(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-171573
(P2004-171573A)
(43)公開日　平成16年6月17日(2004.6.17)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 9/38	G06F	9/38　370C	5B013

審査請求 未請求　請求項の数 26　ＯＬ　（全 20 頁）

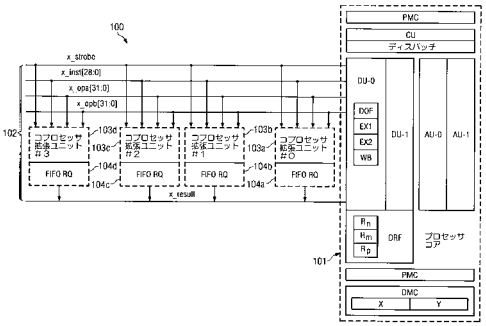
(21)出願番号　特願2003-389690(P2003-389690)
(22)出願日　平成15年11月19日(2003.11.19)
(31)優先権主張番号　10/299120
(32)優先日　平成14年11月19日(2002.11.19)
(33)優先権主張国　米国(US)

(71)出願人　591236448
エスティーマイクロエレクトロニクス，インコーポレイテッド
ＳＴＭｉｃｒｏｅｌｅｃｔｒｏｎｉｃｓ，Ｉｎｃ
アメリカ合衆国，　テキサス　７５００６，　カーロルトン，　エレクトロニクス　ドライブ　１３１０
(74)代理人　100076185
弁理士　小橋　正明
(72)発明者　シバグナナム　パーササラシー
アメリカ合衆国，　カリフォルニア　９２００９，　カールスバッド，　コルテ　マンゴ　２１８７

最終頁に続く

(54)【発明の名称】新規な分割命令トランズアクションモデルを使用して構築したコプロセッサ拡張アーキテクチャ

(57)【要約】
【課題】　プロセッサ命令セットアーキテクチャを拡張又は変更する改良した技術を提供する。
【解決手段】　プロセッサアーキテクチャは、拡張ユニットへオペランド及び命令を供給し且つ拡張ユニットから結果を検索するために分割命令トランズアクションを使用して、プロセッサコアを計算命令を実行する１つ又はそれ以上のコプロセッサ拡張ユニットへ結合させる電気的インターフェースをサポートする。拡張ユニットへ演算及びデータを送り及び／又は拡張ユニットからデータを検索するための包括的な命令は、プロセッサアーキテクチャの再生無しで新たな計算命令を導入することを可能とする。複数個の拡張ユニット及び／又は各拡張ユニット内の複数個の実行パイプ、マルチサイクル実行レイテンシー及び拡張ユニット間又はその中における異なる実行レイテンシー、拡張ユニット命令述語、及びプロセッサコアストール及びインタラプトに関する結果保存／回復を取扱うためのサポートが包含されている。
【選択図】　図１



【特許請求の範囲】
【請求項１】
プロセッサ拡張メカニズムにおいて、
プロセッサコア、
１つ又はそれ以上のオペランドに関し少なくとも１つの計算演算を実施するために前記プロセッサコアを拡張ユニットへ結合させることが可能であり、１つ又はそれ以上の命令及びオペランドバス及び結果バスへの接続を包含している電気的インターフェース、
拡張ユニット内において計算演算の実行を開始させるために前記電気的インターフェース上に第一命令を発行し且つ前記計算演算の結果を検索するために前記電気的インターフェース上に第二命令を発行することが可能な制御器、
を有していることを特徴とするプロセッサ拡張メカニズム。
【請求項２】
請求項１において、前記制御器が前記プロセッサコア内にデータユニットを有していることを特徴とするメカニズム。
【請求項３】
請求項１において、前記制御器が、
演算及びデータを拡張ユニットへ送るための命令、
拡張ユニットへ演算を送り且つ拡張ユニットからのデータをロードするための命令、
同時的に演算及びデータを拡張ユニットへ送り且つその拡張ユニットからのデータをロードする命令、
演算を拡張ユニットへ送る命令、
をサポートすることを特徴とするメカニズム。
【請求項４】
請求項１において、前記電気的インターフェースが、前記プロセッサコアを、各々が１つ又はそれ以上のオペランドに関して少なくとも１つの計算演算を実施する複数個の拡張ユニットへ結合させることが可能であることを特徴とするメカニズム。
【請求項５】
請求項１において、更に、
各々が異なる拡張レイテンシーを有する２つ又はそれ以上の命令を実行することが可能な拡張ユニット、
を有していることを特徴とするメカニズム。
【請求項６】
請求項５において、更に、
前記拡張ユニットと前記制御器との間に結合されており且つ前記拡張ユニットによって実行された計算命令の結果を格納する結果キュー、
を有していることを特徴とするメカニズム。
【請求項７】
請求項６において、更に、
ＯＲ論理又はトライステート装置のいずれかを使用して前記結果キューから前記結果バスへの接続、
を有していることを特徴とするメカニズム。
【請求項８】
請求項１において、前記制御器が、更に、
実行ユニット内において前に開始された演算の結果を保存し且つ回復するメカニズム、
を有していることを特徴とするメカニズム。
【請求項９】
請求項１において、前記プロセッサの設計を再生することなしに拡張を付加、除去又は変更するために前記プロセッサに対する設計を修正することが可能であることを特徴とするメカニズム。
【請求項１０】

プロセッサ内において拡張した命令を実行する方法において、
プロセッサコアを操作し、
１つ又はそれ以上のオペランドに関して少なくとも１つの計算演算を実施するために前記プロセッサコアを拡張ユニットへ結合させることが可能であり且つ１つ又はそれ以上の命令及びオペランドバス及び結果バスへの接続を包含している電気的インターフェースを駆動し、
実行ユニット内において計算命令の実行を開始させるために前記電気的インターフェース上に第一命令を発行し、
前記計算命令の結果を検索するために前記電気的インターフェース上に第二命令を発行する、
ことを包含していることを特徴とする方法。
【請求項１１】
請求項１０において、前記プロセッサコア内のデータユニットが前記第一及び第二命令を発行することを特徴とする方法。
【請求項１２】
請求項１０において、前記第一及び第二命令が、
演算及びデータを拡張ユニットへ送るための命令、
拡張ユニットへ演算を送り且つ拡張ユニットからデータをロードするための命令、
同時的に演算及びデータを拡張ユニットへ送り且つその拡張ユニットからのデータをロードするための命令、
拡張ユニットへ演算を送るための命令、
のうちの１つを有していることを特徴とする方法。
【請求項１３】
請求項１０において、前記電気的インターフェースが、前記プロセッサコアを、各々が１つ又はそれ以上のオペランドに関して少なくとも１つの計算演算を実施する複数個の拡張ユニットへ結合させることを特徴とする方法。
【請求項１４】
請求項１０において、更に、
各々が拡張ユニット内において異なる実行レイテンシーを有している２つ又はそれ以上の命令を実行する、
ことを包含していることを特徴とする方法。
【請求項１５】
請求項１４において、更に、
前記実行ユニットと前記コアとの間に結合されている結果キュー内に実行ユニットによって実行された計算命令の結果を格納する、
ことを包含していることを特徴とする方法。
【請求項１６】
請求項１５において、更に、
ＯＲ論理又はトライステート装置のいずれかを使用して前記結果キューからの結果を前記結果バスへ送信する、
ことを包含していることを特徴とする方法。
【請求項１７】
請求項１０において、更に、
インタラプト又はコアストール期間中に実行ユニット内において前に開始された演算の結果を選択的に保存及び回復する、
ことを包含していることを特徴とする方法。
【請求項１８】
請求項１０において、前記プロセッサの設計を再生することなしに拡張ユニットを付加、除去又は変更するために前記プロセッサに対する設計を修正することが可能であることを特徴とする方法。

