値を設定する手段を含むことを特徴とする請求項２８に記載の装置。
【請求項３０】
未発行の命令を累積して蓄積するスレッドには、命令が前記平均速度よりも高速で実行される、前記累積値によって制限されたバースト実行時間を割り当てることができることを特徴とする請求項２８及び請求項２９に記載の装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、何時でもいくつかの異なる命令スレッドの１つを処理するように構成されたマルチスレッドプロセッサ上の優先順位及び命令発行速度の制御に関する。
【背景技術】
【０００２】
本発明は、本明細書においてその内容が引用により組み込まれる、本出願人の国際特許出願番号ＷＯ９７／３８３７２に説明されたようなシステムと一緒に使用される場合に特に有益である。この文献は、１つ又はそれ以上のデータプロセッサによっていくつかの異

なる命令スレッドを処理することができる処理システムを開示する。このアーキテクチャは、次のクロックサイクルでどの命令スレッドが処理されるべきかを判断するために、使用可能なリソースと実行されるべき命令とを何度も見るのである。そのようなアーキテクチャには、プロセッサがハードウエア周辺装置又は他のプロセッサのような外部装置と対話するリアルタイムシステムにおいて多くの利点がある。リアルタイムシステムでは、イベントに付随する全ての処理を所定時間内に確実に完了させることが極めて重要である。これは、１つのタスクだけを処理するプロセッサに対しては簡単で確認されるが、プロセッサに実行するタスクが多数ある場合には非常に複雑になる。マルチスレッドを有するプロセッサのシステムでは、１つのスレッド上で実行しているプログラムの作動が、別のスレッド上で実行しているプログラムの挙動を変えることによって乱されることが十分に起こりうる。このスレッドの整合性の欠如のために、別のスレッドで何を実行しているかを予め知らないで確実に実行できるプログラムを開発することが困難になる。

【０００３】

【特許文献１】国際特許出願番号ＷＯ９７／３８３７２

【発明の開示】

【発明が解決しようとする課題】

【０００４】

従来のプロセッサは、緊急のタスクを通常又は緊急でないタスクよりも速く処理できる優先順位システムを使用する。しかし、ＷＯ９７／３８３７２に説明されたような多重ハードウエアスレッドもまた有するプロセッサは、従来のプロセッサよりも柔軟性があり、従って、より柔軟性のある優先順位制御が必要となる。

【課題を解決するための手段】

【０００５】

本発明の好ましい実施形態は、マルチスレッドプロセッサシステムのための発行速度制御スキームを提供することを目的とするものである。特に、好ましい実施形態は、１つのスレッド上で実行されているプログラムに、そのプログラムと別のスレッド上で実行されている任意のプログラムとの両方の処理要件が満足される方法で、そのプロセッサリソースの使用を制御させることを目的とするものである。これを行うために、プログラムは、（１）別のスレッド上で実行されているプログラムの挙動に関係なく、その命令が発行される速度を規定すること、（２）別のスレッド上で実行されているプログラムに対するあらゆる混乱を制御しながら、緊急のイベントを素早く処理すること、及び、（３）別のスレッド上の緊急イベントの処理によって引き起こされた、命令発行の規定速度の混乱に適応することが可能であるべきである。

【０００６】

この好ましい実施形態は、スレッドがプロセッサの過負荷から確実な方法で復旧することができ、スレッド上の実行速度の所定の限界からの偏差を確実に検出することができるほど十分に強力である。

更に、この実施形態は、いくつかの複雑性レベルで作動可能である一方で、プログラマーが要求しないスキームの側面をプログラマーが無視することを可能にする制御スキームを提供することを目的としている。

【０００７】

更に別の実施形態では、全てのタスクを完了するのに必要な最小速度でプロセッサを刻時することにより、プロセッサの電力消費を最小化することを目的としている。

好ましい実施形態は、順位付けされた優先順位を命令スレッドに割り当て、プロセッサリソースの最も有効な使用を保証することを目的としている。

本発明は、ここで参照されるべきである特許請求の範囲で規定される。

ここで、本発明の好ましい実施形態は、添付図面を参照し一例として以下に詳細に説明される。

【発明を実施するための最良の形態】

【０００８】

本明細書に説明される本発明の実施形態の基本アーキテクチャが図１に示されている。この中心部分は、密なマルチスレッドプロセッサで構成された媒体制御コア（ＭＣＣ）２である。それは、例えば、ビデオソース、オーディオソース、ビデオ出力、オーディオ出力、データソース、及び、ストレージなどとすることができるリアルタイムデータ入出力装置４に結合することができる複数の入出力を有する。最も単純な例では、単一入力と単一出力だけが設けられることになる。

「ＭＣＣ」２には、複数のデータ処理ユニット６も結合される。これらのユニットは、いずれもデータパイプライン１０を通じてデータ処理を制御するデータ処理コア８を有する。コア８は、パイプライン１０に対するマイクロ命令を復号して順序付ける。

【０００９】

媒体制御コア２に結合されるのは、「ＭＣＣ」２及びデータ処理ユニット６によってデータを検索することができ、「ＭＣＣ」２及びデータ処理ユニット６によってデータを書き込むことができるマルチバンク・キャッシュメモリ１２である。それは、データと、入力データ及び他の内部生成データに対してデータ処理コアによって実行される命令とのための一時的ストレージを含む。これらの種々の命令の組は、活動化された時に、処理スレッドを構成することになる。

【００１０】

「ＭＣＣ」２は、データを入力４からデータ処理コア６又はキャッシュ１２のストレージに誘導し、データを出力に供給する密なマルチスレッド処理ユニットである。仮に要求があれば、それは、クロックサイクル毎にタスクを切り替えることができるように構成される。これを達成するために、それは、実行されるべきタスクと実行されるタスクに利用可能なリソースを見て、それがどのオペレーションを実行できるかをクロックサイクル毎に検査する。それはまた、これらのタスクのどれが最も優先度が高いかを検査する。処理能力が十分であれば、それぞれのクロックサイクル毎に、１つよりも多いオペレーションを開始することができる。

【００１１】

このリソースの検査は、入力ポートに対するデータのような外部リソース、又は、データストレージ又は出力装置の利用可能性を含む、実行されるべき特定のタスクのために必要なことの全てが、命令が出される前に確実に所定の位置にあることを確実にする。それはまた、一時的な記憶のためのレジスタ、処理コア、又は、特定の新しい処理オペレーションのために必要な以前に処理されたデータなどの内部リソースの検査を含む。「ＭＣＣ」２は、データを入力装置から適切なデータ処理ユニット６へ誘導し、処理を行うために適切な命令をユニット６に経路指定し、処理されたデータを必要な時に出力に経路指定して必要に応じてキャッシュを使用するように作動する。命令の組の実行が処理ユニット上で開始された状態で、「ＭＣＣ」２は、データ処理ユニット上でプログラムの実行が続けられる一方で、実行することができる様々なスレッドとこれらのために使用できるリソースを再度見ることができる。

