(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6228459号
(P6228459)
(45)発行日　平成29年11月8日(2017.11.8)　(24)登録日　平成29年10月20日(2017.10.20)

(51)Int.Cl.　F I
G06F 9/54 (2006.01)　G06F 9/46 480B

請求項の数 20 (全 21 頁)

(21)出願番号	特願2013-544736(P2013-544736)
(86)(22)出願日	平成23年12月14日(2011.12.14)
(65)公表番号	特表2013-546105(P2013-546105A)
(43)公表日	平成25年12月26日(2013.12.26)
(86)国際出願番号	PCT/US2011/064859
(87)国際公開番号	WO2012/082867
(87)国際公開日	平成24年6月21日(2012.6.21)
審査請求日	平成26年12月15日(2014.12.15)
審判番号	不服2015-21973(P2015-21973/J1)
審判請求日	平成27年12月11日(2015.12.11)
(31)優先権主張番号	61/422,953
(32)優先日	平成22年12月14日(2010.12.14)
(33)優先権主張国	米国(US)
(31)優先権主張番号	13/307,505
(32)優先日	平成23年11月30日(2011.11.30)
(33)優先権主張国	米国(US)

(73)特許権者　591016172
アドバンスト・マイクロ・ディバイシズ・インコーポレイテッド
ＡＤＶＡＮＣＥＤ　ＭＩＣＲＯ　ＤＥＶＩＣＥＳ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国、９４０８８－３４５３　カリフォルニア州、サニィベイル、ピィ・オゥ・ボックス・３４５３、ワン・エイ・エム・ディ・プレイス、メイル・ストップ・６８（番地なし）
(74)代理人　100108833
弁理士　早川　裕司
(74)代理人　100111615
弁理士　佐野　良太

最終頁に続く

(54)【発明の名称】システムコール要求の通信の最適化

(57)【特許請求の範囲】
【請求項１】
ウェーブフロント内のワークアイテムごとにシステムコール要求を処理するために、対応する単一命令複数データ（ＳＩＭＤ）エレメントに情報を記憶するステップと、
前記ＳＩＭＤエレメントをＳＩＭＤベクトルとしてまとめるステップと、
前記ＳＩＭＤベクトルを、実行対象としてプロセッサに送信するステップと、
各ワークアイテムについての結果を受信するステップと、
を含む、方法。
【請求項２】
前記ＳＩＭＤベクトルを、中央処理装置（ＣＰＵ）が認識することが可能な高優先度キュー内にエンキューするステップをさらに含む、
請求項１の方法。
【請求項３】
各ＳＩＭＤエレメントは、関数セレクタと、引数リストと、前記結果用のメモリ空間とを含む、
請求項１の方法。
【請求項４】
前記ＳＩＭＤベクトルは、複数のウェーブフロントからのシステムコール要求を含む、
請求項１の方法。
【請求項５】

少なくとも１つのプロセッサを有するコンピュータシステムで実行される方法であって、
ウェーブフロント内の各ワークアイテムからのシステムコール要求に対応する単一命令複数データ（ＳＩＭＤ）エレメントを含むＳＩＭＤベクトルを受信するステップと、
各ＳＩＭＤエレメントの各システムコール要求を実行するステップと、
各システムコールの結果を、前記ＳＩＭＤベクトルを用いて、前記ウェーブフロント内の各ワークアイテムに送信するステップと、
を含む、方法。
【請求項６】
前記受信するステップは、前記コンピュータシステム内のグラフィックス処理デバイスが認識することが可能な高優先度キュー内の前記ＳＩＭＤベクトルを受信するステップを含む、
請求項５の方法。
【請求項７】
ウェーブフロント内のワークアイテムごとのシステムコール要求を処理するために、対応する単一命令複数データ（ＳＩＭＤ）エレメントに情報を記憶するように構成されたメモリと、
ＣＰＵとを含み、
前記ＳＩＭＤエレメントは、ＳＩＭＤベクトルとしてまとめられ、
前記ＣＰＵは、
前記ＳＩＭＤエレメントに記憶された各システムコール要求を実行することと、
各システムコール要求の結果を、前記ウェーブフロント内の各ワークアイテムに送信することと、
を行うように構成されている、
システム。
【請求項８】
前記メモリは、前記ＳＩＭＤベクトルをエンキューするように構成された高優先度キューであり、前記高優先度キューは、ＣＰＵによって認識可能である、
請求項７のシステム。
【請求項９】
各ＳＩＭＤエレメントは、関数セレクタと、引数リストと、前記結果用のメモリ空間とを含む、
請求項７のシステム。
【請求項１０】
前記ＳＩＭＤベクトルは、複数のウェーブフロントからのシステムコール要求を含む、
請求項８のシステム。
【請求項１１】
メモリと、
ＣＰＵとを含み、
前記ＣＰＵは、
情報を含む単一命令複数データ（ＳＩＭＤ）エレメントを含むＳＩＭＤベクトルを受信して、対応するシステムコール要求をウェーブフロント内の各ワークアイテムごとに処理することと、
前記ＳＩＭＤベクトルに記憶された各システムコールを実行することと、
各システムコールの結果を、前記ウェーブフロント内の各ワークアイテムに送信することと、
を行うように構成されている、
システム。
【請求項１２】
前記ＣＰＵは、

高優先度キューから前記ＳＩＭＤベクトルを受信すること、を行うように構成されている、
請求項１１のシステム。
【請求項１３】
コンピュータ記憶デバイスに記憶された命令であって、前記命令がコンピューティングデバイスによって実行されると、
対応する単一命令複数データ（ＳＩＭＤ）エレメントに情報を記憶して、ウェーブフロント内のワークアイテムごとにシステムコール要求を処理することと、
前記ＳＩＭＤエレメントをＳＩＭＤベクトルとしてまとめて、システムコール要求データ構造を生成することと、
前記ＳＩＭＤベクトルを、実行対象として前記コンピューティングデバイス内のプロセッサに送信することと、
前記ウェーブフロント内の各ワークアイテムについての結果を受信することと、
を前記コンピューティングデバイスに実行させる、
命令。
【請求項１４】
前記ＳＩＭＤベクトルを、前記プロセッサが認識することが可能な高優先度キュー内にエンキューすることをさらに含む、
請求項１３の命令。
【請求項１５】
コンピュータ記憶デバイスに記憶された命令であって、前記命令がコンピューティングデバイスによって実行されると、
ウェーブフロント内の各ワークアイテムからのシステムコール要求に対応する単一命令複数データ（ＳＩＭＤ）エレメントを含むＳＩＭＤベクトルを受信することと、
前記ＳＩＭＤベクトルからの各システムコール要求を実行することと、
各システムコールの結果を、前記ウェーブフロント内の各ワークアイテムに送信することと、
を前記コンピューティングデバイスに実行させる、
命令。
【請求項１６】
前記受信することは、グラフィックス処理デバイスが認識することが可能な高優先度キュー内の前記ＳＩＭＤベクトルを受信することを含む、
請求項１５の命令。
【請求項１７】
コンピュータ記憶デバイスであって、前記デバイスには命令が記憶されており、前記命令がコンピューティングデバイスによって実行されると、
対応する単一命令複数データ（ＳＩＭＤ）エレメントに情報を記憶して、ウェーブフロント内のワークアイテムごとにシステムコール要求を処理することと、
前記記憶されたシステムコールを、ＳＩＭＤベクトルを用いて、実行対象として前記コンピューティングデバイス内のプロセッサに送信することであって、前記ＳＩＭＤベクトルは、単一データ構造としてまとめられた前記ＳＩＭＤエレメントを含むことと、
前記送信に応じて、前記ウェーブフロント内の各ワークアイテムについての結果を受信することと、
を前記コンピューティングデバイスに実行させる、
コンピュータ記憶デバイス。
【請求項１８】
前記ＳＩＭＤベクトルを、前記プロセッサが認識することが可能な高優先度キュー内にエンキューすることをさらに含む、
請求項１７のコンピュータ記憶デバイス。
【請求項１９】