【請求項１９】
プロセッサにおいて、
データユニットを包含するプロセッサコア、
１つ又はそれ以上のオペランドに関して各々が少なくとも１つの計算演算を実施する１つ又はそれ以上の拡張ユニット、
各々が前記データユニットを前記１つ又はそれ上の拡張ユニットの各々へ結合させる命令バス、少なくとも１つのオペランドバス、及び結果バス、
を有しており、前記データユニットが、実行ユニット内の計算命令の実行を開始させるために前記命令バス上に演算及びデータを拡張ユニットへ送る命令コンポーネント及び前記結果バス上に前記計算命令の結果の検索を開始するために前記命令バス上に拡張ユニットからのデータをロードする命令を発行することを特徴とするプロセッサ。
【請求項２０】
請求項１９において、前記１つ又はそれ以上の実行ユニットのうちの少なくとも１つがマルチサイクル実行レイテンシーで計算命令を実行することを特徴とするプロセッサ。
【請求項２１】
請求項１９において、前記データユニットが、パイプライン型の態様で拡張ユニット内における計算命令の実行を開始するために前記命令バス上に命令を発行することを特徴とするプロセッサ。
【請求項２２】
請求項１９において、前記１つ又はそれ以上の実行ユニットのうちの１つにおける実行パイプの各々が異なる実行レイテンシーで異なる計算命令を実行し、前記異なる計算命令の発行が第一命令の結果が爾後の命令からの結果により先行されることを防止すべく制御されることを特徴とするプロセッサ。
【請求項２３】
請求項１９において、更に、
前記１つ又はそれ以上の拡張ユニットの各々と前記結果バスとの間におけるＦＩＦＯ結果キュー、
を有しており、各ＦＩＦＯ結果キューが夫々の拡張ユニット内の全てのパイプライン型命令に対する結果を受付けるのに充分な深さを有しており、拡張ユニットがプロセッサコアとは独立的に動作することを可能としていることを特徴とするプロセッサ。
【請求項２４】
請求項１９において、更に、
前記拡張ユニットのうちの１つの中に内部レジスタ、
を有しており、前記データユニットが前記内部レジスタをアップデートするために前記命令バス上に拡張演算命令を発行することを特徴とするプロセッサ。
【請求項２５】
請求項１９において、前記データユニットが、更に、実行ユニット内において前に開始された演算の結果を保存し且つ回復するためのメカニズムを有していることを特徴とするプロセッサ。
【請求項２６】
請求項１９において、前記プロセッサに対する設計が、前記プロセッサの設計を再生することなしに実行ユニットを付加、除去又は変更するために修正することが可能であることを特徴とするプロセッサ。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、大略、プロセッサ命令セットアーキテクチャ（ＩＳＡ）に関するものであって、更に詳細には、プロセッサ命令セットアーキテクチャを拡張又は変更する技術に関するものである。
【背景技術】

【０００２】
プロセッサ命令セット条件は、通常、アプリケーションドメインの変化と共に展開する。特に、デジタル信号処理アプリケーションは、特別の命令でより良く実施するようにさせることが可能である。然しながら、このような性能の改善を達成するのに必要とされる特定の命令は、典型的に、時間と共に変化する。
【０００３】
新たなアプリケーションドメインの命令セット条件を受付けるための現在の方法は、命令セットを膨張させ且つ益々複雑なプロセッサ設計とさせる命令セットを拡張すること、又は最適状態には及ばない（遅いコア）ベースライン性能を有するプロセッサコアから調整したプロセッサの再生のいずれかを包含している。迅速な再生のための現在のアプローチは、既存の実行パイプライン内に適合するのに充分に簡単である新たな命令を必要とし、それにより、レイテンシー（待ち時間）及び資源の利用の観点において新たな命令の複雑性を制限する。従って、現在の方法は複雑なデータパス（更なるパイプライン段）、付加的な内部状態（例えば、プライベートレジスタファイル）又はオプションとしてのパイプライン型操作を伴う多数サイクル実行レイテンシーを必要とする新たな命令の必要性を受付けることは不可能である。
【発明の開示】
【発明が解決しようとする課題】
【０００４】
従って、命令スケジューリングのために使用され且つコンパイラーに対して既知のレイテンシーで、複雑なデータパス、付加的な内部状態、及びオプションとしてのパイプライン型演算を包含する場合のある新たな単一又はマルチサイクル命令をサポートしながら、固有の命令に対して良好な性能を与えるアーキテクチャに対する必要性が存在している。
【０００５】
本発明は、以上の点に鑑みなされたものであって、上述した如き従来技術の欠点を解消し、プロセッサ命令セットアーキテクチャを拡張又は変更するための改善した技術を提供することを目的とする。
【課題を解決するための手段】
【０００６】
上述した従来技術の欠点を解消するために、本発明の主要な目的とするところは、プロセッサアーキテクチャにおいて使用するための電気的インターフェースを提供することであり、該電気的インターフェースはプロセッサコアを計算（ユーザが定義した）命令を実行する１つ又はそれ以上のコプロセッサ（コア拡張）ユニットへ結合させるものであって、拡張ユニットへ命令及び命令のオペランドを供給するため及び拡張ユニットから結果を検索するために分割命令トランズアクションモデルを使用している。汎用コア拡張命令は、プロセッサアーキテクチャの再生なしで、拡張ユニットへデータへ送り及び／又は拡張ユニットから結果を検索するための能力と共に、新たな計算命令を導入することを可能とさせる。マルチ拡張ユニット及び／又は各拡張ユニット内のマルチ実行パイプライン、実行ユニット間又は実行ユニット内のマルチサイクル拡張レイテンシー及び異なる実行レイテンシー、実行ユニット命令述語、プロセッサコアストール及びインタラプトに関する結果保存／回復に対するサポートが包含されている。
【発明を実施するための最良の形態】
【０００７】
以下に説明する図１乃至５及び本明細書における本発明の原理を説明するために使用する種々の実施例は単に例示的なものであって本発明の技術的範囲を制限するような態様で解釈されるべきものではない。当業者によって理解されるように、本発明の原理は種々の態様で実現することが可能である。
【０００８】
図１は本発明の１実施例に基づく分割トランズアクションモデルコプロセッサ拡張インターフェースを包含するプロセッサアーキテクチャを示している。再生を介してのアプリ

ケーションドメインの変化に対する命令セットを拡張する代わりに、本発明は、複数個のコプロセッサをサポートする良好に定義されたコプロセッサインターフェースを具備するカスタム化可能なコプロセッサ拡張アーキテクチャを包含する高性能デジタル信号プロセッサに対する柔軟性のある代替物を提供している。このアーキテクチャは、ＳＯＣ（システム・オン・チップ）設計に対するコアハードマクロをコプロセッサと容易に統合することを可能としている。

【０００９】

プロセッサアーキテクチャ１００はコア１０１を有しており、コア１０１は、プログラムメモリ（明示的には示していない）を具備するプログラムメモリ制御器ＰＭＣと、制御ユニットＣＵと、データレジスタファイルＤＲＦと共に作業を行う２データユニットパイプラインＤＵ－０及びＤＵ１と、データメモリバンクＸ及びＹとを包含している。各データユニットは、ＤＲＦからソースオペランドＲｎ，Ｒｐが検索（フェッチ）されるデータオペランドフェッチパイプラインステージ（ＤＯＦ）、標準のコア命令を実行する２個の実行パイプラインステージＥＸ１，ＥＸ２、命令結果ＲｍがＤＲＦ内に書き込まれるライトバック（ＷＢ）ステージを包含している。データユニットパイプラインＤＵ－０へコプロセッサ拡張電気的インターフェース１０２が結合されており、該インターフェースは５ビット拡張ストローブバスｘ－ｓｔｒｏｂｅ、２９ビット拡張命令バスｘ－ｉｎｓｔ、３２ビットコプロセッサ拡張第一オペランドバスｘ－ｏｐｅａ、３２ビットコプロセッサ拡張第二オペランドバスｘ－ｏｐｂ、４０ビットコプロセッサ拡張結果バスｘ－ｒｅｓｕｌｔを有している。