【００１２】

このリソース及び優先度の検査は、ビデオ入力のようなリアルタイムデータに対して実行するタスクが、リアルタイム入力に通常必要な大きなメモリバッファがなくても実行することができることを意味する。例えばビデオ入力においては、「ＭＣＣ」は、入出力ポートでデータが利用可能かどうかを判断するために見て、もしそうであれば、そのデータを受け取り、データ処理ユニット６の１つによる処理のために、それをマルチバンクキャッシュの一部分又はデータ記憶レジスタのいずれかに送ることになる。

【００１３】

データ処理ユニット６のスケジューリングは、「ＭＣＣ」２によって制御される。例えば、図１のデータパイプライン１０は、データ処理アルゴリズムを実行するためにキャッシュから取り出された一連の命令を実行する関連データ処理コア８の制御のもとで、乗算器、加算器、及び、シフタなどのいくつかの処理要素によって構成されることになる。これらの処理コアは、それぞれ、特定のデータ処理を実行するために一連の命令を記憶する

、独自のマイクロ命令ＲＯＭ及びＲＡＭを有する。「ＭＣＣ」２は、その特定のオペレーションシーケンスを実行するために、例えばそのマイクロ命令ＲＯＭの中にアドレスオフセットを通し、それに実行を開始するように命令することにより、データ処理ユニット６を呼び出す。次に、それは、マルチバンクキャッシュからのデータ又は入力装置の１つから「ＭＣＣ」２に通されたデータの何れかに対して特定の処理を実行し、それが完了した時点で、「ＭＣＣ」２に対してその処理が完了したことを信号で知らせることになる。

【００１４】

本発明のこの実施形態では、「ＭＣＣ」２によって実行されるスレッドスケジューリングには２つの主要な要素がある。２つの要素とは、スレッド命令の発行速度制御とスレッド優先順位とである。命令発行速度制御により、それぞれのスレッドによって要求された毎秒１００万の命令（ＭＩＰＳ）の数をバースト速度Ｂｎとして規定することができる。これにより、それぞれのスレッドが他のスレッドの作用と独立して作動できるように、プロセッサの負荷を均衡させることができる。全負荷がプロセッサの能力を決して超えない限り、発行速度制御は、全てのスレッドがそれらが必要とするプロセッサリソースを受け取ることを確実にするのに必要な唯一の機構である。

【００１５】

全てのスレッドによって要求される全体の負荷がプロセッサ能力を超える場合、スレッド優先機構は、最初に、プロセッサリソースが最高優先度スレッドに適用されることを可能にする。発行速度制御システムは、要求したプロセッサリソースを受け取らない優先順位の低いスレッドの処理不足をモニタし、プロセッサの過負荷が終了した時の均衡の回復を保証する。発行速度制御機構及び優先順位機構は、以下に説明される。

【００１６】

「ＭＣＣ」２によってモニタされているそれぞれのスレッドは、２つの状態のうちの何れかとすることができる。これらは、「待ち」と「実行」である。「待ち」状態では、スレッドは、トリガのような外部イベントに対してブロックされ、命令発行速度はゼロである。「実行」状態では、スレッドは普通に実行され、命令の発行速度は、図４の概略図に示されたバースト速度レジスタ４６（Ｂｎ）に書き込まれた値によって制御される。

一般的に、実行スレッドは、タスクが実行される時のプログラムのオペレーションと「実行」期間とを同期させるイベントのための「待ち」の期間を有することになる。いくつかの場合では、スレッドは、いかなる同期イベントの「待ち」もなく連続して実行することができる。実行の両方のパターンは、以下に説明されるイベント速度制御スキームによって処理される。

【００１７】

全てのプロセッサスレッドによって要求される全てのバースト速度の合計は、プロセッサの能力を超える場合がある。処理されている各スレッドには、命令発行に対するスレッドの選択を判断するために使用される割り当てられた優先順位がある。それぞれのクロックサイクルにおいて、「ＭＣＣ」は、新しい命令を受け入れることができるスレッドを識別し、優先順位が最高のスレッドを選択する。これにより、ある期間に対してプロセッサが過負荷の時、スローダウンしたのが重要度が最も低いスレッドであることが保証される。

優先順位制御機構は、発行速度制御機構がするように、「ＭＣＣ」内のスレッドスケジューリング論理と相互作用し、それはまた、任意選択的にクロック発生装置とも相互作用することができる。

【００１８】

優先順位及び命令発行速度の制御システム間のインタフェースを実行する信号、及び、プロセッサ命令スケジューラは、４つの命令スレッドを有するプロセッサに関して以下の表１に示される。

この数は、純粋に一例として選択されたものであって、他の任意のスレッド数も使用できる。

【００１９】

【表１】

優先順位信号		
入出力	信号名	説明
スケジューラへ	ＲＡＮＫ０ｉｄ	ＲＡＮＫ０ｉｄは、最低ランキングのスレッドのアイデンティティを表す。
スケジューラへ	ＲＡＮＫ１ｉｄ	ＲＡＮＫ１ｉｄは、４つのうちのランキング２のスレッドのアイデンティティを表す。
スケジューラへ	ＲＡＮＫ２ｉｄ	ＲＡＮＫ２ｉｄは、４つのうちのランキング３のスレッドのアイデンティティを表す。
スケジューラへ	ＲＡＮＫ３ｉｄ	ＲＡＮＫ３ｉｄは、最高ランキングのスレッドのアイデンティティを表す。
注意：Ｎスレッドプロセッサに対して、Ｎ信号ＲＡＮＫｘｉｄがある。		
発行速度制御信号		
入出力	信号名	説明
スケジューラへ	ＦＲＥＥＺＥｎ（ｎ＝０：３）	スレッドｎが命令の発行を停止したいことを表す。ＦＲＥＥＺＥｎは、命令スケジューラに対するインターロック信号として作用する。
スケジューラへ	ＲＥＱＵＥＳＴｎ（ｎ＝０：３）	スレッドｎが命令の発行を要求することを表す。
スケジューラから	ＧＲＡＮＴｎ（ｎ＝０：３）	命令がスレッドｎに発行されたことを表す。
スケジューラから	ＥＬＯＣＫＥＤｎ（ｎ＝０：３）	スレッドｎがトリガなどの外部イベントに対してブロックされたこと（待ち）を表す。

【００２０】

各プロセッササイクルに対する命令スケジューラの挙動は、一般に次の規則によって規定される。

１．ブロックされている（実行できない）全てのスレッドと、「ＦＲＥＥＺＥ」信号がオンの全てのスレッドとを無視する。

２．ブロックされず、「ＦＲＥＥＺＥ」信号をオンにしておらず、「ＲＥＱＵＥＳＴ」信号をオンにした、最高優先順位のスレッドをスケジュールする。

３．段階２でスレッドが何もスケジュールされていない場合、ブロックされず、「ＦＲＥＥＺＥ」をオンにしていない、最高数値の非「ＲＥＱＵＥＳＴ」中のスレッドをスケジュールする。