コンピュータ記憶デバイスであって、前記デバイスには命令が記憶されており、前記命令がコンピューティングデバイスによって実行されると、
ウェーブフロント内の各ワークアイテムからのシステムコール要求に対応する単一命令複数データ（ＳＩＭＤ）エレメントを含むＳＩＭＤベクトルを受信することと、
前記ＳＩＭＤベクトルからの各システムコール要求を実行することと、
各システムコールの結果を、前記ウェーブフロント内の各ワークアイテムに送信することと、
を前記コンピューティングデバイスに実行させる、
コンピュータ記憶デバイス。
【請求項２０】
前記ＳＩＭＤベクトルは、グラフィックス処理デバイスが認識することが可能な高優先度キューから受信される、
請求項１９のコンピュータ記憶デバイス。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、概して、コンピュータシステムを対象とする。さらに詳細には、本発明は、コンピューティングシステム内のコンピュータコンポーネントの統一のためのアーキテクチャに関する。
【背景技術】
【０００２】
グラフィックス処理ユニット（ＧＰＵ：ｇｒａｐｈｉｃｓ　ｐｒｏｃｅｓｓｉｎｇ　ｕｎｉｔ）を一般的な演算に用いる要請は、ＧＰＵの単位電力および／またはコストあたりの能力が優れることから、最近ではより顕著となってきている。ＧＰＵの計算能力は、対応する中央処理装置（ＣＰＵ：ｃｅｎｔｒａｌ　ｐｒｏｃｅｓｓｉｎｇ　ｕｎｉｔ）の計算能力の成長速度を超える速度で成長してきた。モバイルコンピューティング市場（例えば、ノートブック、モバイルスマートフォン、タブレット、その他）およびその必要な支援サーバ／企業システムの爆発的な成長と組み合わされたこの成長は、特定品質の所望のユーザ経験を提供するために用いられてきている。したがって、並列データの内容を有する作業負荷を実行するためにＣＰＵとＧＰＵとを併用することはボリュームテクノロジとなりつつある。
【０００３】
しかし、ＧＰＵは、従来、主にグラフィックの高速化のために利用可能な制約されたプログラミング環境で動作する。これらの制約は、ＧＰＵが、ＣＰＵと同程度に豊かなプログラミングエコシステムを有さないという事実に起因するものである。したがって、ＧＰＵの使用は、主に、グラフィックスおよびビデオのアプリケーションプログラミングインターフェース（ＡＰＩ：ａｐｐｌｉｃａｔｉｏｎ　ｐｒｏｇｒａｍｍｉｎｇ　ｉｎｔｅｒｆａｃｅ）を処理することに既に順応した、２次元（２Ｄ）グラフィックスと、３次元（３Ｄ）グラフィックスと、少数の最先端マルチメディアアプリケーションとに限られていた。
【０００４】
マルチベンダにサポートされるＯｐｅｎＣＬ（登録商標）、ＤｉｒｅｃｔＣｏｍｐｕｔｅ（登録商標）、標準的なＡＰＩおよびサポート用ツールの出現とともに、従来の用途におけるＧＰＵの限界は、従来のグラフィックスを越えて拡張されてきている。ＯｐｅｎＣＬ（登録商標）およびＤｉｒｅｃｔＣｏｍｐｕｔｅ（登録商標）は将来性のあるスタートではあるが、大部分のプログラミングタスクに対して、ＣＰＵと同程度に流動的にＣＰＵとＧＰＵとの組合せが用いられることを可能にする環境およびエコシステムを作成するには、多数のハードルが残されている。
【０００５】
既存のコンピューティングシステムは、大抵、複数の処理装置を含む。例えば、いくつ

かのコンピューティングシステムは、ＣＰＵおよびＧＰＵの両方を、別個のチップ上（例えば、ＣＰＵはマザーボード上に、ＧＰＵはグラフィックカード上に配置され得る）または単一チップパッケージ上に有し得る。これらの構成の両方は、（ｉ）別個のメモリシステム、（ｉｉ）効率的なスケジューリング、（ｉｉｉ）プロセス間のサービス品質（ＱｏＳ）の保証、（ｉｖ）プログラミングモデルおよび（ｖ）複数のターゲット命令セットアーキテクチャ（ＩＳＡ：ｉｎｓｔｒｕｃｔｉｏｎ　ｓｅｔ　ａｒｃｈｉｔｅｃｔｕｒｅ）へのコンパイル、の全項目を、電力消費を最小化しながら満足することに関して顕著な問題点を依然として含む。

【０００６】

例えば、個別チップ構成においては、各プロセッサがメモリにアクセスするためには、システムアーキテクチャおよびソフトウェアアーキテクチャは、チップ間インターフェースを利用することを余儀なくされる。これらの外部インターフェース（例えばチップ間インターフェース）においては、異機種環境のプロセッサを協働させるために、メモリレイテンシおよび電力消費に弊害がもたらされる一方で、別個のメモリシステム（すなわち、別個のアドレス空間）およびドライバに管理される共有メモリにおいては、きめ細かな負荷軽減に対しては許容されないオーバーヘッドが生成されてしまうこととなる。

【０００７】

別の例において、いくつかのコマンドは、ＧＰＵにて効率的に実行できない場合がある。例えば、ＧＰＵは、オペレーティングシステム（ＯＳ）を用いたコマンドを効率的に実行できない場合がある（例えば、メモリまたは印刷データを、コンピュータ画面に割りあてる命令は、ＣＰＵでしか処理することができない）。このように、ＧＰＵは、これらのタスクを実行することができないため、ＣＰＵに対して、これらのタスクを実行するように要求する。これらの要求は、システムコール（シスコール）として知られている。

【０００８】

シスコールは、ＣＰＵによる処理対象としては高コストである。すなわち、シスコールは、高優先度コマンドであるため、ＣＰＵに送球に対応してもらわなくてはならない場合が多い。ＣＰＵは、シスコール要求を受信するたびに、現在のプロセス処理を停止して、ＯＳを呼び出し、当該シスコールを処理した後に処理作業に戻る。

【０００９】

ＧＰＵがウェーブフロントを処理する場合には、各ワークアイテムは、メモリ割りあてまたはＧＰＵによる処理が不可能（若しくは処理が容易ではない）な他の命令のために、シスコールを必要とし得る。従来のシステムでは、ＧＰＵは、ワークアイテムごとに別個のシスコール要求をＣＰＵに発行する。各ワークアイテムは並列に実行するため、同一のシスコール要求が、各ワークアイテムからＣＰＵへ発行される。

【００１０】

ＣＰＵにシスコール要求が到着するたびに、ＣＰＵは、処理作業を停止し、ＯＳを呼び出し、ＧＰＵからの要求を処理した後に元の処理作業に戻る。複数のワークアイテムから別個のシスコール要求が同時に発行された場合には、ＣＰＵの処理時間の無駄が発生する。なぜならば、ＣＰＵは、自身の処理作業を繰り返して一時停止し、ＯＳを呼び出し、ＧＰＵからのシスコール要求を処理するからである。

【発明の概要】

【００１１】

したがって、必要とされるのは、ＣＰＵとＧＰＵとの間のシスコールに関連する通信を最適化する（すなわち、向上させる）ためのシステムおよび方法である。

【００１２】

ＧＰＵと、アクセラレーテッドプロセッシングユニット（ＡＰＵ）と、ＧＰＵによる汎目的計算（ＧＰＧＰＵ）とは、この分野において一般に用いられる用語であるが、「アクセラレーテッド処理デバイス」（ＡＰＤ）は、より広い表現とみなされる。例えば、ＡＰＤは、従来のＣＰＵ、従来のＧＰＵ、ソフトウェアおよび／またはこれらの組合せと比較して高速化された方法で行われる、グラフィックス処理タスク、データ並列タスク、また

はネスト化されたデータ並列タスクの高速化に関連する機能および演算を実行する任意のハードウェアおよび／またはソフトウェアの協働的集合体を指す。

【００１３】

本発明の実施形態は、システムコールの通信を最適化するためのシステム、方法および製品を含む。上記方法は、ウェーブフロント内のワークアイテムごとのシステムコールを記憶するステップと、前記記憶されたシステムコールを実行対象としてプロセッサに送信するステップとを含む。また、上記方法は、前記送信に応じて、前記ウェーブフロント内のワークアイテムごとの結果を受信するステップを含む。

【００１４】

本発明の追加的な特徴および利点、ならびに本発明の様々な実施形態の構造および動作は、添付の図面を参照して以下で詳細に説明される。本発明は本明細書において説明される特定の実施形態に制限されないことに注意すべきである。係る実施形態は、例示目的のみのために本明細書において提示される。追加的実施形態は、本明細書に含まれる教示に基づいて当業者に明らかとなるであろう。

【００１５】

添付の図面は、本明細書に組み込まれ本明細書の一部を構成するものであって、本発明を例示し、説明とともに、本発明の原理を説明し、当業者が本発明を実施することを可能にするように、さらに機能する。本発明の様々な実施形態は、添付の図面を参照して以下で説明される。なお、添付の図面の全体を通じて、同様の参照番号は同様の構成要素を参照するために用いられる。

【図面の簡単な説明】

【００１６】

【図１Ａ】本発明の実施形態に係る処理システムの例示的ブロック図である。

【図１Ｂ】図１Ａに示されたＡＰＤの例示的ブロック図である。

【図２】ＣＰＵとＡＰＤとの間の最適化された通信処理の例示的ブロック図２００である。

【図３】シスコール要求をＣＰＵに通信するための単一命令複数データ（ＳＩＭＤ）ベクトルを用いたＡＰＤの例示的フローチャート３００である。

【発明を実施するための形態】

【００１７】

以下の詳細な説明においては、「１つの実施形態」、「ある実施形態」、「例示的実施形態」またはその他を参照することは、本明細書において説明される実施形態が、特定の特徴、構造または特性を有することを示すが、全ての実施形態がその特定の特徴、構造または特性を必ずしも含むとは限らない。さらに、係る語句は、同一の実施形態を参照するとは限らない。さらに、特定の特徴、構造または特性が１つの実施形態に関連して説明されるとき、係る特長、構造または特性を他の実施形態との関連で実施することは、明示されているか否かによらず、当業者の知識の範囲内であることが提起される。

【００１８】

「本発明の実施形態」という用語は、本発明の実施形態の全部が検討の対象である特徴、特長または操作モードを含むことを必ずしも要求しない。代替的な実施形態が本発明の範囲から逸脱することなく考案され、本発明の既知の構成要素は、本発明の関連する詳細を不明瞭化しないために、必ずしも詳細に説明されるとは限らず、または省略され得る。加えて、本明細書において用いられる用語は、特定の実施形態を説明することのみを目的とし、本発明を制限することを意図するものではない。例えば、本明細書において用いられる単数形の「１つの」および「その」は、内容的に明らかに単数のみを指す場合を除き、複数形をも含むことを意図するものである。本明細書において用いられる場合、「含む」、「備える」、「包含する」および／または「有する」という用語は、述べられた特徴、整数、ステップ、操作、構成要素および／またはコンポーネントが存在することを指定するが、１つまたは複数の他の特徴、整数、ステップ、操作、構成要素、コンポーネントおよび／またはこれらの群の存在または追加を除外しない。