【００１０】

拡張インターフェース１０２は、コア１０１を１つ又はそれ以上のコプロセッサ拡張ユニット１０３ａ－１０３ｄへ結合しており、各拡張ユニットは、オプションとして、先入先出（ＦＩＦＯ）出力バッファ１０４ａ－１０４ｄを包含している。例示的実施例においては、コプロセッサ拡張ユニット１０３ａ－１０３ｄは、２ビット識別子を使用してアドレスされ、そのことは最大で４個のコプロセッサ拡張ユニットをサポートすることを暗示している。然しながら、本発明は、より少ない数又はより多い数のコプロセッサ拡張ユニットへ容易に制限又は拡張させることが可能である。

【００１１】

ｘ－ｓｔｒｏｂｅ信号は、コア１０１の異なるパイプライン段からの情報を有しており、該情報は、１ビットコプロセッサ拡張命令有効ｘ－ｖａｌｉｄ信号と、該有効なコプロセッサ拡張命令がスケジュールされ且つ実行段１（ＥＸ１）パイプ段内にあることを表わす１ビット信号ｘ－ｓｃｈｅｄｕｌｅと、２ビットコプロセッサ拡張命令述語値信号ｘ－ｇｕａｒｄと、該コプロセッサ拡張命令が結果を発生するものと予測されることを表わす１ビットライトバック信号ｘ－ｗｂとを包含している。

【００１２】

以下の表１は、コプロセッサ拡張インターフェース１０２内の信号を要約したものであり、尚「タイプ」は信号がプロセッサコアに対する入力であるか又は出力であるかを意味している。

【００１３】

【表１】

信号名	幅	タイプ	パイプ段	記述
x_valid	1	出力	DOF	コプロセッサ命令は有効
x_inst	28:0	出力	DOF	コプロセッサ拡張命令 [31:7|3:0]
x_schedule	1	出力	EX1	サイクルＤＯＦにおいて有効であったコプロセッサ命令がＤＵパイプＥＸ１段へエンター
x_opa	31:0	出力	EX1	第１コプロセッサオペランド
x_opb	31:0	出力	EX1	第２コプロセッサオペランド
x_wb	1	出力	EX1	ＸＬＤコンポーネントを有するコプロセッサ命令がＥＸ１段内に有る
x_except	1	入力	EX1	コプロセッサにより駆動される例外ステータス
x_guard	2	出力	EX2	述語Ｇｘ、（Ｇｘ１６）状態
x_result	39:0	入力	EX2	コプロセッサ結果バス

【００１４】
データオペランドフェッチ段においてコプロセッサ命令を発行した後に、プロセッサコア１０１は潜在的にストールする場合があり、その場合に、ｘ－ｓｃｈｅｄｕｌｅ信号は以下のサイクルにおいて不活性状態である。従って、ｘ＿ｓｃｈｅｄｕｌｅが活性状態となるまで、コプロセッサ拡張は前のサイクルにおいて発行されたｘ＿ｉｎｓｔを継続してデコードすべきである。

【００１５】
図２Ａ乃至２Ｃは、本発明の種々の実施例に基づいて、パイプライン型及び非パイプライン型動作での及び分割トランズアクションコプロセッサ拡張に対する結果キューが有る場合及び無い場合の分割トランズアクション実行の動作を例示したタイミング線図である。最も簡単な形態においては、コプロセッサ拡張はＬサイクルのレイテンシー及びパイプライン型の態様で使用されるＴサイクルの処理能力を具備する応用特定機能を包含している。コプロセッサ拡張命令は分割トランズアクションモデルを追従し、その場合にコプロセッサ内の動作を実行し且つ該コプロセッサから動作結果を検索するために最小で２個のコプロセッサ拡張命令が必要とされる。対比的に、単一トランズアクションモデルは、動作を開始させ且つ内部コアデスティネーション（宛先）を提供するために単一のコプロセッサ命令を必要とするに過ぎない。このモデルは、又、内部状態として又はコプロセッサレジスタファイル内のいずれかにおいて、動作結果がコプロセッサ内部に拡張される場合において本発明において使用することも可能である。

【００１６】
分割トランズアクションモデルの場合には、第一コア拡張命令がオペランド及び演算（操作）をプロセッサへ供給し且つ第二コア拡張命令が前の命令によって発生された結果を検索する（及び、オプションとして、後述するように関連するオペランドを有する別の演算を送る）。この分割トランズアクションモデルはコプロセッサ拡張機能のユーザが定義したレイテンシーを可能とし、且つ最小でＬ個のサイクルを採る。

【００１７】
ユーザが定義した演算（操作）及びオペランドをコプロセッサ拡張ユニット１０３ａ－１０３ｄへ送る第一コア拡張命令は、プロセッサコア１０１に対する命令セットアーキテクチャ（ＩＳＡ）において以下の如くに表わされる。

【００１８】
【表１ａ】

```
XSD    Rn, Rp, #xspc11        // XSD (Store Data to eXtension)
                              // Data to store: registers Rn, Rp of DRF
                              // #xspc11 specifies 11 user-defined bits
```

【００１９】
尚、ｘｓｐｃは（ユーザが定義した）コプロセッサ演算即ち操作を特定する拡張特定命令フィールドである。演算即ち操作を送り且つ結果を検索する第二コア拡張命令は以下の如

くにコアＩＳＡ内において表わされる。

【００２０】

【表１ｂ】

```
XLD    Rm, #xspc15        // XLD (Load Data from eXtension)
                          // Retrieved data is written to register Rm of DRF
                          // #xspc15 specifies 15 user-defined bits
```

【００２１】

図２Ａのパイプライン線図は非パイプライン型演算（操作）に対しレイテンシーＬ＝３の場合の分割トランズアクションを例示しており、ロードデータ命令ＸＬＤの最適なスケジューリングはＸＳＤ命令の後３個のサイクルである。図２Ａの上側部分２０１ａはプロセッサコア１０１内の動作を例示しており、一方下側（イタリック）部分２０１ｂはコプロセッサ拡張インターフェース１０２内の対応する操作を例示している。図２Ａに示した命令シーケンスに対する対応するアッセンブリコードを以下の表２に示してある。

【００２２】

【表２】

PC [31:0] (GP32モード)	DU命令	
	スロット1	スロット0
0x100	DU-OP	Gx? XSD Rn, Rp, #xspc11
0x108	DU-OP	DU-OP
0x110	DU-OP	DU-OP
0x118	DU-OP	Gx? XLD Rm, #xspc15
0x120	DU-OP	Gy? XSD Rn, Rp, #xspc11
0x128	DU-OP	DU-OP
0x130	DU-OP	DU-OP
0x138	DU-OP	Gy? XLD Rm, #xspc15

【００２３】

尚、プログラムカウンタ（ＰＣ）値は、プロセッサコア１０１が包括的３２ビット汎用命令セット（ＧＰ３２）を使用するモードで実行しているものと仮定する。図２Ａの線図は、最適なスケジューリングの場合に、ロードデータ命令ＸＬＤが、コプロセッサ拡張ユニットのレイテンシーに等しい格納データ命令ＸＳＤより３サイクル後であることを示している。

【００２４】

明らかに、コプロセッサ拡張がパイプライン型の態様で操作される場合には、サイクル毎にコプロセッサ拡張ユニットから１つの結果を発生することが可能である。然しながら、コプロセッサ拡張アーキテクチャの定義は、この実施例においては、サイクル毎に１つのコプロセッサ命令のみを発行することに制限されている。その拘束条件が与えられると、ＸＬＤが続くＸＳＤを包含するコードシーケンスに対するパイプライン型態様でコプロセッサ拡張を動作させることは不可能である。従って、フォーム送給データ及びロードデータＸＳＤＬＤの命令が必要とされ、そのことはオペレーション即ち演算及びオペランドを拡張へ送り且つ同時的に拡張から結果をロードすることを暗示している。図２Ｂのパイプライン線図は、パイプライン型の態様で動作されるレイテンシーＬ＝３に対する分割トランズアクションを例示しており、対応するアッセンブリコードを以下の表３に示してある。