４．スレッドがスケジュールされている場合は、対応する「ＧＲＡＮＴ」信号をオンにする。

【００２１】

この機構は、スレッド命令発行速度に影響を与えるが、それをサイクル毎に直接制御することはない。

クロック発生器内にクロックゲート回路を任意に追加することにより、全ての信号「ＲＥＱＵＥＳＴｎ」がオフにされた時はいつも、プロセッサのクロックパルスを確実に削除することができる。すなわち、ゲートプロセッサのクロック速度は、全てのスレッドによって要求される全体の命令速度を達成するために、正確に要求された速度で実行するように制御することができ、従って、処理のための電力消費は最小限に維持される。

【００２２】

図１の「ＭＣＣ」２は、あらゆる重要な点で拡張可能である。何故ならば、それは、ストレージ（レジスタファイル）をローカライズするバンクで構成され、いかなる管理不能の経路指定や相互接続の問題を生じさせることなく、処理（ＡＬＵ）用の追加バンクを付加できるからである。次に、サポートすることができる処理スレッドの数は、マルチバン

クキャッシュ内に含まれるプログラムカウンタバンクにレジスタを追加し、それに応じて制御ユニットを変更することによって増やすことができる。「ＭＣＣ」によってサポートされる入力／出力の流れの数は、更に別のＩ／Ｏバンクを追加することによって増やすことができる。

【００２３】

スレッドのスケジューリングを処理する「ＭＣＣ」２の一部分のブロック図が図２に示されている。それは、データ処理ユニットの状態と、パイプライン及びメモリバンクの状態と、Ｉ／Ｏポートの状態とに関するリソース状態情報を、発行要求制御ユニット２２の各組からの「ＦＲＥＥＺＥｎ」信号出力に対する接続と共に受け取るリソースチェッカ２０を含む。これらの発行要求制御ユニット２２は、それぞれ、スレッド順位付けユニット２４に対して、「ＲＥＱＵＥＳＴｎ」出力と「ＤＥＬＡＹ_ＣＯＵＮＴｎ」出力とを有する。これはまた、それぞれの命令スレッドに付随する「ＰＲＩＯＲＩＴＹｎ」信号と「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」信号とを受け取る。これら２つの信号の信号ソースは、図４及び図５を参照して説明される。このスレッド順位付けユニットは、各命令スレッドに対するランキング信号を優先順位選択ユニット２６に対して送出する。

【００２４】

優先順位選択ユニット２６はまた、どのスレッドが実行するのに利用可能なリソースを持っているかを表示する、リソースチェッカ２０からのインターロック信号を受け取る。従って、最高順位のスレッドが利用可能リソースを全部持っていない場合は、優先順位選択ユニットは、利用可能リソースを有する２番目に上位の順位のスレッドを選択することになる。優先順位選択ユニット２６は、次のクロックサイクルでの命令をどのスレッドが実行することになるかを表示する「ＧＲＡＮＴｎ」出力を有する。この「ＧＲＡＮＴｎ」信号はまた、発行速度要求制御ユニット２２にフィードバックされる。

【００２５】

図２に示されたスケジューラとの間接的な対話特性のために、「ＲＥＱＵＥＳＴｎ」信号のオンと関連スレッドに対する命令の発行との間にいくらかの遅れが発生する場合がある。プロセッサが過負荷状態の場合、この遅れは非常に長くなり、そのために命令発行要求を保持するのにバッファが必要になる。発行要求バッファは、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２と呼ばれるカウンタとして実行され、発行要求制御ユニット２２の一部を形成する。

【００２６】

発行要求制御ユニットは、図４に詳細に示される。このユニットは、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０と、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２とを有する。「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０は、「ＭＡＸ_ＰＯＯＬ_ＣＯＵＮＴｎ」レジスタ４９から、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０内の最高許容値を表す値を受け取る。それはまた、平均速度レジスタ４８（Ａｎ）からの値を受け取る。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２は、その入力の１つで信号「ＢＬＯＣＫＥＤｎ」を受け取り、他の入力でバースト速度レジスタ４６（Ｂｎ）内の値を受け取る。「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０と、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２とは、両方とも図２の優先順位セレクタ２６から信号「ＧＲＡＮＴｎ」を受け取る。

【００２７】

「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２と、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０とは、スレッドの実行に互いに別々に影響を与えるので、別々に説明される。最初に、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２の作動について説明する。

スレッドは、そのバースト速度レジスタ４６に値を書き込むことにより、特定のタスクに必要な命令発行速度を規定する。バースト速度レジスタ４６内の値は、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２に繰り返し加算され、スレッドｎによって命令発行が要求される速度が表示される。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２は、信

号「ＧＲＡＮＴｎ」がオンにされる度に区分的に減らされ、そのためにプロセッサがＢｎ内の値によって規定された速度で命令を実行すると、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２内の値は０の近くに保たれることになる。プロセッサがＢｎ内の値によって規定された速度で命令を実行できない場合、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２内に正の値が残ることになる。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２内の値は、スレッドｎが現在のタスクのために必要であるが、まだ受け取っていない命令発行の数を表示する。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２内の値は、スレッド上の命令発行速度が制御されるように、スレッドｎに対する信号「ＲＥＱＵＥＳＴｎ」及び「ＦＲＥＥＺＥｎ」を発生させるために使用される。

【００２８】

必要な命令発行速度を規定するためにＢｎに書き込まれる値は、次の式によって計算される。

Ｂｎ＝ｒｅｑｕｉｒｅｄ_ｉｓｓｕｅ_ｒａｔｅ ／ ｄｅｌａｙ_ｃｏｕｎｔ_ｕｐｄａｔｅ_ｒａｔｅ

「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２は、プロセッサのクロックサイクル毎に更新することができ、その場合、Ｂｎは次の式によって計算される。

Ｂｎ＝ｒｅｑｕｉｒｅｄ_ｉｓｓｕｅ_ｒａｔｅ ／ ｐｒｏｃｅｓｓｏｒ_ｃｌｏｃｋ_ｒａｔｅ

【００２９】

電力消費を節約するために、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２は、ＫＲを速度制御選択係数とすると、ＫＲプロセッサクロックサイクルのそれぞれの累積期間に一度だけ更新することができる。それぞれの更新の際に、バースト速度レジスタ４６内の値は、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」に加算され、同時に「ＤＥＬＡＹ_ＣＯＵＮＴｎ」は、累積期間に「ＧＲＡＮＴｎ」がオンにされた回数だけ区分的に減らされる。この機構を使用することにより、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２内の各累積期間の最後の値は、仮にプロセッサのクロックサイクル毎に更新が加えられた場合に保持されたであろう値と確実に一致する。他の全ての時間においては、その値は異なってもよい。それにもかかわらず、より粗いタイミング分解度という犠牲を払って、電力消費は係数ＫＲだけ低減される。