【００１９】
図１Ａは、ＣＰＵ１０２およびＡＰＤ１０４を含む統合化されたコンピューティングシステム１００の例示的な図である。ＣＰＵ１０２は、１つ以上のシングルコアまたはマルチコアＣＰＵを含み得る。本発明の一実施形態において、システム１００は、統合化されたプログラミング環境および実行環境を提供するために、ＣＰＵ１０２およびＡＰＤ１０４を組み合わせた単一のシリコンダイまたはパッケージ上に形成される。この環境は、ＡＰＤ１０４が、いくつかのプログラミングタスクに対して、ＣＰＵ１０２と同程度に流動的に用いられることを可能にする。しかし、ＣＰＵ１０２およびＡＰＤ１０４が単一のシリコンダイ上に形成されることは、本発明の絶対的な要件ではない。いくつかの実施形態において、ＣＰＵ１０２およびＡＰＤ１０４は別個に形成され、同一の基板または異なる基板上に搭載されることが可能である。
【００２０】
１つの例において、システム１００は、メモリ１０６、オペレーティングシステム１０８および通信インフラストラクチャ１０９を含む。オペレーティングシステム１０８および通信インフラストラクチャ１０９は、以下でより詳細に説明される。
【００２１】
システム１００は、カーネルモードドライバ（ＫＭＤ：ｋｅｒｎｅｌ　ｍｏｄｅ　ｄｒｉｖｅｒ）１１０と、ソフトウェアスケジューラ（ＳＷＳ：ｓｏｆｔｗａｒｅ　ｓｃｈｅｄｕｌｅｒ）１１２と、例えば入出力メモリ管理ユニット（ＩＯＭＭＵ：ｉｎｐｕｔ／ｏｕｔｐｕｔ　ｍｅｍｏｒｙ　ｍａｎａｇｅｍｅｎｔ　ｕｎｉｔ）等のメモリ管理ユニット１１６とを含む。システム１００の構成品は、ハードウェア、ファームウェア、ソフトウェアまたはこれらの任意の組合せとして実装され得る。当業者は、システム１００が図１Ａに示される実施形態において示されるものに加えて、またはこれらとは異なる、１つ以上のソフトウェアコンポーネント、ハードウェアコンポーネントおよびファームウェアコンポーネントを含み得ることを理解するであろう。
【００２２】
１つの例において、ＫＭＤ１１０等のドライバは、通常、ハードウェアが接続されたコンピュータバスまたは通信サブシステムを通してデバイスと通信する。呼び出しプログラムがドライバにおいてルーチンを呼び出すと、ドライバは、デバイスに対してコマンドを発行する。デバイスがドライバに対してデータを戻すと、ドライバは、元の呼び出しプログラムにおいてルーチンを呼び出し得る。１つの例において、ドライバは、ハードウェア依存的であり、オペレーティングシステム固有である。ドライバは、通常、任意の必要な非同期的時間依存性のハードウェアインターフェースに対して要求されるインタラプトハンドリングを提供する。デバイスドライバは、特に現代のウィンドウズ（登録商標）プラットフォームにおいては、カーネルモード（リング０）またはユーザモード（リング３）において実行され得る。
【００２３】
ドライバをユーザモードにおいて実行することの利益は、安定性が改善されることである。なぜなら、不完全な形で書かれたユーザモードデバイスドライバは、カーネルメモリを上書きすることによってシステムをクラッシュさせることができないためである。一方、ユーザモード／カーネルモードの遷移は、通常、顕著な性能オーバーヘッドを与え、それにより、低レイテンシおよび高スループット要件目的のユーザモードドライバを阻害する。カーネルスペースは、システムコールの使用を通してのみ、ユーザモジュールからのアクセスが可能である。ＵＮＩＸ（登録商標）シェルまたは他のＧＵＩベースのアプリケーションなどのエンドユーザプログラムは、ユーザスペースの一部である。これらのアプリケーションは、カーネルにサポートされた機能を通してハードウェアと相互作用する。
【００２４】
ＣＰＵ１０２は、制御プロセッサ、フィールド・プログラマブル・ゲートアレイ（ＦＰＧＡ：ｆｉｅｌｄ　ｐｒｏｇｒａｍｍａｂｌｅ　ｇａｔｅ　ａｒｒａｙ）、特定用途集積回路（ＡＳＩＣ：ａｐｐｌｉｃａｔｉｏｎ　ｓｐｅｃｉｆｉｃ　ｉｎｔｅｇｒａｔｅｄ

ｃｉｒｃｕｉｔ）またはデジタル・シグナル・プロセッサ（ＤＳＰ：ｄｉｇｉｔａｌ ｓｉｇｎａｌ ｐｒｏｃｅｓｓｏｒ）のうち１つ以上を含み得る（図示せず）。ＣＰＵ１０２は、例えば、コンピューティングシステム１００の動作を制御する、オペレーティングシステム１０８、ＫＭＤ１１０、ＳＷＳ１１２およびアプリケーション１１１を含む制御ロジックを実行する。この例示的な実施形態において、ＣＰＵ１０２は、１つの実施形態によれば、アプリケーション１１１の実行の起動および制御を、例えばそのアプリケーションに関連する処理をＣＰＵ１０２とＡＰＤ１０４等の他の処理リソースとの間に分散させることにより行う。

【００２５】

ＡＰＤ１０４は、とりわけ、グラフィックス演算や、例えば特に並列処理に好適となり得る他の演算等の、選択された機能のためのコマンドおよびプログラムを実行する。一般に、ＡＰＤ１０４は、ピクセル処理、幾何学演算およびディスプレイに対する画像のレンダリング等のグラフィックスパイプライン処理を実行するために、しばしば用いられ得る。本発明の様々な実施形態において、ＡＰＤ１０４は、ＣＰＵ１０２から受信したコマンドまたは命令に基づいて、計算処理演算も実行し得る。

【００２６】

例えば、コマンドは、ＩＳＡに定義されない特殊な命令であるとみなすことができ、所与のＩＳＡまたは独特なハードウェアからの１組の命令によって大抵得られる。コマンドは、例えばディスパッチプロセッサ、コマンドプロセッサまたはネットワークコントローラ等の特殊なプロセッサにより実行され得る。一方、命令は、例えばコンピュータアーキテクチャ内のプロセッサの単一の演算であるとみなされ得る。１つの例において、２セットのＩＳＡが用いられる場合には、いくつかの命令は、ｘ８６プログラムを実行するために用いられ、いくつかの命令は、ＡＰＤ計算ユニット上でカーネルを実行するために用いられる。

【００２７】

例示的な実施形態において、ＣＰＵ１０２は、選択されたコマンドをＡＰＤ１０４に伝達する。これらの選択されたコマンドは、並列実行に適したグラフィックスコマンドと、並列実行に適した他のコマンドとを含み得る。計算処理コマンドを含み得るこれらの選択されたコマンドは、実質的にＣＰＵ１０２から独立して実行され得る。

【００２８】

ＡＰＤ１０４は、例えば１つ以上の単一命令複数データ（ＳＩＭＤ：ｓｉｎｇｌｅ ｉｎｓｔｒｕｃｔｉｏｎ ｍｕｌｔｉｐｌｅ ｄａｔａ）処理コア等ではあるがこれに制限されない、自身の計算ユニット（図示せず）を含み得る。本明細書で参照されるＳＩＭＤは、数学パイプライン、すなわち、それぞれが自身のデータおよび共有プログラムカウンタを有する複数の処理エレメント上でカーネルが同時に実行されるプログラミングモデルである。全ての処理エレメントは、完全に同一の１組の命令を実行する。プレディケーションを用いることにより、発行された各コマンドに対して、ワークアイテムを関与させることまたはさせないことが可能となる。

【００２９】

１つの例において、各ＡＰＤ計算ユニット１０４は、１つ以上のスカラーおよび／またはベクトル浮動小数点演算ユニットおよび／または算術論理ユニット（ＡＬＵ：ａｒｉｔｈｍｅｔｉｃ ａｎｄ ｌｏｇｉｃ ｕｎｉｔ）を含み得る。ＡＰＤ計算ユニットは、逆平方根ユニットおよびサイン／コサインユニット等の特殊用途処理ユニット（図示せず）も含み得る。１つの例において、ＡＰＤ計算ユニットは、本明細書においてシェーダコア１２２と総称される。

【００３０】

１つ以上のＳＩＭＤが存在することにより、一般に、ＡＰＤ１０４は、グラフィックス処理において一般的なデータ並列タスク等のデータ並列タスクの実行に、理想的に好適なものとなる。

【００３１】

ピクセル処理等のいくつかのグラフィックスパイプライン処理と、他の並列演算処理とは、同一のコマンドストリームまたは計算カーネルが、ストリームまたは入力データ要素の集合体上で実行されることを要求する。同一の計算カーネルのそれぞれのインスタンス化は、上記データ要素を並列に処理するために、シェーダコア１２２の複数の計算ユニット上で同時に実行され得る。本明細書で参照されるように、例えば計算カーネルは、プログラム上で宣言され、ＡＰＤ計算ユニット上で実行される命令を含む関数である。この関数は、カーネル、シェーダ、シェーダプログラムまたはプログラムとも称される。

【００３２】

１つの例示的な実施形態において、各計算ユニット（例えばＳＩＭＤ処理コア）は、入力されるデータを処理するために、特定ワークアイテムのそれぞれのインスタンス化を実行し得る。ワークアイテムは、コマンドによりデバイス上で呼び出されるカーネルの並列実行の集合体のうちの１つである。ワークアイテムは、計算ユニット上で実行されるワークグループの一部として、１つ以上の処理エレメントにより実行され得る。