【００２５】

【表３】

PC [31:0] (GP32モード)	DU命令	
	スロット1	スロット0
0x100	DU-OP	XSD Rn, Rp, #xspc11 (1)
0x108	DU-OP	XSD Rn, Rp, #xspc11 (2)
0x110	DU-OP	XSD Rn, Rp, #xspc11 (3)
0x118	DU-OP	XSDLD Rm, Rn, Rp #xspc7 (4)(1)
0x120	DU-OP	XSDLD Rm, Rn, Rp #xspc7 (5)(2)
0x128	DU-OP	XLD Rm, #xspc15 (3)
0x130	DU-OP	XLD Rm, #xspc15 (4)
0x138	DU-OP	XLD Rm, #xspc15 (5)

【００２６】

コプロセッサ拡張命令は括弧内に識別子でタグが付けられており、各データ送給命令ＸＳＤとマッチするロードデータ命令ＸＬＤを表わしている。データ送給及びデータロード命令ＸＳＤＬＤの場合には、括弧内の一対の識別子が使用され、最初のものはデータ送給部分に対応しており且つ２番目のものはその結果がデータロード部分を検索する命令に対応している（Ｒｍ：デスティネーションオペランドレジスタ、Ｒｎ，Ｒｐ：ソースオペランドレジスタ）。

【００２７】

図２Ｂはコプロセッサ拡張演算のパイプラインを例示しており、最適に実施される場合に、その結果が、データロード命令ＸＬＤ又は結合された命令ＸＳＤＬＤのデータロードコンポーネントのいずれかを使用して検索されることを仮定している。然しながら、プロセッサコア１０１がストールする（コプロセッサはコアのストールに気づいていない）か、又はインタラプトに起因してスレッドをスイッチするかのいずれかであり、従って最適な態様で結果をピックアップすることが不可能である場合には、その結果に対する一時的な格納部が存在しない場合にはコプロセッサの結果を失うことの明確な可能性が存在している。

【００２８】

この問題を解決するために、スピルオーバーを保持するために各コプロセッサ拡張ユニット１０３ａ－１０３ｄに対してＦＩＦＯ結果キュー（ＲＱ）構造１０４ａ－１０４ｄが必要とされ、コアストール条件により影響されることなしにプロセッサコア１０１とは独立的にコプロセッサ拡張パイプラインをランさせることを可能とする（何故ならば、プロセッサコア１０１がストールした場合にコプロセッサ拡張ユニット１０３ａ－１０３ｄはストールすることがないからである）。結果キューが存在することは、データロード命令（即ち結合した命令のデータロードコンポーネント）が、常に、ＦＩＦＯ構造の底部から結果を検索することを意味している。

【００２９】

各ＦＩＦＯ結果キュー１０４ａ－１０４ｄは、コプロセッサレイテンシーＬ及びコプロセッサスループット１／Ｔ（Ｔサイクル毎に１個の結果）の両方の関数である少なくとも深さＤを有しており、それは以下の式によって与えられる。

【００３０】

Ｄ＝Ｃｅｉｌｉｎｇ（Ｌ／Ｔ）

ＦＩＦＯ結果キュー１０４ａ－１０４ｄ内のエキストラなエントリは命令スケジューリングを容易なものとさせることが可能である。

【００３１】

例示的実施例においては、最大で４個のコプロセッサ拡張ユニット１０３ａ－１０３ｄをプロセッサコア１０１へ接続させることが可能であり、その各々はＦＩＦＯ結果キュー１０４ａ－１０４ｄを有するべきである。レイテンシー及びスループット（処理能力）は個別的なコプロセッサ拡張ユニット１０３ａ－１０３ｄのパラメータであり且つ異なるコプロセッサに対して異なるものとすることが可能であるので、夫々のＦＩＦＯ結果キュー１０４ａ－１０４ｄの深さも異なる場合がある。

【００３２】

コプロセッサ拡張ユニット１０３ａ－１０３ｄの内部的状態は、命令述語（データユニット実行段２ＤＵ－ＥＸ２期間中に到着する述語ステータス）が真である場合にのみアップデートされるものであるが、常にその述語状態とは独立的にコプロセッサ結果が夫々のＦＩＦＯ結果キュー内に書込まれ且つＦＩＦＯ有効カウンタがアップデートされる。ＸＬＤ又は対応する命令コンポーネントは、常に、ＦＩＦＯ結果キューから読取られ且つその述語ステータスに関してＦＩＦＯステータスをアップデートし、そのことはＤＵ－ＥＸ２段における述語の遅れた到着に起因するタイミング拘束条件を除去する。

【００３３】

図２ＣはＦＩＦＯ結果キュー無しで実現した単純なコプロセッサ拡張のパイプラインを例示しており、対応するアッセンブリコードは以下の表４に示してある。

【００３４】

【表４】

PC [31:0] (GP32モード)	DU命令		
	スロット1	スロット0	コメント
0x100	DU-OP	XSD CR0, Rn, Rp, xfn (1)	CR0=xfn(Rn, Rp)
0x108	DU-OP	XSD CR1, Rn, Rp, xfn (2)	CR1=xfn(Rn, Rp)
0x110	DU-OP	XLD Rm, CR0, xfn_nop (3)	Rm=CR0
0x118	DU-OP	XLD Rm, CR1, xfn_nop (4)	Rm=CR1

【００３５】

ＦＩＦＯ結果キュー無しでのコプロセッサ拡張ユニット２０３ａ－２０３ｄは、コプロセッサ拡張結果バスｘ－ｒｅｓｕｌｔへ結合された結果レジスタを包含すべきである。例示的な実施例においては、ＣＲ０及びＣＲ１は、各々、コプロセッサ特定レジスタファイルに対する内部状態か又はレジスタ識別子のいずれかを表わすものと仮定されており、一方ｘｆｎは該命令の＃ｘｓｐｃフィールドを使用して特定したコプロセッサ機能を表わしている。

【００３６】

パイプラインタイミングに関連して、図２Ｃにおける臨界的なコプロセッサパイプライン段は「内部状態読取及び拡張のＲｅｓｕｌｔ＿Ｒｅｇｉｓｔｅｒ書込」である。

【００３７】

ＦＩＦＯ結果キュー無しでの実現は、レンテンシーの短いコプロセッサソリューションに対して適したものであるが、述語はＤＵ－ＥＸ２段においてのみ使用可能であるという事実に対して注意深い考慮を払わねばならない。ＦＩＦＯ結果キューを有するコプロセッサ拡張は、コンパイラーがループパイプラインを使用する場合に効率的である。然しながら、あるコプロセッサソリューションはＦＩＦＯ結果キューを保存し且つ回復する付加された複雑性を正当化させるものでない場合がある。

【００３８】

ＸＳＤ，ＸＬＤ，ＸＳＤＬＤコプロセッサ拡張命令に加えて、そこからオペランドではなくオペレーション即ち演算のみがコプロセッサへ送られ且つ結果が検索されることのない付加的な命令が望ましく、

ＸＯＰ　＃ｘｓｐｃ１９

尚、該命令は１９個のユーザが定義したビットを包含している。このフォーマットは、コプロセッサ拡張ユニットが、直接的又は内部状態のいずれかを使用して内部演算を実施し且つ内部状態をアップデートすることを可能とする（然しながら、内部状態のアップデートはＸＯＰに制限されるものではない）。

【００３９】

上述したアーキテクチャは、エンコーディングにおいて同一でありプライマリー演算コード（ｏｐｃｏｄｅ）記述においてのみ異なるＸＳＤ命令の４つのバージョンと共にエスティーマイクロエレクトロニクス、インコーポレイテッドから入手可能なＳＴ１２０プロセッサで実現することが可能である。同様に、ＸＬＤ命令の２つのバージョン、ＸＳＤＬＤ命令の４つのバージョン、及びＸＯＰ命令の２つのバージョンを実現することが可能である。該命令のビット毎の構成を以下の表５に示してある。