【００３０】

ＫＲの値は。電力消費と命令発行制御の細分性との間の妥協点として選択されるべきである。指針として４～１６の範囲のＫＲの値は、許容できるタイミングの細分性をもたらし、一方で電力消費を妥当なレベルに保つことが期待される。

「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２が、ＫＲプロセッサクロックサイクル毎に一度更新されると、Ｂｎは、次の式によって計算される。

Ｂｎ＝ｒｅｑｕｉｒｅｄ_ｉｓｓｕｅ_ｒａｔｅ＊ＫＲ ／ ｐｒｏｃｅｓｓｏｒ_ｃｌｏｃｋ_ｒａｔｅ

【００３１】

スレッド上で実行されているプログラムは、新しい命令発行速度を規定するためにそのバースト速度レジスタ４６に何時でも書き込むことができる。これは、通常、プロセッサの活動を必要とするイベントに応答して、又は、タスクの完了に応答して行われることになる。

「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータに対する「ＢＬＯＣＫＥＤｎ」信号は、図２のリソースチェッカ２０によって生成され、スレッド上の実行がブロックされ、外部イベント待ちであることを表示する。「ＢＬＯＣＫＥＤｎ」がオンにされると、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２の更新は禁止される。「ＢＬＯＣＫＥＤｎ」がオフにされてイベントが発生したことが表示されると、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２は、最初にクリアされ、続いて、値Ｂｎが加算され、信号「ＧＲＡＮＴｎ」がオンにされるたびに１が減じられて、累積期間毎に一度更新される。

「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２の値は、数値ラッピングに起因する

エラーを回避するために、それが表示できる最大の２の補数値に制限される。作動時には、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２は、命令発行速度がバースト速度レジスタ４６によって規定された速度と一致することを保証するが、その値が最高値に達しないことを条件とする。

【００３２】

図４の「ＲＥＱＵＥＳＴｎ」発生ユニット４４は、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２から出力を受け取り、これに応答してスレッド順位付けユニットに対して「ＲＥＱＵＥＳＴｎ」信号を生成する。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」信号はまた、スレッド順位付けユニットに直接出力される。図４の「ＦＲＥＥＺＥｎ」発生ユニット４５は、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２及び「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０から出力を受け取り、これに応答して、リソース検査ユニット２０に「ＦＲＥＥＺＥｎ」信号を発生する。

【００３３】

「ＲＥＱＵＥＳＴｎ」発生ユニット４４は、次の累積期間の最初に「ＤＥＬＡＹ_ＣＯＵＮＴｎ」の値を調べ、「ＲＥＱＵＥＳＴｎ」を適切な回数オンにすることにより、次の累積期間に要求される命令の数を判断する。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」がゼロに等しいか又はそれ以下の場合、「ＲＥＱＵＥＳＴｎ」はオンにされないことになる。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」の値がＫＲに等しいか又はそれ以上の時、「ＲＥＱＵＥＳＴｎ」は、ＫＲ回オンにされることになる。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」の値がゼロを超え、ＫＲよりも小さい時、「ＲＥＱＵＥＳＴｎ」は、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」回オンにされることになる。「ＲＥＱＵＥＳＴｎ」発生ユニット４４は、「ＲＥＱＵＥＳＴｎ」が適切な回数オンにされることで、命令要求が累積期間に亘って最大に分布されることを保証する。

【００３４】

「ＧＲＡＮＴｎ」の連続するオンにより、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」はゼロよりも小さくなることができ、その結果、「ＲＥＱＵＥＳＴｎ」はオフにされ、プロセッサの処理能力が別のスレッドを実行するために解放される。バースト速度のレジスタ４６に引き続き値を加算することにより、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」はゼロを上回り、それによりそのスレッドに対する「ＲＥＱＵＥＳＴｎ」がオンにされることになる。

【００３５】

「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２の出力は、「ＦＲＥＥＺＥｎ」信号をリソースチェッカに送る「ＦＲＥＥＺＥｎ」信号発生ユニット４５に接続される。この「ＦＲＥＥＺＥｎ」信号発生ユニット４５に対する他の入力は、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０の出力である。上述の任意選択のクロックゲートが実施されると、スレッドｎ上の命令発行速度は、信号「ＲＥＱＵＥＳＴｎ」によって制御され、実施されない場合、命令発行速度は、信号「ＦＲＥＥＺＥｎ」によって制御される。

上述の任意選択のクロックゲートが実施されると、「ＦＲＥＥＺＥｎ」は、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」が－１５よりも小さい時にオンにされ、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」がゼロを超えるとオフにされる。クロックゲートが実施されない時、「ＦＲＥＥＺＥｎ」は、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」がゼロよりも小さい時にオンにされ、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」がＢｎを超えるとオフにされる。

【００３６】

「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２の作動は、命令が発行される速度を、それがバースト速度レジスタ４６で規定された速度と一致するように制御する。これは、スレッドの負荷の変化に応じて頻繁に変更される場合があり、従って、１つのスレッドに起因する全プロセッサ負荷が制限されてそれが規定値を超えないことを保証するのは困難である。プロセッサの平均負荷を把握し、プロセッサが過負荷を持続するのを制限し、１つのスレッドがプロセッサを占有できないことを保証するために、値「ＰＯＯＬ_ＣＯＵＮＴｎ」を有する「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０が設けられる。「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０の作動について以下に説明する。

【００３７】
必要な平均命令発行速度を規定するスレッドｎは、これをその平均速度レジスタ４８（Ａｎ）に書き込むことによって行う。「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０は、「Ａｎ」内の値によって規定された通常の速度で区分的に増加され、そのスレッドに対する「ＧＲＡＮＴｎ」がオンにされる度に区分的に減らされる。これは、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータを更新するために使用されたのと同じＫＲプロセッサのクロックサイクル累積期間を使用して行われる。「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０がＫＲプロセッサのクロックサイクル毎に一度更新される場合、Ａｎは次の式で計算される。

Ａｎ＝ａｖｅｒａｇｅ_ｉｓｓｕｅ_ｒａｔｅ＊ＫＲ ／ ｐｒｏｃｅｓｓｏｒ_ｃｌｏｃｋ_ｒａｔｅ

【００３８】
スレッド上で実行されるプログラムは、通常、そのプログラムの開始の時にその平均速度レジスタ４８に一度だけ書き込むことになる。それが要求する平均命令発行速度が確立されると、平均速度レジスタ４８上の値は通常変更されない。