【００３３】

ワークアイテムは、自身のグローバルＩＤおよびローカルＩＤによって、集合体内の他の実行から区別される。１つの例において、ワークグループにおいて１つのＳＩＭＤエンジンで同時に実行されるワークアイテムの一部は、ウェーブフロント１３６と称され得る。ウェーブフロントの幅は、計算ユニット（例えば、ＳＩＭＤ処理コア）のハードウェアの特性である。本明細書で参照されるワークグループは、単一の計算ユニット上で実行される関連するワークアイテムの集合体である。ワークグループ内のワークアイテムは、同一のカーネルを実行し、ローカルメモリおよびワークグループバリアを共有する。

【００３４】

１つのワークグループからの全てのウェーブフロントは、同一のＳＩＭＤエンジンで処理される。ウェーブフロントにわたる命令は１つずつ発行され、全てのワークアイテムが同一の制御フローに従う場合には、各ワークアイテムは、同一のプログラムを実行する。実行マスクおよびワークアイテムプレディケーションは、ウェーブフロント内の拡散的な制御フローを可能にするために用いられる。なお、拡散的な制御フローにおいては、各個別のワークアイテムは、カーネルを通じて一意的なコードパスを実際に取り得る。部分的に占められたウェーブフロントは、ワークアイテムの全てのセットがウェーブフロント開始時にて必ずしも利用可能とは限らない場合に、処理され得る。ウェーブフロントは、ワープ、ベクトルまたはスレッドと称され得る。

【００３５】

コマンドは、ウェーブフロントに対して１つずつ発行され得る。全てのワークアイテムが同一の制御フローに従う場合には、各ワークアイテムは同一のプログラムを実行し得る。１つの例では、実行マスクおよびワークアイテムプレディケーションは、各個別のワークアイテムがカーネルドライバを通じて一意的なコードパスを実際に取り得る拡散的な制御フローを可能にするために用いられる。ワークアイテムの全てのセットが開始時において利用できない場合には、部分的なウェーブフロントを処理することができる。例えば、シェーダコア１２２は、所定数のウェーブフロント１３６を同時に実行することができ、各ウェーブフロント１３６は、所定数のワークアイテムを含む。

【００３６】

システム１００において、ＡＰＤ１０４は、グラフィックスメモリ１３０等の自身のメモリを含む。グラフィックスメモリ１３０は、ＡＰＤ１０４における計算実行の間の使用のために、ローカルメモリを提供する。シェーダコア１２２内の個々の計算ユニット（図示せず）は、自身のローカルデータ記憶装置（図示せず）を有し得る。１つの実施形態において、ＡＰＤ１０４は、ローカルグラフィックスメモリ１３０へのアクセス、ならびにメモリ１０６へのアクセスを含む。他の実施形態において、ＡＰＤ１０４は、ダイナミックランダムアクセスメモリ（ＤＲＡＭ：ｄｙｎａｍｉｃ　ｒａｎｄｏｍ　ａｃｃｅｓｓ　ｍｅｍｏｒｙ）、または、ＡＰＤ１０４には直接的に接続されているがメモリ１０６からは分離している他のメモリ（図示せず）へのアクセスを含み得る。

【００３７】
図示の例において、ＡＰＤ１０４は、１つまたは「ｎ」個のコマンドプロセッサ（ＣＰ：ｃｏｍｍａｎｄ　ｐｒｏｃｅｓｓｏｒ）１２４を含み得る。ＣＰ１２４は、ＡＰＤ１０４内の処理を制御する。ＣＰ１２４は、実行されるべきコマンドを、メモリ１０６内のコマンドバッファ１２５から取得し、ＡＰＤ１０４でのこれらのコマンドの実行を調整する。

【００３８】
１つの例において、ＣＰＵ１０２は、アプリケーション１１１に基づくコマンドを、適切なコマンドバッファ１２５に入力する。本明細書において参照されるように、アプリケーションは、ＣＰＵ内またはＡＰＤ内の計算ユニット上で実行されるプログラム部分の組合せである。

【００３９】
複数のコマンドバッファ１２５は、各プロセスがＡＰＤ１０４での実行のためにスケジュールされた状態で、保持され得る。

【００４０】
ＣＰ１２４は、ハードウェア、ファームウェア、ソフトウェアまたはこれらの組合せにおいて実装され得る。１つの実施形態において、ＣＰ１２４は、スケジューリングロジックを含むロジックを実装するためのマイクロコードを有する縮小命令セットコンピュータ（ＲＩＳＣ：ｒｅｄｕｃｅｄ　ｉｎｓｔｒｕｃｔｉｏｎ　ｓｅｔ　ｃｏｍｐｕｔｅｒ）エンジンとして実装される。

【００４１】
ＡＰＤ１０４は、１つまたは「ｎ」個のディスパッチコントローラ（ＤＣ：ｄｉｓｐａｔｃｈ　ｃｏｎｔｒｏｌｌｅｒ）１２６を含み得る。本願において、ディスパッチという用語は、１セットの計算ユニット上の１セットのワークグループに対するカーネルの実行のスタートを起動するために、コンテキスト状態を使用するディスパッチコントローラにより実行されるコマンドを指す。ＤＣ１２６は、シェーダコア１２２内でワークグループを起動するためのロジックを含む。いくつかの実施形態において、ＤＣ１２６は、ＣＰ１２４の一部として実装され得る。

【００４２】
システム１００は、ＡＰＤ１０４上で実行するためのプロセスを実行リスト１５０から選択するためのハードウェアスケジューラ（ＨＷＳ：ｈａｒｄｗａｒｅ　ｓｃｈｅｄｕｌｅｒ）１２８を含む。ＨＷＳ１２８は、ラウンドロビン方式、優先レベルを用いて、または他のスケジューリングポリシーに基づいて、プロセスを実行リスト１５０から選択し得る。優先レベルは、例えば、動的に決定され得る。ＨＷＳ１２８は、例えば、新規のプロセスを追加することによって、あるいは既存のプロセスを実行リスト１５０から削除することによって、実行リスト１５０を管理する機能を含み得る。ＨＷＳ１２８の実行リスト管理ロジックは、実行リストコントローラ（ＲＬＣ：ｒｕｎ　ｌｉｓｔ　ｃｏｎｔｒｏｌｌｅｒ）と称されることもある。

【００４３】
本発明の様々な実施形態において、ＨＷＳ１２８が、実行リスト１５０からプロセスの実行を開始すると、ＣＰ１２４は、対応するコマンドバッファ１２５からのコマンドの取得および実行を開始する。いくつかの事例において、ＣＰ１２４は、ＣＰＵ１０２から受信したコマンドに対応する、ＡＰＤ１０４内で実行される１つ以上のコマンドを生成し得る。１つの実施形態において、ＣＰ１２４は、ＡＰＤ１０４リソースおよび／またはシステム１００のリソースの利用が改善または最大化されるように、ＡＰＤ１０４におけるコマンドの優先化およびスケジューリングを、他のコンポーネントとともに実装する。

【００４４】
ＡＰＤ１０４は、インタラプトジェネレータ１４６に対してアクセスを有するか、またはインタラプトジェネレータ１４６を含み得る。インタラプトジェネレータ１４６は、ＡＰＤ１０４がページフォールト等のインタラプトイベントに遭遇すると、ＡＰＤ１０４に

よってオペレーティングシステム１０８にインタラプトをかけるよう構成され得る。例えば、ＡＰＤ１０４は、ＩＯＭＭＵ１１６内のインタラプト生成ロジックに依存して、上述のページフォールトインタラプトを生成し得る。

【００４５】

ＡＰＤ１０４は、シェーダコア１２２内で現在実行中のプロセスを切り替えるためのプリエンプションおよびコンテキストスイッチロジック１２０を含み得る。コンテキストスイッチロジック１２０は、例えばプロセスを停止させ、その現在状態（例えばシェーダコア１２２状態およびＣＰ１２４状態）を保存する機能を含む。

【００４６】

本明細書において参照される状態という用語は、初期状態、中間状態および／または最終状態を含み得る。初期状態は、機械がプログラム順序に従って入力データセットを処理することによって、データの出力セットを生成する開始点である。例えば、処理を前進させるためにいくつかのポイントにおいて記憶される必要がある中間状態が存在する。この中間状態は、他のプロセスによってインタラプトがかけられた場合に、後に実行を継続することを可能にするために記憶される場合もある。出力データセットの一部として記録され得る最終状態も存在する。

【００４７】

プリエンプションおよびコンテキストスイッチロジック１２０は、他のプロセスを、ＡＰＤ１０４にコンテキストスイッチするためのロジックを含み得る。他のプロセスをＡＰＤ１０４で実行するようにコンテキストスイッチするための機能は、ＡＰＤ１０４上で実行するために、例えばＣＰ１２４およびＤＣ１２６によってプロセスをインスタンス化することと、当該プロセスに対して以前に保存された状態を復元することと、当該プロセスの実行を開始することと、を含み得る。

【００４８】

メモリ１０６は、ＤＲＡＭ（図示せず）等の非永続型メモリを含み得る。メモリ１０６は、例えば、アプリケーションまたは他の処理ロジックの部分を実行する間に、処理ロジック命令、定数および様々な変数を記憶し得る。例えば、１つの実施形態において、ＣＰＵ１０２上で１つ以上の演算を実行するための制御ロジックの部分は、ＣＰＵ１０２によって演算のそれぞれの部分が実行される間、メモリ１０６内に常駐し得る。本明細書において用いられる「処理ロジック」または「ロジック」といいう用語は、制御フローコマンド、計算実行コマンドおよびリソースアクセス関連コマンドを指す。