【００４０】
【表５】

<table>
<tr><td>[31:28]</td><td>[27:24]</td><td>[23:22]</td><td>[21]</td><td>[20:17]</td><td>[16:11]</td><td>[10:7]</td><td>[6:0]</td></tr>
<tr><td rowspan="2">Ga</td><td rowspan="2">xspc
[10:7]</td><td rowspan="2">Ext#[1:0]</td><td rowspan="2">xspc
[6]</td><td>Rb</td><td rowspan="2">xspc
[5:0]</td><td>Rc</td><td rowspan="2">XSD
(4 opcodes)</td></tr>
<tr><td>ソース</td><td>ソース</td></tr>
<tr><td rowspan="2">Ga</td><td>Ra</td><td rowspan="2">Ext#[1:0]</td><td rowspan="2">xspc
[6]</td><td>Rb</td><td rowspan="2">xspc
[5:0]</td><td>Rc</td><td rowspan="2">XSDLD
(4 opcodes)</td></tr>
<tr><td>Dest.</td><td>ソース</td><td>ソース</td></tr>
<tr><td rowspan="2">Ga</td><td>Ra</td><td rowspan="2">Ext#[1:0]</td><td colspan="4" rowspan="2">Xpsc
[14:0]</td><td rowspan="2">XLD
(2 opcodes)</td></tr>
<tr><td>Dest.</td></tr>
<tr><td>Ga</td><td>xspc
[10:7]</td><td>Ext#[1:0]</td><td colspan="4">Xpsc
[14:0]</td><td>XOP
(2 opcodes)</td></tr>
</table>

【００４１】
この命令セットエンコーディングは、最大で４つのコプロセッサ拡張ユニットのアドレッシングを可能とする２ビットフィールドＥｘｔ＃［１：０］を包含している。
【００４２】
ＳＴ１２０アーキテクチャに準拠したプロセッサ内のＧＰ３２命令に対するセマンティックス及びシンタックスを以下の表６に示しており、その場合に、ｘｆｎはコプロセッサ拡張機能を意味しており、「Ｘａ」はＦＩＦＯ結果キューを意味しており、且つ「ａ」は２ビットコプロセッサ拡張番号である。
【００４３】
【表６】

ニーモニックシンタックス	GP32セマンティクス	xfnフィールド幅
Gx? XSD<I>a, Rn, Rp, #xspc11	Gx? [Xa=]xfn (Rn,Rp)	xspc[10:0]
Gx? XSDLDa, Rm, Rn, Rp, #xspc7	Gx? Rm=Xa||xfn(Rn,Rp)	xspc[6:0]
Gx? XLD<I>a, Rm, #xspc15	Gx? Rm=Xa, xfn	xspc[14:0]
Gx? XOP<I>a, #xspc19	Gx? [Xa=]xfn	xspc[18:0]

【００４４】
表６の２番目の欄における方程式の左側において、「Ｘａ」はＦＩＦＯ結果キュー内への書込みを意味しており、一方、「Ｘａ」はＦＩＦＯ結果キューの読出しを意味している。更に、＜Ｉ＞は命令のフォームを表わしている（例えば、ＸＳＤ１，ＸＳＤ２，ＸＳＤ３又はＸＳＤ４）。全てのコプロセッサ命令は拡張特定フィールドｘｓｐｃを有しており、その幅はビット単位で添え書きした数字で表わしている。
【００４５】
全てのコプロセッサ命令は述語指定されている。コプロセッサ拡張は２つのタイプ、即ち何等の内部状態又はレジスタファイルなしでのコプロセッサ、即ち結果キュー（ＲＱ）を有する純粋な実行ユニット（タイプＡ）、及び内部状態及び／又はレジスタファイルを有するコプロセッサ、即ち実行ユニット及び結果キュー（タイプＢ）のうちの１つとすることが可能である。両方のタイプのコプロセッサに対する述語の利用に関してのＳＴ１２０コプロセッサ命令セマンティックスを以下の表７に示してある。
【００４６】

【表７】

命令	コプロセッサセマンティクス	コメント
Gx? XSD Rn, Rp, #xspc11	[RQ=]xfn(Rn,Rp)	ＲＱへの書込は述語指定されておらず
	Gx? [Cn=]xfn(Rn,Rp)	内部レジスタの場合、Ｇｘが真である場合にのみＣｎが書き込まれる
Gx? XLD Rm, #xspc15	x_result=RQ	ＲＱは常に読み取られる
	Gx? Rm=x_result	Ｇｘが真である場合にＲｍが書き込まれる
	Gx? xfn()	ガードした内部演算
Gx? XSDLD Rm, Rn, Rp, #xspc7	[RQ=]xfn(Rn,Rp)	ＲＱへの書込は述語指定されておらず
	Gx? [Cn=] xfn(Rn,Rp)	内部レジスタの場合、Ｇｘが真である場合にのみＣｎが書き込まれる
	x_result=RQ	ＲＱは常に読み取られる
	Gx? Rm=x_result	Ｇｘが真である場合にＲｍは書き込まれる
Gx? XOP #xspc19	[RQ=]xfn()	ＲＱへの書込は述語指定されておらず
	Gx? [Cn=]xfn()	内部レジスタの場合、Ｇｘが真である場合にのみＣｎが書き込まれる

【００４７】

「ＲＱ」はＦＩＦＯ結果キューを意味しており、「Ｃｎ」はコプロセッサ内部状態（例えば、内部レジスタ）を意味しており、且つ「ｘ＿ｒｅｓｕｌｔ」はコアへ接続するコプロセッサ結果バスを意味している。括弧（例えば、「｛ＲＱ＝｝」）は、コプロセッサソリューションに基づいて書込（結果キュー又は内部レジスタＣｎに対して）がオプションであることを表わしている。内部レジスタＣｎを参照する代替物は、コプロセッサが内部状態（タイプＢ）を有している場合にのみ存在する。

【００４８】

タイプＡコプロセッサ拡張（内部状態なし）の場合には、述語の利用は既存のＦＴ１２０のものと同様であり、ＸＳＤ演算又はＸＳＤＬＤ又はＸＯＰ命令のコンポーネント部分が投機的に実行され且つＤＲＦへの書込を行うＸＬＤコンポーネントの部分のみが述語指定される。注意することが重要であるが、結果キューへの書込及び結果キューからの読取は述語指定されるものではなく、タイプＡコプロセッサ拡張はコアからの述語を必要とするものではない。

【００４９】

タイプＢコプロセッサ拡張（内部状態有り）の場合には、内部状態アップデート及び純粋な内部コプロセッサ演算が述語指定される。ＸＳＤ及びＸＬＤコンポーネントは、両方のコンポーネントが同一の述語を使用する場合にのみ１つのＸＳＤＬＤへ合体することが可能である（そのことは、ＸＯＰ及びＸＬＤの場合にも当て嵌まる）。１例として、典型的な場合は「述語指定されない」コード（両方のコンポーネントに対してＧｘ＝Ｇ１５）又はＧｘにより述語指定されるコードブロック（Ｇｘを使用して「変換される場合」に完全な組の命令）である。ＸＳＤ及びＸＬＤコンポーネントが同一の述語を保持することが不可能である場合には、ＸＳＤＬＤ命令は２つの（ＸＳＤ及びＸＬＤ）命令も分解され、それらの夫々の独特の述語が以下の表８に示した例において例示してある。

【００５０】

【表８】

PC (GP32)	スロット1	スロットO	命令コンポーネント	
			XSD	XLD
0x100	DU-OP	Gx? XSD Rn, Rp, #xspc11	Gx? XSD	
0x108	DU-OP	Gy? XSD Rn, Rp, #xspc11	Gy? XSD	
0x110	DU-OP	Gx? XSDLD Rm, Rn, Rp, #xspc7	Gx? XSD	Gx? XLD
0x118	DU-OP	Gy? XLD Rm, #xspc15		Gy? XLD
0x120	DU-OP	Gx? XLD Rm, #xspc15		Gx? XLD