「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０内の値は、スレッドがそのタスクを完了させるために必要であるが、まだ受け取っていない命令発行の数を表す。また、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０内の値は、プロセッサ過負荷の最大可能継続時間を推定するために、別のスレッド上で実施されているプログラムによって読み出すことができる。「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０がゼロ又は負の値になると、それらがスレッドｎがその命令発行割り当てを既に使い果たしたことを表すので、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２の状態に関係なく、「ＲＥＱＵＥＳＴｎ」は、そのスレッドに対してオフにされる。

【００３９】
「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０内の値は、「ＭＡＸ_ＰＯＯＬ_ＣＯＵＮＴｎ」レジスタ４９によって規定された最高値に限定される。これは、スレッドが後でより高速で実行するために「信用で」蓄積することができる命令の数を規定し、この値に制限を設けることは、プロセッサ過負荷の継続期間を制限することを可能にする。「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０が「ＭＡＸ_ＰＯＯＬ_ＣＯＵＮＴｎ」レジスタ４９内の値に到達すると、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０の更なる区分的増加が禁止される。すなわち、高速で実行する前に外部イベントを待つスレッドは、それが待つ間に「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０内に命令の「リザーバ」を構築することができ、その後、イベントに続く制限された時間に、バースト速度レジスタ４６によって規定された高速で実行する。高速実行バーストの間に蓄積される他のスレッド内のいかなる処理の不足も、それらの「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２の作動によって後で均衡されることになる。

【００４０】
「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０内の値が負の値になるように、スレッドに追加命令を発行する機能を提供することが可能である。これは、他のあらゆるスレッドがその要求した命令速度を受け取るのをそれが妨げない場合にだけ行うことができる。換言すれば、この機能により、予備の命令を別のスレッドに割り当てることが可能となり、それによってプロセッサの使用が最適化される。

【００４１】
オペレーションのデフォルトモードは「サイクル・ストリクト」と呼ばれる。このモードでは、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータがゼロ又は負になると、図４の「ＦＲＥＥＺＥｎ」発生ユニット４４を通じて「ＦＲＥＥＺＥｎ」はオンにされ、平均速度レジスタ４８の値による区分的増分の結果として、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０が再度ゼロを超えるまで、スレッドはそれ以上の命令発行を受け取らないことになる。すなわち、スレッドがプロセッサ上に課すことができる強い制限が負荷に対して強制される。

【００４２】
任意選択の「緩いサイクル」モードでは、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０が１よりも小さい時は、「ＦＲＥＥＺＥｎ」はオンにされない。この状態が発生して、命令発行を受け取るのに他のスレッドがどれも利用可能でない時に空の命令プールを有するスレッドを実行することができる場合、プールが空のスレッドに対して命令スケジューラが命令を発行することができ、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０の値を更に負の値にさせる。
【００４３】
オペレーションモードに関係なく、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０が１よりも小さい時に、「ＲＥＱＵＥＳＴｎ」がオフにされることに注意すべきである。「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０が決して空にならないように、リアルタイムシステムが通常設計されることになることにも注意すべきである。
「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０が、それが表示できる最大の負の値に到達すると、アキュムレータの値が負の値から正の値に循環しないこと保証するために、それ以上の区分的減少が禁止される。
【００４４】
従って、要約すると、図４の回路により、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２に応答して「ＲＥＱＵＥＳＴｎ」信号が生成され、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２又は「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０に応答して「ＦＲＥＥＺＥｎ」信号が生成されることになる。「ＲＥＱＵＥＳＴｎ」及び「ＦＲＥＥＺＥｎ」のどちらもオンにされない場合には、他のどのスレッドも実行することができない場合、スレッドｎに対して命令が発行されることになるが、実行することができる他のいかなるスレッドもスレッドｎよりも優先順位が高いことになる。
【００４５】
プロセッサ上で実行される命令スレッドには、いくつかの異なる可能な状態がある。それらは以下の通りである。
「実行」状態
この状態において、命令は、バースト速度レジスタ４６内の値によって規定される速度で通常発行される。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」の値は、別のスレッドのバースト実行速度が高くない限り、通常はゼロの近くに留まることになる。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２内の値は、停止か又は他のスレッドのオペレーションの何れかによってスレッドがその意図した実行パターンから遅延された命令数を表示する。
【００４６】
「ＰＯＯＬ_ＥＭＰＴＹｎ」状態
「ＰＯＯＬ_ＥＭＰＴＹｎ」アキュムレータ４０が特定のスレッドに関してゼロ又はゼロよりも小さい場合、そのスレッドは、「ＰＯＯＬ_ＥＭＰＴＹｎ」状態であるとして規定される。この状態では、スレッドは、実行状態での作動のように作動するが、スレッドがサイクルストリクトモードで作動している場合、発行速度は、その平均速度レジスタ４８からの値「ＡＮ」に制限することができる。スレッドｎが「ＡＮ」を超える発行速度で長時間実行しようとするイベントにおいては、他のスレッドに保護がもたらされる。通常の作動では、「ＰＯＯＬ_ＥＭＰＴＹｎ」状態には決して入るべきではない。
【００４７】
「待ち」状態
スレッドがトリガなどの外部イベントに対してブロックされると、「ＢＬＯＣＫＥＤｎ」がリソース検査ユニット２０によってオンにされ、次に、スレッドは、「待ち」として規定される。上述の任意選択のクロックゲートが実施されると、「待ち」状態のスレッドに対してプロセッサのクロックパルスが生成されることはない。
図２の優先順位制御選択ユニット２６は、図１のメディア制御コア内の命令スケジューラに対して、スレッド選択のランク順位を提供する。全入力スレッドに対する全バースト速度Ｂｎの合計が、利用可能なプロセッサ命令速度よりも小さい時の条件においては、優

先順位機構は正味の影響を及ぼさない。換言すれば、優先順位制御ユニットは、プロセッサが過負荷の場合だけ関連がある。

【００４８】

プロセッサが過負荷の場合は、命令スケジューラは、インターロックされていない最高順位のスレッドに対して命令を発行する。スレッドのランク順位を確立するために、図３に示されるように３つのメトリックの組合せが使用される。このプロセッサについては以下に説明される。

それぞれの命令スレッドに関連するスレッド順位付けユニットに対する入力には、０、１、２、及び、３の番号が付けられる。スレッド０だけが完全に図示されている。スレッドに対する優先順位信号は、シフトレジスタに入力され、そこから加算器に供給される。「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴ」及び「ＤＥＬＡＹ_ＣＯＵＮＴｎ」は、それぞれ、類似のシフトレジスタ３０に通される前にフォーマット変換ユニット３２に入り、３つが全て加算器３４内で結合される前にそれらを優先信号で正規化し、従って、そのスレッドに関連するメトリックを生成する。他の各スレッドに対する同様のメトリックが作られ、これらはランク順位比較器３６で比較される。これは、それぞれのスレッドに次に順位を割り当てるゲートユニット３８を含む。これらの順位は並列に出力される。