【００４９】

実行中、個別のアプリケーション、オペレーティングシステム関数、処理ロジックコマンドおよびシステムソフトウェアは、メモリ１０６に常駐し得る。オペレーティングシステム１０８に対して必須である制御ロジックコマンドは、一般に、実行中にはメモリ１０６に常駐することとなるであろう。他のソフトウェアコマンド、例えばカーネルモードドライバ１１０およびソフトウェアスケジューラ１１２は、システム１００の実行中にはメモリ１０６に常駐し得る。

【００５０】

この例において、メモリ１０６は、コマンドをＡＰＤ１０４に送るために、ＣＰＵ１０２によって用いられるコマンドバッファ１２５を含む。メモリ１０６は、プロセスリストおよびプロセス情報（例えば、アクティブリスト１５２およびプロセス制御ブロック１５４）を含み得る。これらのリストおよび情報は、スケジュール情報を、ＡＰＤ１０４および／または関連するスケジューリングハードウェアに伝えるために、ＣＰＵ１０２上で実行されるスケジューリングソフトウェアによって使用される。メモリ１０６に対するアクセスは、メモリ１０６に接続されたメモリコントローラ１４０によって管理され得る。例えば、メモリ１０６に対する読み出しおよび書き込みを実行するための、ＣＰＵ１０２または他のデバイスからの要求は、メモリコントローラ１４０によって管理される。

【００５１】

システム１００の他の態様に戻ると、ＩＯＭＭＵ１１６は、マルチコンテキスト・メモ

リ管理ユニットである。

【００５２】

本明細書で用いられるコンテキスト（プロセスともよばれる）という用語は、カーネルが実行される環境であって、同期およびメモリ管理が定義されるドメインであるとみなされる。コンテキストは、１セットのデバイスと、これらのデバイスに対してアクセス可能であるメモリと、対応するメモリ特性と、メモリオブジェクトにおけるカーネル（単数または複数）または演算の実行をスケジュールするために用いられる１つ以上のコマンドキューとを含む。一方、プロセスは、コンピュータ上で実行するプロセスを発生するアプリケーション用のプログラムの実行とみなすことができる。オペレーティングシステムは、実行対象プログラム用のデータ記録および仮想メモリアドレス空間を生成し得る。プログラム実行のメモリおよび現在の状態は、プロセスと呼ばれ得る。オペレーティングシステムは、メモリ上で動作させるべきプロセス用のタスクを、初期状態から最終状態までスケジュールする。

【００５３】

図１Ａにおいて示される例に戻ると、ＩＯＭＭＵ１１６は、ＡＰＤ１０４を含むデバイスに対するメモリページアクセスに対して、仮想アドレスから物理アドレスへの変換を実行するためのロジックを含む。ＩＯＭＭＵ１１６は、例えば、ＡＰＤ１０４等のデバイスによるページアクセスの結果としてページフォールトが生じる場合に、インタラプトを生成するためのロジックを含み得る。ＩＯＭＭＵ１１６は、トランスレーションルックアサイドバッファ（ＴＬＢ：ｔｒａｎｓｌａｔｉｏｎ　ｌｏｏｋａｓｉｄｅ　ｂｕｆｆｅｒ）１１８を含むか、あるいはＴＬＢ１１８に対するアクセスを有し得る。ＴＬＢ１１８は、１つの例として、メモリ１０６内のデータ用にＡＰＤ１０４によりなされた要求に対して、論理（すなわち仮想）メモリアドレスから物理メモリアドレスへの変換を高速化するために、コンテントアドレサブルメモリ（ＣＡＭ：ｃｏｎｔｅｎｔ　ａｄｄｒｅｓｓａｂｌｅ　ｍｅｍｏｒｙ）に実装され得る。

【００５４】

示された例において、通信インフラストラクチャ１０９は、必要に応じてシステム１００のコンポーネントを相互接続する。通信インフラストラクチャ１０９は、周辺構成要素相互接続（ＰＣＩ）バス、拡張ＰＣＩ（ＰＣＩ－Ｅ）バス、アドバンスト・マイクロコントローラ・バス・アーキテクチャ（ＡＭＢＡ）バス、アドバンスト・グラフィックス・ポート（ＡＧＰ）または他の通信インフラストラクチャのうち１つ以上を含み得る（図示せず）。通信インフラストラクチャ１０９は、イーサネット（登録商標）若しくは同様のネットワークまたはアプリケーションの転送速度要求を満足する任意の好適な物理的通信インフラストラクチャを含み得る。通信インフラストラクチャ１０９は、コンピューティングシステム１００のコンポーネントを含むコンポーネントを相互接続するための機能を含む。

【００５５】

この例において、オペレーティングシステム１０８は、システム１００のハードウェアコンポーネントを管理する機能と、共通サービスを提供するための機能とを含む。様々な実施形態において、オペレーティングシステム１０８は、ＣＰＵ１０２上で実行し、共通サービスを提供する。これらの共通サービスは、例えば、ＣＰＵ１０２内での実行のためにアプリケーションをスケジューリングすることと、フォールト管理と、インタラプトサービスと、他のアプリケーションの入力および出力を処理することと、を含む。

【００５６】

いくつかの実施形態において、オペレーティングシステム１０８は、例えばインタラプトコントローラ１４８等のインタラプトコントローラによって生成されたインタラプトに基づいて、適切なインタラプトハンドリングルーチンを呼び出す。例えば、オペレーティングシステム１０８は、ページフォールト・インタラプトを検出すると、関連するページをメモリ１０６にロードし始め、且つ、対応するページテーブルを更新するために、インタラプトハンドラを呼び出す。

【００５７】
オペレーティングシステム１０８は、オペレーティングシステムにより管理されるカーネル機能を通して、ハードウェア部品に対するアクセスが仲介されることを確保することによって、システム１００を保護する機能を含み得る。事実、オペレーティングシステム１０８は、アプリケーション１１１等のアプリケーションが、ＣＰＵ１０２上でユーザスペースにおいて実行されることを確保する。オペレーティングシステム１０８は、アプリケーション１１１が、ハードウェアにアクセスするためにオペレーティングシステムにより提供されるカーネル機能および／または入出力機能を呼び出すことも確保する。
【００５８】
例として、アプリケーション１１１は、ＣＰＵ１０２上でも実行されるユーザ計算を実行するための様々なプログラムまたはコマンドを含む。このような統一コンセプトによって、選択されたコマンドをＡＰＤ１０４上での処理対象としてＣＰＵ１０２からシームレスに送ることが可能になる。この統一ＡＰＤ／ＣＰＵフレームワークにおいて、アプリケーション１１１からの入力／出力要求は、対応するオペレーティングシステム機能を通じて処理される。
【００５９】
１つの例において、ＫＭＤ１１０は、ＣＰＵ１０２、ＣＰＵ１０２上で実行されるアプリケーションまたは他のロジックが、ＡＰＤ１０４の機能を呼び出し得るアプリケーションプログラミングインタフェース（ＡＰＩ）を実装する。例えば、ＫＭＤ１１０は、ＣＰＵ１０２からのコマンドを、コマンドバッファ１２５にエンキューし得る。なお、ＡＰＤ１０４は、このコマンドバッファ１２５からコマンドを続けて取得することとなる。加えて、ＫＭＤ１１０は、ＡＰＤ１０４上で実行されるプロセスのスケジューリングを、ＳＷＳ１１２とともに実行する。ＳＷＳ１１２は、例えば、ＡＰＤ上で実行されるプロセスの優先度リストを保持するためのロジックを含み得る。
【００６０】
本発明の他の実施形態において、ＣＰＵ１０２上で実行するアプリケーションは、コマンドをエンキューするときに、ＫＭＤ１１０を完全にバイパスし得る。
【００６１】
いくつかの実施形態において、ＳＷＳ１１２は、ＡＰＤ１０４上で実行されるプロセスのアクティブリスト１５２を、メモリ１０６に保持する。ＳＷＳ１１２は、アクティブリスト１５２におけるプロセスのうち、ハードウェアのＨＷＳ１２８により管理される一部を選択する。各プロセスをＡＰＤ１０４上で実行することに関する情報は、ＣＰＵ１０２からプロセス制御ブロック（ＰＣＢ：ｐｒｏｃｅｓｓ　ｃｏｎｔｒｏｌ　ｂｌｏｃｋ）１５４を通して、ＡＰＤ１０４に伝えられる。
【００６２】
アプリケーション、オペレーティングシステムおよびシステムソフトウェアのための処理ロジックは、マスクワーク／フォトマスクの生成を通して最終的に製造プロセスを構成することで、本明細書において説明される本発明の態様を具体化するハードウェア装置を生成することを可能にするための、例えば、Ｃ言語等のプログラム言語および／またはＶｅｒｉｌｏｇ、ＲＴＬ等のハードウェア記述言語もしくはネットリストにおいて指定されるコマンドを含み得る。
【００６３】
当業者は、コンピューティングシステム１００が、図１Ａにおいて示されるコンポーネントよりも多数または少数のコンポーネントを含み得ることを、本明細書を読むことで理解するであろう。例えば、コンピューティングシステム１００は、１つ以上の入力インターフェースと、不揮発性ストレージと、１つ以上の出力インターフェースと、ネットワークインターフェースと、１つ以上のディスプレイまたはディスプレイインターフェースと、を含み得る。
【００６４】
図１Ｂは、図１Ａにおいて示されるＡＰＤ１０４のより詳細な例示を示す実施形態であ

る。図１Ｂにおいて、ＣＰ１２４は、ＣＰパイプライン１２４ａ，１２４ｂ，１２４ｃを含み得る。ＣＰ１２４は、図１Ａにおいて示されるコマンドバッファ１２５から入力として提供されるコマンドリストを、処理するように構成され得る。図１Ｂの典型的な動作において、ＣＰ入力０（１２４ａ）は、コマンドをグラフィックスパイプライン１６２に駆動することを担当する。ＣＰ入力１および２（１２４ｂおよび１２４ｃ）は、コマンドを計算パイプライン１６０に伝える。ＨＷＳ１２８の動作を制御するためのコントローラ機構１６６も提供される。