【００５１】
表８の例においては、ＰＣ＝０ｘ１００，ＰＣ＝０ｘ１１０，ＰＣ＝０ｘ１２０における命令がタイプＢコプロセッサに対するガード条件に関して述語指定され、一方ＰＣ＝０ｘ１０８及びＰＣ＝０ｘ１１８における命令がガード条件Ｇｙに関して述語指定される。
【００５２】
両方のタイプＡ及びタイプＢのコプロセッサ拡張の場合に、ＦＩＦＯ結果キューが、命令の述語ステータスに拘わらずに書込まれ且つアップデートされ、且つ本発明の述語ステータスに拘わらずにＸＬＤコンポーネントに関し、ＦＩＦＯ結果キューが読み取られ且つＦＩＦＯ結果キューポインターがアップデートされる。然しながら、タイプＢコプロセッサ拡張の場合、対応する先の命令の結果を検索する別の命令により後の時間において続くオペランド及び拡張特定オプコードを供給するコプロセッサ拡張は同一の述語を共用すべきである。
【００５３】
図３は本発明の１実施例に基づく分割トランズアクションコプロセッサ拡張ユニット内の非一様な命令レイテンシーを取扱う方法を例示している。コプロセッサ拡張ユニット１０３は、概念的に、各々が異なる実行レイテンシーを有する異なるタイプの命令を実行する幾つかの実行ユニット３０１，３０２を有することが可能である。最適にコプロセッサを使用し且つコンパイラー／オプティマイザーが施行する規則を定義するために、コプロセッサ命令の結果はプログラム順に検索されねばならないことを記憶すべきである。レイテンシーＬ１，Ｌ２を有する２つの連続した命令ｉ１，ｉ２がコプロセッサへ送られ（必ずしも連続したサイクルではない）、且つこれらの命令を実行する実行ユニット３０１，３０２が単一のポート３０３を介してＦＩＦＯ結果キュー１０４へ結果を書込むものと仮定すると、２つの場合が提供される。即ち、
（ａ）Ｌ２≧Ｌ１、これは（Ｌ２－Ｌ１）＋１サイクルに対し命令ｉ２により拡張結果キューがアップデートされることが無いことを暗示しており（本実施例においては、「＋１」は、最善である場合に、命令ｉ１及びｉ２が連続したサイクルで発行されるという事実に起因していることに注意すべきである）、命令は順番でアップデートされる結果キューに対し競合するものでないので許容可能な状況であり、且つ
（ｂ）Ｌ２＜Ｌ１、この場合には、結果キューに対し競合する拡張内において両方の実行ユニットと潜在的なコンフリクトが発生する場合があり、コアが命令ｉ２の発行をストールさせない限り、命令ｉ２が命令ｉ１の前に結果キューをアップデートしようとする可能性がある。
【００５４】
従って、Ｌ２＜Ｌ１の不定の場合を検知し且つ取扱うために特別のスケジューリング論理が必要である。そのスケジューラーは、同一のコプロセッサ拡張に対しての発行のために受取られる連続した（又は近づいて離れた）命令の夫々のレイテンシーのことを気がついておらねばならず、且つ、Ｌ２＜Ｌ１の場合には、２番目の命令ｉ２をコプロセッサ拡張へディスパッチする前にＬ１－Ｌ２サイクル待機すべきである。スケジューラーは、Ｌ２≧Ｌ１である場合には、命令ｉ２をディスパッチするために待機することは必要ではない。スケジューラーは、コプロセッサ拡張内の実行エンジンを最適に使用するためにこれらの規則を適用すべきである。
【００５５】
図４は本発明の１実施例に基づいた分割トランズアクションコプロセッサ拡張ユニット内の複数個のコプロセッサ拡張の結果バスを接続する１つの可能なメカニズムを例示している。上述した例示的実施例は、最大で４つの異なるコプロセッサ拡張ユニットを可能とし、潜在的に異なる命令レイテンシーを有している。各拡張ユニットがＦＩＦＯ結果キューを有している場合には、各拡張ユニットのスケジューリング規則を尊重することが必要である。
【００５６】
コプロセッサが集約的に使用される場合には、コプロセッサは、新たな計算を開始し且

つ新たな結果をサイクル毎にＤＲＦへ書込むことが可能である。その結果、インタラプトが発生すると、コプロセッサによって発生された結果はパイプラインのドレイン期間中にデータユニット内へ移動させることは不可能であり、且つその結果は一時的な格納部がない場合には失われることとなる。従って、ＦＩＦＯ結果キューがコプロセッサの最後のパイプ段とコアデータユニットの間に配置させてコプロセッサにより発生されたこれらの結果を保持する。プログラマーは、任意のサイクルにおいて、未決の結果の数がＦＩＦＯ結果キュー深さ以下であり、従ってインタラプト又はコアストール条件の場合においてＦＩＦＯ結果キューがオーバーフローすることがないことを確保せねばならない。

【００５７】

ＦＩＦＯ結果キューにおける有効なエントリの数はレジスタＦＩＦＯカウンタ（不図示）内に包含されており、ＦＩＦＯ結果キューに対する読取及び書込は以下の規則に準拠する。

【００５８】

※読取の場合、空のＦＩＦＯ結果キューは既にリターンされた最後の有効な結果をリターンし且つＦＩＦＯカウンタはゼロにおいて不変のまま止まる；

※読取の場合、空でないＦＩＦＯ結果キューは最も古いエントリをリターンし、次いでそのエントリを落とし且つＦＩＦＯカウンタをデクリメントする；

※読取及び同時的に書込まれる場合、空のＦＩＦＯ結果キューは前にリターンされた最後の有効な結果をリターンし、書込まれたデータで新たなエントリを作成し、且つＦＩＦＯカウンタをインクリメントする；

※読取及び同時的に書込まれる場合、空でないＦＩＦＯ結果キューは最も古い結果をリターンし、その結果を落とし、書込まれたデータで新たなエントリを作成し、且つＦＩＦＯカウンタを不変のままとさせる；

※読取及び同時的に書込まれる場合、満杯のＦＩＦＯ結果キューはＦＩＦＯ結果キューマイクロアーキテクチャにより又はＦＩＦＯ結果キュー理論的寸法により必要とされるより１つの多くのエントリを実現することにより保証される挙動で説明した如くに動作する；

※書込まれた場合、空のＦＩＦＯ結果キューは書込まれたデータで新たなエントリを作成し且つＦＩＦＯカウンタをインクリメントする（０から１へ）；

※書込まれた場合、空でないＦＩＦＯ結果キューは書込まれたデータで新たなエントリを作成し且つＦＩＦＯカウンタをインクリメントする（０から１へ）；

※書込まれた場合、満杯のＦＩＦＯ結果キューは未定義の挙動を有している。

【００５９】

インタラプトの発生時にＦＩＦＯ結果キューを保存し且つ回復することが可能であるためには、以下の表９により示されるように幾つかのコプロセッサ命令がそのタスクに対して専用のものとされねばならない。

【００６０】

【表９】

コプロセッサ命令	記述
XLD Rm, xfn_rdrqcnt	• ＦＩＦＯをDUレジスタRm内に移動し且つＦＩＦＯカウンタ値をＦＩＦＯ内にコピー • ＦＩＦカウンタが0である場合、インクリメントさせ；そうでない場合、不変のままである。
XLD Rm, xfn_rdrq	• ＦＩＦＯをDUレジスタRm内へ移動 • ＦＩＦＯカウンタが0である場合、不変のままであり：そうでない場合、デクリメントされる。
XOP xfn_rstfifo	• ＦＩＦＯカウンタをリセット
XSD Rn, Rp, xfn_wrrq	３２ビットＦＩＦＯ専用： • Ｒｎ[３１：０]を３２ビットＦＩＦＯ内に移動（Ｒｐ[３１：０]を喪失） • ＦＩＦＯカウンタは１だけインクリメントされる ４０ビットＦＩＦＯ専用： • Ｒｐ[２３：１６]及びＲｎ[３１：０]の連結を４０ビットＦＩＦＯ内に移動（Ｒｐ部分は8MSB及びＲｎ部分は３２ＬＳＢ） • ＦＩＦＯカウンタは１だけインクリメントされる
XSD Rn, xfn_wrrqcnt	• Ｒｎ[３１：０]を３２ビットＦＩＦＯカウンタ内に移動