【００４９】

ランク順位を確立するのに最重要のメトリックは、スレッド優先順位である。スレッドは、符号のないバイト値をその優先順位レジスタに書き込むことによりその優先順位を規定する。この優先順位レジスタ内の数字が大きいほど、その順位が高い。

２番目に重要なメトリックは、図５に示された「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」であり、これは、スレッド順位付けユニット２４に対して第２の入力を供給する。「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」機構により、スレッドは、その時間内にタスクを完了しなければならない締切時間を規定することができる。「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」レジスタ５０は、スレッドが「待ち」状態の時は静止状態に保たれ、それはまた、「ＤＥＡＤＬＩＮＥ_ＤＥＦＡＵＬＴｎ」に対して初期化され、続いてスレッドが外部イベントに応答して「待ち」状態から「実行」状態に変わると通常の速度で区分的に減少される。

【００５０】

締切カウンタ５０に対する入力は、リソース検査ユニット２０からの「ＢＬＯＣＫＥＤｎ」信号、そのスレッドが実行するための最大の締切を示す、「ＤＥＡＤＬＩＮＥ_ＤＥＦＡＵＬＴｎ」レジスタに保持された値、スレッドがそれぞれ締切が異なるいくつかの異なるイベントを処理する時に締切を調節するために「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」に加算される「ＤＥＡＤＬＩＮＥ_ＩＮＣＲＥＭＥＮＴｎ」値、及び、「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」カウンタの作動を可能にする「ＤＥＡＤＬＩＮＥ_ＥＮＡＢＬＥｎ」信号である。

【００５１】

初期設定値は、イベントから最短の締切が過ぎるまでの時間を表す。「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」レジスタ５０は、次に通常の速度で区分的に減少され、その値が締切が過ぎるまでの残り時間を表示することを保証する。電力消費を節約するために、「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」は、ＫＤクロックサイクル期間毎に一度だけ更新され、ここで、ＫＤは、図４の「ＤＥＬＡＹ_ＣＯＵＮＴｎ」アキュムレータ４２の更新と同じ方法による締切カウンタの選択係数である。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」が減少すると、そのスレッドのランキングも増加するはずである。全てのスレッドに締切優先順位機構が設けられた場合、全ての締切に確実に間に合うように、プロセッサリソースが最良の方法で割り当てられる。通常のシステムでは、締切のないスレッドが存在する場合があり、従って、全てのスレッドがこの機構を使用することが絶対に必要ということはない。

【００５２】

スレッドが「実行」状態から「待ち」状態に戻ると、「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」レジスタは区分的増加を停止し、それが「待ち」状態を再度離れるまでその最終値を保持し、その時点で、それは「ＤＥＡＤＬＩＮＥ_ＤＥＦＡＵＬＴｎ」の新しい値に再初

期化される。
スレッドは、それぞれ締切が異なるいくつかの異なるイベントを処理することができる。この場合、「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」５０に対する「ＤＥＡＤＬＩＮＥ_ＤＥＦＡＵＬＴｎ」レジスタ入力は、処理される最短の締切を表示するようにプログラムされる。締切の長いイベントが発生する場合、そのスレッドは、レジスタの「ＤＥＡＤＬＩＮＥ_ＩＮＣＲＥＭＥＮＴｎ」（図示しない）に締切の区分的増分を書き込み、次にこれが締切カウンタ５０内の値に加算される。

【００５３】
締切カウンタ５０に対する「ＤＥＡＤＬＩＮＥ_ＥＮＡＢＬＥｎ」入力により、制御ビットを用いて締切カウンタのオンオフの切換ができる。締切カウンタが使用不可にされた場合、「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」は、その最大値に設定される。更に別の入力（図示しない）を使用して、ソフトな締切スケジューリング中の「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」の区分的減少を休止させることができる。

【００５４】
締切カウンタ５０は、締切を有するイベントが発生した時にスレッドｎが「待ち」状態にある場合にのみ正常に作動する。これは、スレッドが１つのイベントだけを処理し、次のイベントの前にイベント処理が終了しなければならない場合にのみ保証することができる。そのような制約が課せられていない場合には、締切制御は、そのタスク専用の別の処理スレッドによって処理されるべきである。
そのような専用の締切制御スレッドは、全イベントの初期の処理を実行し、締切が過ぎる時間を認識し、それらのイベントを適切なスレッドの待ち行列に入れるであろう。このスレッドによって為される仕事は、イベントを検出するのにかかる時間をできるだけ短くするためにできるだけ最小に抑えられる。スレッドがその待ち行列からイベントを拾い上げる時、それは、締切制御スレッドを通じてタスク待ち行列エントリに記録された量だけ、そのスレッドに対する「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」を調節する。調節値は、イベントが発生した時間に締切制御スレッドによって計算されることになる。

【００５５】
スレッド順位付けユニットによって使用される最も重要度が低いメトリックは、上述の「ＤＥＬＡＹ_ＣＯＵＮＴｎ」である。このメトリックがスレッドの順位付けを制御するために使用される場合、プロセッサが過負荷のイベントにおいては、命令スケジューラは、全てのスレッドが意図した処理プロフィールから同じ遅れを受けるようにプロセッサリソースを割り当てる。
それぞれのスレッドに対して、図３の加算器３４における連結により、３つのメトリックが単一の複合メトリックに結合される。優先順位は、最重要の位置と、最も重要度の低い「ＤＥＬＡＹ_ＣＯＵＮＴｎ」とにある。
フォーマットコンバータ３２は、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」に対してビット的反転を行い、カウントの減少が数の増加を与えることを保証する。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」は、その最上位のビットを反転させ、それを２の補数からオフセット／２進表示に変換する。これは、負のカウント値を誤って解釈するのを回避するためである。

【００５６】
複合メトリックは、バブルソート・アルゴリズムとして公知の連続実施法を使用するランク順位比較器３６により、ランクの順番に配置される。これは、次の通りである。
１．ソータの各クロックサイクル上で２つの複合メトリックが比較され、下位の順位付けの値が上位の順位付けの値と等しいか又はそれ以上の場合、それらの順位が交換される。
２．連続するクロックサイクル上で、複合メトリックの最下位の順位の対から最上位の順位の対に向って比較が続けられる。
３．複合メトリックの最上位の順位の対の比較に続くクロックサイクル上で、最下位の順位の対が比較される。

【００５７】

このアルゴリズムの効果は、Ｎ－１以下（Ｎはスレッドの数）のクロックサイクルで最大複合メトリックを最上位の順位の位置に置くことであり、最大N^2／２－Ｎ／２クロックサイクルで正しい順番を確立することができる。２つ又はそれ以上の複合メトリックの値が等しい場合は、値が等しいと交換が発生するために、それらのランク順位は、ある時間に亘って循環することになる。この挙動は、タイムスライシングによって同じ順位付けをエミュレートし、従って、複合メトリックが同等のスレッドは全て、命令スケジューラから同等の優先順位を受け取ることを保証する。