【００６５】

図１Ｂにおいて、グラフィックスパイプライン１６２は、本明細書において順序化パイプライン１６４と称される、１セットのブロックを含み得る。例えば、順序化パイプライン１６４は、頂点グループ変換器（ＶＧＴ：ｖｅｒｔｅｘ　ｇｒｏｕｐ　ｔｒａｎｓｌａｔｏｒ）１６４ａと、プリミティブアセンブラ（ＰＡ：ｐｒｉｍｉｔｉｖｅ　ａｓｓｅｍｂｌｅｒ）１６４ｂと、スキャンコンバータ（ＳＣ：ｓｃａｎ　ｃｏｎｖｅｒｔｅｒ）１６４ｃと、シェーダエクスポート・レンダーバック・ユニット（ＳＸ／ＲＢ：ｓｈａｄｅｒ－ｅｘｐｏｒｔ，ｒｅｎｄｅｒ－ｂａｃｋ　ｕｎｉｔ）１７６とを含む。順序化パイプライン１６４内の各ブロックは、グラフィックスパイプライン１６２内の異なる段階のグラフィックス処理を表し得る。順序化パイプライン１６４は、固定機能ハードウェアパイプラインであり得る。本発明の精神および範囲に含まれ得る他の実装を用いることも可能である。

【００６６】

わずかな量のデータが、入力としてグラフィックスパイプライン１６２に提供されるが、このデータは、グラフィックスパイプライン１６２からの出力として提供される回数分だけ増幅されることとなるであろう。グラフィックスパイプライン１６２は、ＣＰパイプライン１２４ａから受け取ったワークアイテムグループ内の範囲にわたってカウントするためのＤＣ１６６も含む。ＤＣ１６６を通して提示された計算作業は、グラフィックスパイプライン１６２と準同期している。

【００６７】

計算パイプライン１６０は、シェーダＤＣ１６８，１７０を含む。ＤＣ１６８，１７０のそれぞれは、ＣＰパイプライン１２４ｂ，１２４ｃから受け取ったワークグループ内の計算範囲にわたってカウントするように構成されている。

【００６８】

図１Ｂにおいて示されるＤＣ１６６，１６８，１７０は、入力範囲を受け取り、入力範囲をワークグループに分割し、次いでこれらのワークグループをシェーダコア１２２に伝える。

【００６９】

グラフィックスパイプライン１６２は、一般に固定機能パイプラインであるため、その状態を保存および復元することは困難であり、そのためグラフィックスパイプライン１６２は、コンテキストスイッチが困難である。したがって、ほとんどの場合、本明細書において論じられるコンテキストスイッチは、グラフィックス処理におけるコンテキストスイッチに関係しない。例外は、シェーダコア１２２におけるグラフィックス作業であり、これはコンテキストスイッチされ得る。

【００７０】

シェーダコア１２２は、グラフィックスパイプライン１６２および計算パイプライン１６０により共有され得る。シェーダコア１２２は、汎用プロセッサであり、ウェーブフロントを実行するように構成されている。

【００７１】

１つの例において、計算パイプライン１６０内の全てのワークは、シェーダコア１２２内で処理される。シェーダコア１２２は、プログラム可能なソフトウェアコードを実行し、多様な形態のデータ（例えば、状態データ）を含む。しかし、計算パイプライン１６０は、処理対象ワークをグラフィックスパイプライン１６２に送らない。グラフィックスパ

イプライン１６２内におけるワーク処理が完了した後、この完了したワークを、レンダーバックユニット１７６を通じて処理する。レンダーバックユニット１７６は、デプスおよび色の計算を行った後に、この最終結果を、グラフィックスメモリ１３０に書き込む。

【００７２】

以下に記載するように、本発明は、ソフトウェア、ハードウェア、ファームウェアおよび／または図示のエンティティの多数の異なる実施形態において実行することが可能であることが当業者にとって明らかである。本発明を実行するためのハードウェアの特殊制御を用いた実際のソフトウェアコードは、本発明を限定しない。従って、本明細書に記載された詳細に鑑みれば、実施形態の改変および変更が可能であるとの理解の下、本発明の動作挙動について説明する。

【００７３】

さらに、当業者であれば理解するように、（上記したような）コンピュータで読み出し可能なコード（例えば、汎用プログラミング言語（例えば、ＣまたはＣ＋＋）、Ｖｅｒｉｌｏｇ　ＨＤＬ、ＶＨＤＬ、ＡｌｔｅｒａＨＤＬ（ＡＨＤＬ）などを含むハードウェア記述言語（ＨＤＬ）または他の利用可能なプログラミングおよび／または回路図入力ツール（例えば、回路入力ツール））の利用を通じて、本発明の多様な実施形態のシミュレーション、合成および／または製造を遂行することが部分的に可能である。コンピュータで読み出し可能なコードは、任意の公知のコンピュータにおいて利用可能な媒体（例えば、半導体、磁気ディスク、光学ディスク（例えば、ＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ））内に配置することもできるし、あるいはコンピュータデータ信号としてコンピュータで利用可能な（例えば、読み出し可能な）伝送媒体（例えば、搬送波または他の任意の媒体（例えば、デジタル媒体、光学媒体、またはアナログ媒体））内に埋め込むことも可能である。

【００７４】

従って、上記コードは、通信ネットワーク（例えば、インターネットおよびイントラネット）を通じて送信することができる。上述のシステムおよび技術によって達成される機能および／または上述のシステムおよび技術によって提供される構造は、プログラムコードに具現化されているコア（例えば、ＡＰＤコアおよび／またはＣＰＵコア）として表現され、集積回路製造の一部としてハードウェアに変換され得ることが理解される。

【００７５】

本発明の実施形態により、プログラマは、ＣＰＵとＡＰＤとの間のデータ処理移動をシームレスに行うアプリケーションを書くことが可能になり、両者の最高の特性が得られるという利益を受けることが可能になる。統一された単一プログラミングプラットフォームによって、並列処理を利用する言語、フレームワークおよびアプリケーションの開発のための強固な基盤を得ることが可能になる。

【００７６】

本発明の実施形態により、プログラマは、ＣＰＵとＡＰＤとの間のデータ処理移動をシームレスに行うアプリケーションをプログラマが書くことが可能になり、両者の最高の特性が得られるという利益を受けることが可能になる。統一された単一プログラミングプラットフォームにより、並列処理を利用する言語、フレームワークおよびアプリケーションの開発のための強固な基盤を得ることが可能になる。

【００７７】

図２は、ＡＰＤとＣＰＵとの間のシスコール要求についての最適化された通信プロセスの例示的ブロック図２００である。ブロック図２００は、ウェーブフロント１３６と、ＳＩＭＤベクトル２０８と、キュー２１０とを含む。

【００７８】

ウェーブフロント１３６は、シェーダコア１２２によって順次処理される。各ウェーブフロントは、複数のワークアイテム２０４を含む。各ワークアイテム２０４には、処理すべきタスクまたは上記タスクの一部が割りあてられている。シェーダコア１２２は、ウェーブフロント１３６内のワークアイテム２０４を、並列に、且つ、同一の１組の命令と共に処理する。その結果、ウェーブフロント１３６内の各ワークアイテム２０４は、シスコ

ールをＣＰＵ１０２に同時に発行し得る。
【００７９】
ＡＰＤが、各ワークアイテムからのシスコール要求をＣＰＵに個別に送る構成である従来のシステムとは対照的に、ＡＰＤ１０４は、ＳＩＭＤベクトル２０６を用いて要求を送るため、シスコール要求を単一データ構造としてまとめることが可能になる。ＳＩＭＤベクトル２０６は、ＳＩＭＤエレメント２０８を含む。各ＳＩＭＤエレメントは、シスコールデータ構造を含む。シスコールデータ構造は、関数セレクタパラメータ（特定のシスコール要求）と、引数リストと、シスコール要求結果をＡＰＤ１０４へ返送するためのメモリ空間とを含む。本明細書では、一実施形態として例示的なシスコールデータ構造について説明する。
【００８０】
ＯＳを必要とするプロセスがワークアイテム２０４から要求された場合には、ＡＰＤ１０４は、各ワークアイテム２０４からのシスコール要求を、対応するＳＩＭＤエレメント２０８に記憶する。例えば、図１において、ワークアイテムＷＩ１は、シスコールＳＣ１を、ＳＩＭＤエレメント２０８に記憶し、ワークアイテムＷＩ１は、シスコールＳＣ２を、別のＳＩＭＤエレメント２０８に記憶する。ＡＰＤ１０４は、各ワークアイテム２０４からのシスコール要求の種類を、関数セレクタパラメータ内に保存する。また、ＡＰＤ１０４は、必要であれば、引数リストを引数リスト部に挿入する。さらに、ＡＰＤ１０４は、複数のウェーブフロント１３６のワークアイテムからのシスコールを、１つのＳＩＭＤベクトル２０６に記憶し得る。
【００８１】
キュー２１０は、高優先度の公的メモリキューである。キューは、（ファーストインファーストアウト（ＦＩＦＯ）原理に従って動作する。公的キューは、ＣＰＵ１０２およびＡＰＤ１０４プロセッサが認識することが可能なキューである。すなわち、先行してキューに入れられたワークロードは、先行してキューから取り外される。また、当業者であれば、キューデータ構造を用いた例は例示目的のためであり、限定的なものではなく、他のデータ構造も利用可能であることを理解するであろう。
【００８２】
ＡＰＤ１０４は、ＳＩＭＤベクトル２０６と共にキュー２１０にエンキューする。ＡＰＤ１０４がＳＩＭＤベクトル２０６をキューに入れた後、一実施形態において、ＡＰＤは、ＣＰＵ１０２がＳＩＭＤベクトル２０６を処理する（すなわち、ＳＩＭＤベクトル２０６を受信し、内部に保存されているシスコールを処理し、各シスコール結果をＡＰＤ１０４へ送信する）まで停止して待機する。別の実施形態において、ＡＰＤ１０４がキュー２１０をエンキューした後、ＡＰＤ１０４は、ウェーブフロントの状態をメモリ１０６中に保存し、別のウェーブフロントの処理を開始する。ＡＰＤ１０４は、処理完了を示す信号をＣＰＵ１０２から受信すると、元のウェーブフロント１３６をメモリ１０６から取り出し、処理を回復させる。
【００８３】
ＣＰＵ１０２は、高優先度キューから受信したタスクを、他のプロセスよりも優先して処理する。よって、ＣＰＵ１０２は、高優先度キュー（例えば、キュー２１０）から要求を受信すると、現在のプロセスを保存し、上記受信した要求を処理する。本明細書中に記載される高優先度公的キューの例は、例示的なものであり、限定的なものではなく、当業者であれば、他のメモリ保存構造も利用可能であることを理解するであろう。
【００８４】
ＣＰＵ１０２は、ＳＩＭＤベクトル２０６をキュー２１０から取り出し、ＳＩＭＤエレメント２０８の処理を開始する。ＣＰＵ１０２は、ＯＳを呼び出し、各ＳＩＭＤエレメント２０８内の関数セレクタパラメータに記憶されたシスコール要求の処理を開始する。また、ＣＰＵ１０２は、必要であれば、ＳＩＭＤエレメント２０８に記憶された引数リストを読み出す。ＣＰＵ１０２は、各シスコール要求を完了した後に、その結果を、各ＳＩＭＤエレメント２０８に割りあてられたメモリアドレスに書き込む。