【００６１】
注意すべきことであるが、ＦＩＦＯ結果キュー幅はこの例示的実施例においては３２又は４０ビットのいずれかとすることが可能である。
【００６２】
４つのエントリを有する３２ビット又は４０ビットのいずれかのＦＩＦＯ結果キューを仮定すると、ＦＩＦＯ保存ルーチンに対する擬似コードを以下に示してある。
【００６３】
【表９ａ】

```
G15? XLD R0, xfn_rdrqcnt;     // state 1 -> state 2
G15? XLD R1, xfn_rdrq;        // state 2 -> state 3
G15? XLD R2, xfn_rdrq;        // state 3 -> state 4
G15? XLD R3, xfn_rdrq;        // state 4 -> state 5
G15? XLD R4, xfn_rdrq;        // state 5 -> state 6
...                           // code saving R0 to R4 somewhere
```

【００６４】
４つのエントリを有する３２ビットＦＩＦＯ結果キューを仮定すると、ＦＩＦＯ回復ルーチンに対する擬似コードを以下に示してある。
【００６５】
【表９ｂ】

```
...                           // code which restores R0 to R4 from the
...                           // location used by the save routine
G15? XOP xfn_rstfifo;         // state 7 -> state 8
G15? XSD R0, R0, xfn_wrrq;    // state 8 -> state 9 (second R0 is dummy)
G15? XSD R1, R1, xfn_wrrq;    // state 9 -> state 10 (second R1 is dummy)
G15? XSD R2, R2, xfn_wrrq;    // state 10 -> state 11 (second R2 is dummy)
G15? XSD R3, R3, xfn_wrrq;    // state 11 -> state 12 (second R3 is dummy)
G15? XSD R4, xfn_wrrqcnt;     // state 12 -> state 13
```

【００６６】
４つのエントリを有する４０ビットＦＩＦＯ結果キューを仮定すると、ＦＩＦＯ回復ルーチンに対する擬似コードを以下に示してある。
【００６７】
【表９ｃ】

```
...                           // code which restores R0-R3, R10-R13 and R4
...                           // from the location used by the save routine
G15? XOP xfn_rstfifo;         // state 7 -> state 8
G15? XSD R0, R10, xfn_wrrq;   // state 8 -> state 9
G15? XSD R1, R11, xfn_wrrq;   // state 9 -> state 10
G15? XSD R2, R12, xfn_wrrq;   // state 10 -> state 11
G15? XSD R3, R13, xfn_wrrq;   // state 11 -> state 12
G15? XSD R4, xfn_wrrqcnt;     // state 12 -> state 13
```

【００６８】
注意すべきことであるが、回復ルーチンの実行の前にＦＩＦＯ結果キューが空である場合にはＦＩＦＯカウンタのクリア動作は必要ではなく、コプロセッサに対して発行された命令により発生された全ての結果を消費することは強く推奨されているソフトウエアのプラクティスであるので、そのことは通常の場合である。
【００６９】
図５は本発明の１実施例に基づくガロア（Ｇａｌｏｉｓ）乗算器分割トランズアクションコプロセッサ拡張ユニットの動作を例示したタイミング線図である。この例示的なガロア乗算器拡張（ＧＭＸ）ユニットは２つの３２ビットオペランドを採り且つ４つの８ビットフィールドに関してのガロア乗算を実施する３２ビット結果を発生する。この拡張ユニットはＥｘｔ＃フィールド内においてＧＭＸとして識別してあり、且つ完全にパイプライン型の２サイクル拡張実行エンジンと、それに続く結果キューへの単一サイクル書込とを包含するものと仮定されており、従ってコア上のプログラムにより見られるレイテンシーは３サイクルである。ソフトウエアパイプラインによって、１つの結果を発生し且つ以下の擬似コードシーケンスに示されるように、初期的なレイテンシーの後サイクル当たりＤ

ＲＦへ書込ませることが可能である。

【００７０】

【表９ｄ】

```
        .sliwmd
//Software pipelining: prologue

        10      G15? !d R0, @(P0 !+4)
        11      G15? !d R1, @(P1 !+4)
        12      G15? xsd gmx, R0, R1, gmpy    // gmx: Ext#   gmpy: Extension Specific
        13      nop

        14      G15? !d R0, @(P0 !+4)
        15      G15? !d R1, @(P1 !+4)
        16      G15? xsd gmx, R0, R1, gmpy
        17      nop

        18      G15? !d R0, @(P0 !+4)
        19      G15? !d R1, @(P1 !+4)
        20      G15? xsd gmx, R0, R1, gmpy
        21      nop
// Loop Starts
        22      G15? !d R0, @(P0 !+4)
        23      G15? !d R1, @(P1 !+4)
        24      G15? xsdld gmx, R2, R0, R1, gmpy
        25      nop

        26      G15? sdw @(P2 !+4), R2
        27      nop
        28      nop
        29      nop
// Loop Ends
// Epilogue

        30      G15? xld gmx, R2
        31      nop
        32      G15? sdw @(P2 !+4), R2
        33      nop

        34      G15? xld gmx, R2
        35      nop
        36      G15? sdw @(P2 !+4), R2
        37      nop

        38      G15? xld gmx, R2
        39      nop
        40      G15? sdw @(P2 !+4), R2
        41      nop
```

【００７１】

基本的なモデルは、パイプラインを準備するために、メモリからガロアフィールドを読取り且つ２つの３２ビットデータレジスタファイル（ＤＲＦ）オペランドをＸＳＤを使用して拡張ユニットへ書込むことである。パイプラインがセットアップされると、ＸＳＤＬＤを使用して結果を検索し且つ次の２つの３２ビットＤＲＦオペランドを供給する。注意すべきことであるが、命令２４において、１つのオペレーション（演算）及び２つのオペランドが供給され且つ単一の命令を使用して結果が検索される。

【００７２】

図５はオペランドがメモリから検索されねばならず且つ拡張命令の結果がメモリ内に格納されねばならない場合の拡張のオペレーションを例示している。命令内の数字は上の擬似コードにおける命令に対応している。図示例においては、結果が１つ置きのサイクル毎に検索され、且つコアはサイクル毎に２つのメモリアドレスユニット（ＡＵ）動作を許容するに過ぎず、一方各ループ繰返しは３つのメモリアクセス（２つのロードと１つのストア）を必要とする。然しながら、コプロセッサインターフェースはサイクル毎に結果を発生する能力を有している。ＦＩＦＯ結果キューは結果をコアへリターンする前に結果を保持するために３の深さであることを必要とし、そのことは、インタラプトがこのループ内において又はストール条件下において発生する場合に重要なものとなる。インタラプトハンドラーは、別のスレッドがこのコプロセッサの使用を必要とする場合に、３エントリ深さのＦＩＦＯからの結果を検索し、格納し且つ回復することが可能であるべきである。