説明したスキームは、プロセッサの処理限界を超えないことを条件として、全てのスレッドに対する命令発行速度を制御することになる。仮に処理限界を超える場合、デバッグと特徴付けを支援するための検出機構の提供が役立つであろう。これらについては以下に説明する。

【００５８】

命令発行の不足及び過剰をモニタ及び制御するために、それぞれのスレッドに対して２つのトリガが設けられる。「ＰＯＯＬ_ＦＩＬＬＥＤｎ」トリガは、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０の値が「ＭＡＸ_ＰＯＯＬ_ＣＯＵＮＴｎ」レジスタ４９内の値と等しい時に設定される。「ＰＯＯＬ_ＥＭＰＴＹ」トリガは、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０の値がゼロ又は負の時に設定される。次に、「ＰＯＯＬ_ＦＩＬＬＥＤｎ」及び「ＰＯＯＬ_ＥＭＰＴＹｎ」の状態と、「ＰＯＯＬ_ＣＯＵＮＴｎ」アキュムレータ４０の値とを、状態レジスタから読み出すことができる。「ＤＥＬＡＹ_ＣＯＵＮＴｎ」の値は、状態レジスタから読み出すことができる。締切優先順位機構は、それぞれのスレッドに対するトリガによってモニタすることができる。これは、「ＤＥＡＤＬＩＮＥ_ＭＩＳＳＥＤｎ」と呼ばれ、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」がゼロになった時に設定される。「ＤＥＡＤＬＩＮＥ_ＭＩＳＳＥＤｎ」の状態と、「ＤＥＬＡＹ_ＣＯＵＮＴｎ」の値とは、両方とも状態レジスタから読み出すことができる。

ここで説明した発行速度制御システムのレジスタ及びトリガの要約表は、以下の表２に示される。

【００５９】

【表２】

入出力	名称	フォーマット			説明
		N	F	T	
R／W	Ａｎ	８	－４	ｔ	（ＭＩＰＳでの平均命令発行速度）＊ＫＲ_／Ｒ＋
R／W	Ｂｎ	８	－４	ｔ	（ＭＩＰＳでのバースト命令発行速度）＊ＫＲ／Ｒ
R／W	ＰＲＩＯＲＩＴＹｎ	８	０	ｕ	スレッド優先順位
R／W	ＤＥＡＤＬＩＮＥ_ＤＥＦＡＵＬＴＮ	２０	０	ｕ	スレッドに対するデフォルト締切（単位ＫＤ＊／Ｒ）
Ｗ	ＤＥＡＤＬＩＮＥ_ＩＮＣＲＥＭＥＮＴｎ	２０	０	ｕ	締切カウンタを拡張する値（単位ＫＤ／Ｒ）
R／W	ＭＡＸ_ＰＯＯＬ_ＣＯＵＮＴｎ	２２	０	Ｕ	命令プールカウンタの最大値
R／W	ＡＭＡ_ＣＯＮＴＲＯＬｎ	３	０	ｕ	ＡＭＡ制御レジスタ
Ｒ	ＤＥＬＡＹ_ＣＯＵＮＴｎ	２７	－４	ｔ	スレッドｎがその必要なバースト速度から遅延された命令の数
R／W	ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ	２０	０	ｕ	イベント締切期限＃までに残っているサイクル数
Ｒ	ＰＯＯＬ_ＣＯＵＮＴｎ	２７	－４	ｔ	スレッドが利用可能な潜在的な命令の数
Ｒ	ＤＥＡＤＬＩＮＥ_ＭＩＳＳＥＤｎ	－	－	－	締切が過ぎたことを示すトリガ
Ｒ	ＰＯＯＬ_ＦＩＬＬＥＤｎ	－	－	－	命令プールが充たされた（「ＭＡＸ_ＰＯＯＬ_ＣＯＵＮＴｎ」に達した）ことを示すトリガ
Ｒ	ＰＯＯＬ_ＥＭＰＴＹｎ	－	－	－	命令プールが空になったことを示すトリガ

【００６０】

＋Ｒ：ＭＨｚで表したプロセッサのクロック速度。

＃：２０ビットの「ＤＥＬＡＹ_ＣＯＵＮＴｎ」が、Ｄ｛２３．．．４｝の位置に読み出され、「ＭＣＣ」によって読み出されたカウント値が残っているサイクル数（ＫＤの次の最も低い倍数に切り捨てられた）であることを保証する。

フォーマット：

Ｎ－ビットの全数

Ｆ－端数ビットの数

Ｔ－データストレージの種別：ｕ－符号のないデータ、ｔ－２の補数データ

「ＡＭＡ」制御レジスタ：ビットは以下に対して準備される。

－サイクル・ストリクト・オペレーション

－締切の使用不可

－締切の休止

【００６１】

表２に与えられた２３整数ビットの「ＤＥＬＡＹ_ＣＯＵＮＴｎ」は、最大2^{22}、すなわち、約４００万の命令まで累積する。５０ＭＨｚの命令発行速度では、カウンタは、約８０ミリ秒の間累積することができる。これは、「ＰＯＯＬ_ＣＯＵＮＴｎ」に関しても同様である。

２０ビットの「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」は、最高2^{20}まで累積する。締切選択係数（ＫＤ）の１６が与えられると、「ＤＥＡＤＬＩＮＥ_ＣＯＵＮＴｎ」は、従って