【００８５】
ＣＰＵ１０２は、全てのＳＩＭＤエレメント２０８の処理を完了した後に、一実施形態において、ＳＩＭＤベクトル２０６をキュー２１０上に配置し、ＳＩＭＤベクトル２０６をＡＰＤ１０４に返送する。典型的には、ＣＰＵ１０２は、ＳＩＭＤベクトル２０６を、ＡＰＤ１０４が認識することが可能なメモリキュー２１０上に配置する。
【００８６】
別の実施形態において、ＣＰＵ１０２は、ＳＩＭＤベクトル２０６の処理を完了すると、セマフォ機構を用いて、信号をＡＰＤ１０４に送る。当業者であれば、セマフォ機構を用いることにより、ＡＰＤ１０４が、処理要求されたシスコールをＣＰＵ１０２が完了するのを待機しているときに、他のウェーブフロントを処理する事態が無くなることを理解するであろう。
【００８７】
ＡＰＤ１０４は、ＳＩＭＤベクトル２０６をキューから取り外すか、あるいはシスコールが処理された旨の信号をＣＰＵ１０２から受信した後に、上記要求されたシスコールの結果を用いて、ウェーブフロント１３６の処理を開始する。ＡＰＤ１０４が、プロセスＳＩＭＤベクトル２０６をＣＰＵが処理するのを待機しつつ、別のウェーブフロントを処理することが可能な実施形態において、ＡＰＤ１０４は、ウェーブフロント１３６をＡＰＤメモリ１３０から取り出した後に、処理を継続する。
【００８８】
シスコールの一例として、メモリに対する要求（例えば、ｍａｌｌｏｃ（）関数）がある。ｍａｌｌｏｃ（）要求は、特定の処理または関数用のメモリをシステムメモリ１０６に割りあてる。ＡＰＤ１０４は、ｍａｌｌｏｃ（）要求を処理することができない。なぜならば、ＡＰＤ１０４は、ＯＳへのアクセスを有していないからである。そのため、ＡＰＤ１０４は、ｍａｌｌｏｃ（）要求用のシスコールを、ＣＰＵ１０２に送る。
【００８９】
ＡＰＤ１０４は、ウェーブフロント１３６内のワークアイテム２０４がメモリを要求した場合に、ｍａｌｌｏｃ（）要求を発行する。従来のシステムの場合、ＡＰＤが個別のｍａｌｌｏｃ（）要求を各ワークアイテムからＣＰＵに送る従来のシステムとは異なり、ＡＰＤ１０４は、ウェーブフロント１３６内のワーキングアイテム２０４ごとのｍａｌｌｏｃ（）要求を含む１つのＳＩＭＤベクトル２０６を、ＣＰＵ１０２に送る。ＡＰＤ１０４は、ｍａｌｌｏｃ（）要求に必要な情報を、対応するＳＩＭＤエレメント２０８にワークアイテムごとに記憶する。上記必要な情報は、関数セレクタと、引数リストと、空パラメータとを含む。上記関数セレクタは、ｍａｌｌｏｃ（）関数に対するメモリアドレスである。上記引数リストは、ＣＰＵ１０２が各ワークアイテム２０４に割りあてることが必要なメモリサイズを含む。ＣＰＵ１０２は、上記割りあてられた空間のアドレスを、上記空パラメータに記憶する。
【００９０】
各ワークアイテムが、各シスコールの処理に必要なｍａｌｌｏｃ（）パラメータを含んだ後に、ＡＰＤ１０４は、本明細書に記載のように、ＳＩＭＤベクトル２０６をキュー２１０上に配置する。ＣＰＵ１０２は、ＳＩＭＤベクトル２０６をキュー２１０から取り出して、ＳＩＭＤエレメント２０８の処理を開始する。ＣＰＵ１０２が、ＳＩＭＤベクトル２０６内のｍａｌｌｏｃ（）要求を処理すると、ＣＰＵ１０２は、ＯＳに対して１つのコールを発行する。その後、ＣＰＵ１０２は、上記コールのワークアイテム２０４ごとのメモリを、ＯＳに割りあてる。その後、ＣＰＵ１０２は、ＳＩＭＤエレメント２０８内のワークアイテム２０４ごとに割りあてられたメモリ空間に、アドレスを記憶する。ＣＰＵ１０２は、全てのシスコール要求を完了した後に、ＳＩＭＤベクトル２０６をＡＰＤ１０４に返送する。
【００９１】
ＳＩＭＤエレメント２０８は、シスコールをＣＰＵ１０２へ送るための複数の構造を含む。一実施形態において、各ＳＩＭＤエレメント２０８は、関数セレクタパラメータと、

引数リストと、シスコールの結果とを記憶するためのデータ構造を含み得る。非限定的な例において、例示的なデータ構造を以下に示す。

```
struct MyTask {
  MyPtr _myCodePtr
            myCPUCodePtr : pointer to code (e.g., x86 binary format)
            myAPDCodePtr :
                //GPR usage in kernel
                //LDS required by kernel
                //Pointer to code (e.g., shader binary format)
                //other parameters
  MyPtr _myDataPtr :
            myExecRange:
                //Global grid dimensions
                //Local grid dimensions
            myArgSize
            myArgs {(variable size)}
  MyNotification
      //Notification mechanism
}
```

【００９２】

上記のＭｙＴａｓｋ構造は、ＡＰＤ１０４上の命令処理のためのＭｙＰｔｒｍｙＡＰＤＣｏｄｅＰｔｒポインタと、ＣＰＵ１０２上の命令処理のためのＭｙＰｔｒｍｙＣＰＵＣｏｄｅＰｔｒポインタと、データポインタであるｍｙＰｔｒ＿ｍｙＤａｔａＰｔｒとを含む。ワークアイテム２０４が、ＣＰＵ１０２からのシスコールを要求すると、ｍｙＡＰＤＣｏｄｅＰｔｒポインタおよびｍｙＣＰＵＣｏｄｅＰｔｒポインタが、特定のシスコール機能のメモリアドレスをポイントする。上記ｍｔＤａｔａＰｔｒポインタは、引数リスト用のパラメータと、各シスコール結果を含むメインメモリ１０６内のメモリアドレスへのポインタとを含む。

【００９３】

さらに、ＭｙＴａｓｋ構造は、ＭｙＮｏｔｉｆｉｃａｔｉｏｎ機構を含む。ＡＰＤ１０４は、上記通知機構を用いて、処理を必要とするＭｙＴａｓｋがキュー１１０に存在していることを、ＣＰＵ１０２に通知する。同様に、ＣＰＵ１０２は、ＭｙＮｏｔｉｆｉｃａｔｉｏｎを用いて、ＣＰＵ１０２によるシスコール処理が完了したことを、ＡＰＤ１０４に通知する。

【００９４】

図３は、ＳＩＭＤベクトル２０６を用いてシスコール要求を処理するシステム１００の例示的なフローチャート２００である。ステップ３０２において、ウェーブフロント１３６内のワークアイテム２０４が、ＣＰＵ１０２を用いた処理を必要とするシスコールを要求した場合に、ＡＰＤ１０４は、ＳＩＭＤベクトル２０６を初期化する。ステップ３０４において、各ワークアイテム２０４は、本明細書に記載のように、シスコール要求の処理に必要な情報を、対応するＳＩＭＤエレメント２０８に記憶する。ステップ３０６において、ＡＰＤ１０４は、ＳＩＭＤベクトル２０６をキュー２１０にエンキューする。ステップ３０８において、ＣＰＵ１０２は、ＳＩＭＤベクトル２０６をキュー２１０から取り出す。ＣＰＵ１０２は、ＳＩＭＤベクトル２０６をキューから取り出した後に、ＯＳを呼び出し、各ＳＩＭＤエレメント２０８内のシスコールの処理を開始する。