【００７３】
本発明においては、分割トランズアクション拡張ユニットの使用が、高性能デジタル信号プロセッサをカスタム化するための柔軟性のある代替物を提供しており、アーキテクチャの完全な再設計なしで特別の命令を有する実行ユニットを付加することを可能としている。このコプロセッサアーキテクチャは複数個のコプロセッサをサポートする良好に定義されたインターフェースでカスタム化することが可能であり、システム・オン・チップ（ＳＯＣ）プロセッササブシステム上のコアハードマクロとコプロセッサとの容易な統合を可能としている。このコプロセッサアーキテクチャは、別のデータユニットのように見えるコアデータユニットへ緊密に結合されている分割命令トランズアクションモデルに基づいている。本発明のアーキテクチャはマルチサイクルレイテンシー拡張実行を受付け且つ分離されたアーキテクチャを完全に使用する。関連する保存／回復命令フォームを有するスケーラブルなＦＩＦＯ結果キューが、効率的なパイプライン化及びコプロセッサ演算の静的スケジューリングをサポートするために定義され、且つ付加的な内部状態を任意に付加することが可能である。

【００７４】
本発明はマルチサイクル実行動作を効果的にパイプライン化するためにマルチサイクルレイテンシー実行拡張及び汎用コプロセッサ命令をサポートしている。スケーラブルＦＩＦＯ結果キューはレジスタ圧力を取除き且つコアにおけるストールを受付け、コプロセッサ拡張パイプラインがコアに関して独立的に動作することを可能とする。命令述語がコプロセッサ拡張に対して使用可能である。

【００７５】
例示的実施例のアーキテクチャは、最大で４個のコプロセッサ拡張ユニット及びコプロセッサからの結果バスに対する少なくとも２つのタイプの接続（ＯＲ論理又はトライステート）をサポートしている。コプロセッサ拡張及びマルチサイクルにおける非一様なレイテンシーがサポートされる。

【００７６】
以上、本発明の具体的実施の態様について詳細に説明したが、本発明は、これら具体例にのみ制限されるべきものではなく、本発明の技術的範囲を逸脱することなしに種々の変形が可能であることは勿論である。

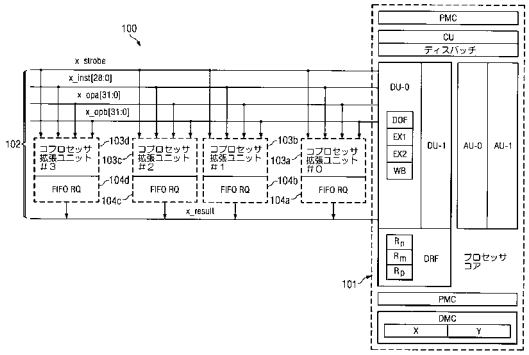
【図面の簡単な説明】

【００７７】
【図１】本発明の１実施例に基づく分割トランズアクションモデルコプロセッサ拡張インターフェースを包含するプロセッサアーキテクチャを示した概略図。
【図２Ａ】本発明の１実施例に基づいて分割トランズアクションコプロセッサ拡張に対する結果キューの有る場合及び無い場合且つパイプライン型及び非パイプライン型動作を有する分割トランズアクション実行の動作を示したタイミング線図。
【図２Ｂ】本発明の１実施例に基づいて分割トランズアクションコプロセッサ拡張に対する結果キューの有る場合及び無い場合且つパイプライン型及び非パイプライン型動作を有する分割トランズアクション実行の動作を示したタイミング線図。
【図２Ｃ】本発明の１実施例に基づいて分割トランズアクションコプロセッサ拡張に対する結果キューの有る場合及び無い場合且つパイプライン型及び非パイプライン型動作を有する分割トランズアクション実行の動作を示したタイミング線図。
【図３】本発明の１実施例に基づく分割トランズアクションコプロセッサ拡張ユニット内の非一様な命令レイテンシーを取扱う方法を示したブロック図。
【図４】本発明の１実施例に基づく分割トランズアクションコプロセッサ拡張ユニット内の複数個のコプロセッサ拡張の結果バスを接続する１つの可能なメカニズムを示した概略図。
【図５】本発明の１実施例に基づくガロア乗算器分割トランズアクションコプロセッサ拡張ユニットの動作を例示したタイミング線図。

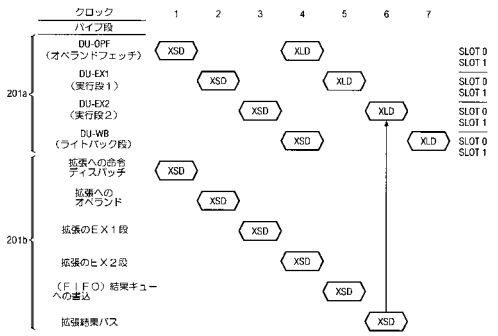
【符号の説明】

【００７８】
１００　プロセッサアーキテクチャ
１０１　コア
１０２　コプロセッサ拡張電気的インターフェース
１０３　コプロセッサ拡張ユニット
１０４　ＦＩＦＯ出力バッファ
１０４　ＦＩＦＯ結果キュー（ＲＱ）構造

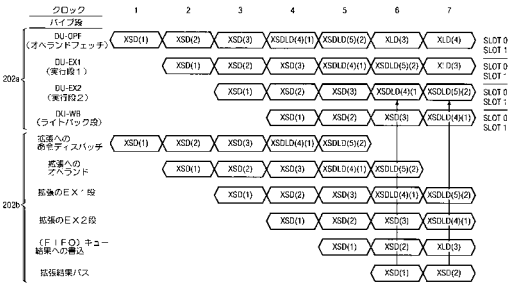
【図１】



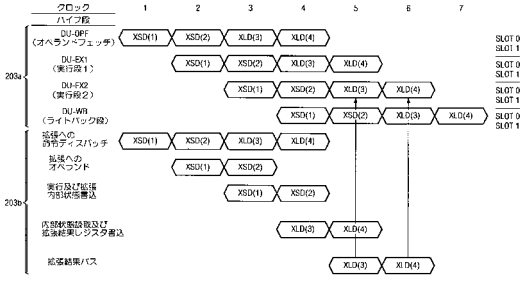
【図２Ａ】



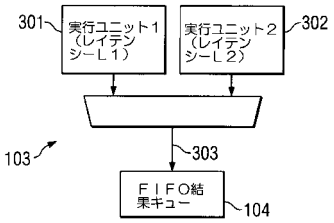
【図２Ｂ】



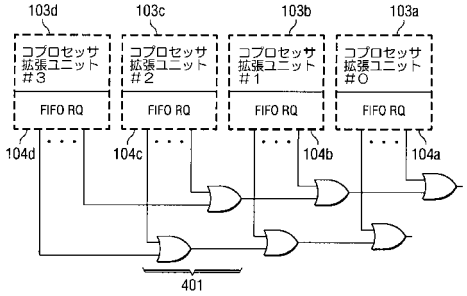
【図２Ｃ】



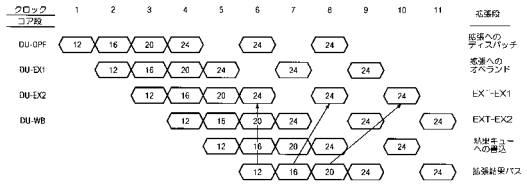
【図３】


301
実行ユニット１
（レイテンシー Ｌ１）
実行ユニット２
（レイテンシー Ｌ２）
302
103
303
ＦＩＦＯ結果キュー
104

【図４】


103d
103c
103b
103a
コプロセッサ拡張ユニット＃３
コプロセッサ拡張ユニット＃２
コプロセッサ拡張ユニット＃１
コプロセッサ拡張ユニット＃０
FIFO RQ
104d
104c
104b
104a
401

【図５】


クロック
コア段
1 2 3 4 5 6 7 8 9 10 11
処理段
DU-DPF
DU-EX1
DU-EX2
DU-WB
拡張へのディスパッチ
拡張へのオペランド
EX1-EX1
EX1-EX2
処理ユニットへの書込
拡張結果バス

フロントページの続き

(72)発明者　アレキサンダー　ドライカー
アメリカ合衆国,　カリフォルニア　９２１３０,　サン　ディエゴ,　マノー　リッジ　レーン　５０３３

Ｆターム(参考) 5B013 DD03