最大約１６７０万サイクルまで表すことができる。２００ＭＨｚのプロセッサに対して、これは、約８４ミリ秒の締切を表す。

【図面の簡単な説明】

【００６２】

【図１】マルチスレッドプロセッサシステムの基本アーキテクチャのブロック図である。

【図２】図１の「媒体制御コア」のスレッドスケジューリング部分のブロック図である。

【図３】図２のスレッド順位付け回路のブロック図である。

【図４】図２の発行要求制御回路のブロック図である。

【図５】締切カウンタの構成を示す図である。

【符号の説明】

【００６３】

４０　プール計数アキュムレータ

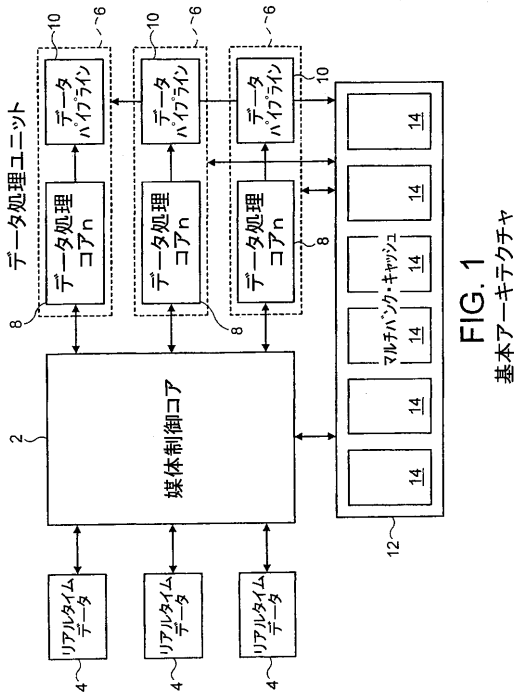
４２　遅延計数アキュムレータ

４４　「ＲＥＱＵＥＳＴ」発生、「ＦＲＥＥＺＥ」発生

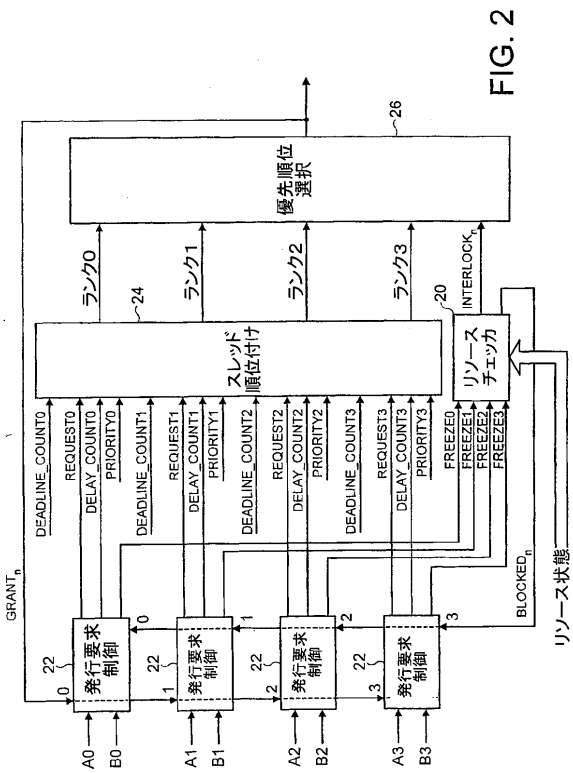
４６　バースト速度レジスタ

４８　平均速度レジスタ

【図１】



【図２】



【図３】

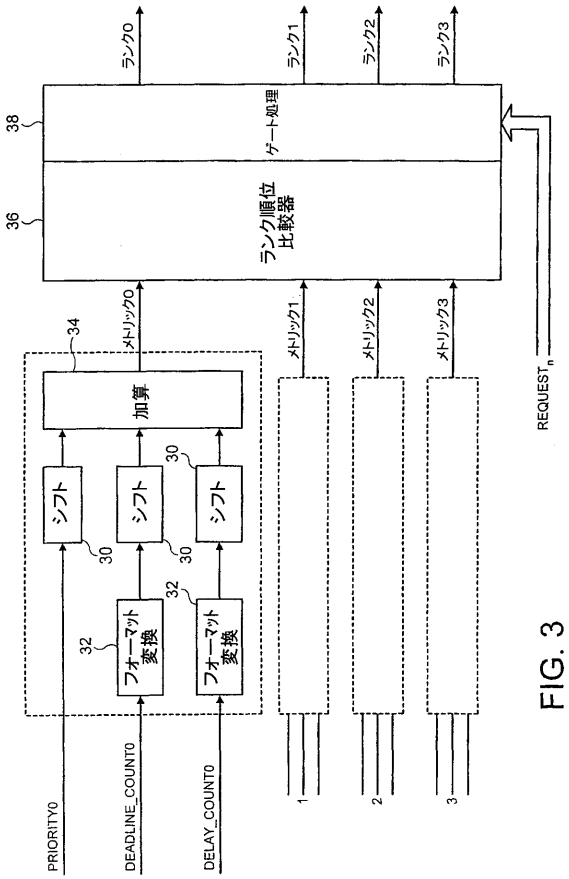


FIG. 3

【図４】

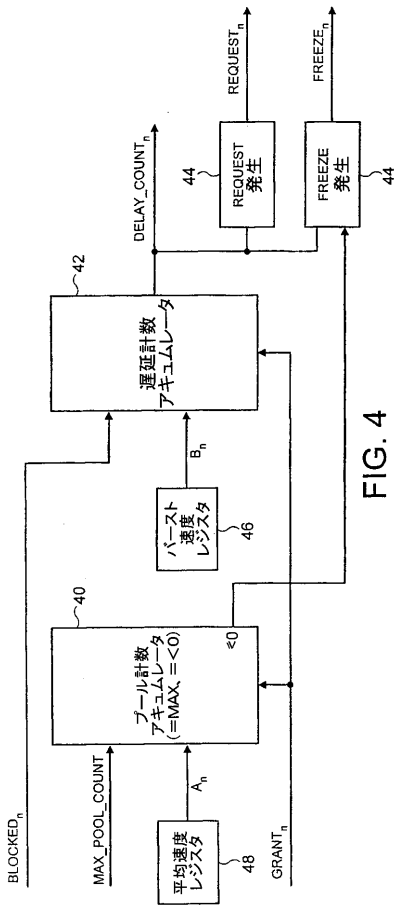


FIG. 4

【図５】

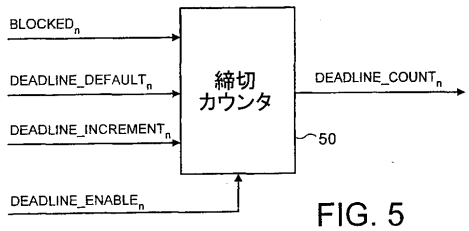


FIG. 5

フロントページの続き

(74)代理人 100074228
弁理士 今城 俊夫
(74)代理人 100084009
弁理士 小川 信夫
(74)代理人 100082821
弁理士 村社 厚夫
(74)代理人 100086771
弁理士 西島 孝喜
(74)代理人 100084663
弁理士 箱田 篤
(72)発明者 アンダーソン アドリアン ジョン
イギリス ハートフォードシャー ダブリューディー４ ８エルゼット キングス ラングリー ホーム パーク エステイト イマジネイション テクノロジーズ リミテッド内
(72)発明者 ウッドヘッド マーティン ジョン
イギリス ハートフォードシャー ダブリューディー４ ８エルゼット キングス ラングリー ホーム パーク エステイト イマジネイション テクノロジーズ リミテッド内

審査官 後藤 彰

(56)参考文献 特表２０００－５０９５２８（ＪＰ，Ａ）
特表２０００－５０４４５４（ＪＰ，Ａ）
特開平１１－２３７９９５（ＪＰ，Ａ）
特開昭５７－１０１９４８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/30 - 9/38
G06F 9/46 - 9/48