【００９５】

ステップ３１０において、ＣＰＵ１０２は、各シスコールの結果をＳＩＭＤエレメント２０８に書き込む。当業者であれば、ステップ３１０を、ステップ３０８と共に実行することが可能であることを理解するであろう。ステップ３１２において、ＣＰＵ１０２は、シスコール処理が完了したことをＡＰＤ１０４に通知する。一実施形態において、ＣＰＵ

１０２は、ＡＰＤ１０４が認識することが可能なキュー２１０を用いて、ＳＩＭＤベクトル２０６をＡＰＤ１０４に返送する。別の実施形態において、ＣＰＵ１０２は、セマフォを用いて、信号をＡＰＤ１０４に送信する。ステップ３１４において、ＡＰＤ１０４は、ＳＩＭＤベクトル２０６をキュー２１０から取り出し、ウェーブフロント１３６の処理を継続する。

【００９６】

本発明の様々な態様は、ソフトウェア、ファームウェア、ハードウェアまたはこれらの組み合わせによって実装することが可能である。例えば、図３のフローチャート３００によって示す方法を、図１の統一コンピューティングシステム１００において実行することが可能である。本発明の多様な実施形態について、本例の統一コンピューティングシステム１００を用いて説明する。他のコンピュータシステムおよび／またはコンピュータアーキテクチャを用いて本発明を実行するための方法が、当業者にとって明らかである。

【００９７】

本文書において、「コンピュータプログラム媒体」および「コンピュータで利用可能な媒体」とは、例えばリムーバブルストレージユニットやハードディスクドライブなどの媒体を主に指す。また、コンピュータプログラム媒体およびコンピュータで利用可能な媒体は、メモリ（例えば、システムメモリ１０６およびグラフィックスメモリ１３０）を指す。上記メモリは、メモリ半導体（例えば、ＤＲＡＭ）であり得る。これらのコンピュータプログラム製品は、ソフトウェアを統一コンピューティングシステム１００に提供するための手段である。

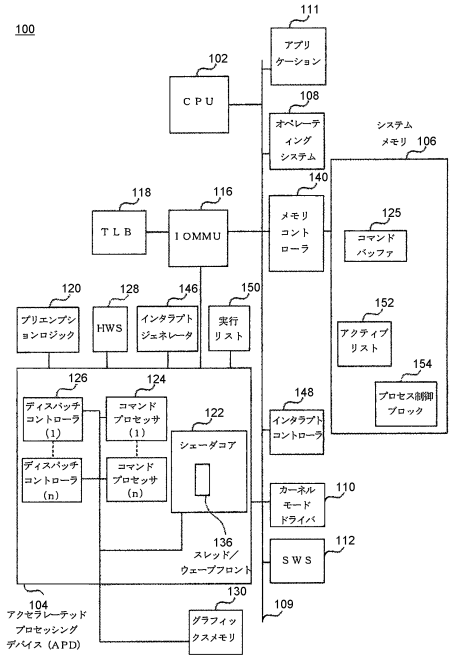
【００９８】

本発明は、任意のコンピュータで利用可能な媒体に記憶されたソフトウェアを含むコンピュータプログラム製品にも関する。このようなソフトウェアが１つ以上のデータ処理デバイスにおいて実行された場合、データ処理デバイス（単数または複数）は、本明細書にて記載したように動作し、コンピューティングデバイス（例えば、ＡＳＩＣまたはプロセッサ）の合成および／または製造を許容して、本明細書に記載の本発明の実施形態の実行が可能となる。本発明の実施形態において、現在公知であるかまたは将来において公知となるコンピュータで利用可能な媒体、またはコンピュータで読み出し可能な媒体がすべて用いられる。コンピュータで利用可能な媒体の例を非限定的に挙げると、一次記憶デバイス（例えば、任意の種類のランダムアクセスメモリ）、二次記憶デバイス（例えば、ハードドライブ、フロッピー（登録商標）ディスク、ＣＤＲＯＭ、ＺＩＰディスク、テープ、磁気記憶デバイス、光学記憶デバイス、ＭＥＭＳ、ナノ技術記憶デバイス）ならびに通信媒体（例えば、有線通信ネットワークおよび無線通信ネットワーク、ローカルエリアネットワーク、広域ネットワーク、イントラネット）がある。

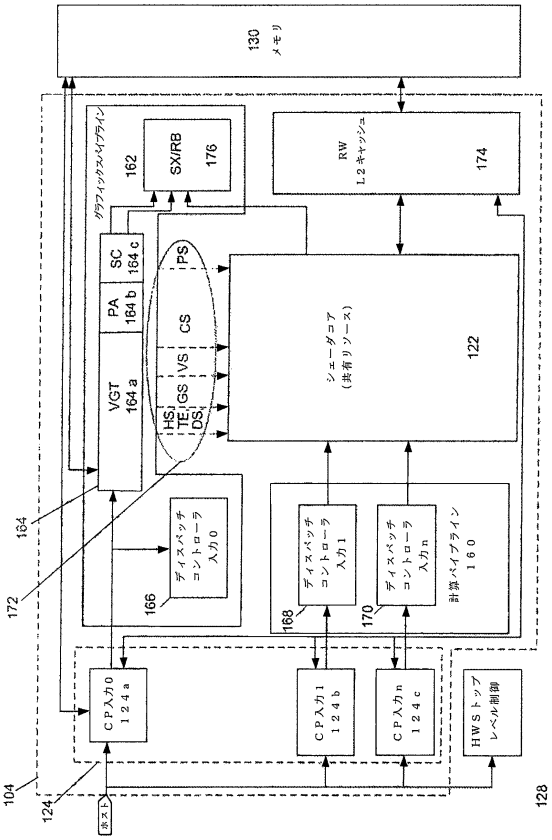
【００９９】

本発明の多様な実施形態について上記において説明してきたが、これらの実施形態はひとえに例示的なものであり、制限的なものではないことが理解されるべきである。当業者であれば、これらの実施形態において、形態および詳細における多様な変更が（添付の特許請求の範囲に記載のような本発明の意図および範囲から逸脱することなく）可能であることを理解する。本発明はこれらの例に限定されないことが理解されるべきである。本発明は、本明細書中に記載のように動作する要素に適用することが可能である。よって、本発明の範囲は、上記した例示的実施形態のいずれによっても限定されるべきではなく、以下の特許請求の範囲およびその均等物によって規定されるべきものである。

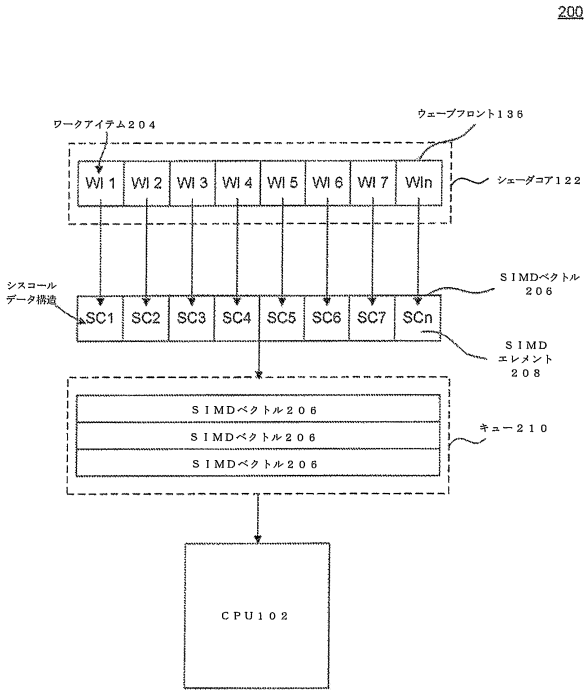
【図１Ａ】


100
111
アプリケーション
102
CPU
108
オペレーティングシステム
システムメモリ
106
140
メモリコントローラ
118
TLB
116
IOMMU
125
コマンドバッファ
120
プリエンプションロジック
128
HWS
146
インタラプトジェネレータ
150
実行リスト
152
アクティブリスト
154
プロセス制御ブロック
126
ディスパッチコントローラ(1)
ディスパッチコントローラ(n)
124
コマンドプロセッサ(1)
コマンドプロセッサ(n)
122
シェーダコア
148
インタラプトコントローラ
110
カーネルモードドライバ
112
SWS
136
スレッド／ウェーブフロント
104
アクセラレーテッドプロセッシングデバイス（APD）
130
グラフィックスメモリ
109

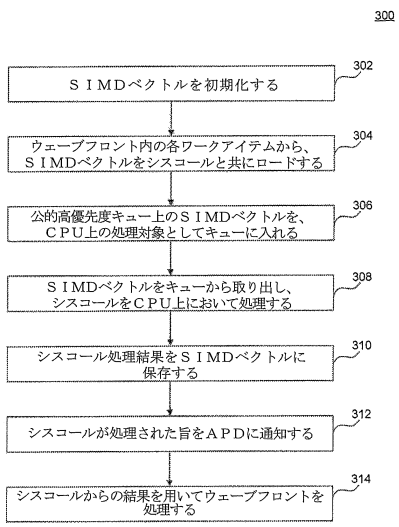
【図１Ｂ】


130
メモリ
グラフィックスパイプライン
162
SX/RB
176
RW
L2キャッシュ
174
SC
164c
PA
164b
VGT
164a
PS
CS
VS
GS
HS
TE
DS
シェーダコア
（共有リソース）
122
ディスパッチコントローラ入力0
166
ディスパッチコントローラ入力1
168
ディスパッチコントローラ入力n
170
計算パイプライン
160
164
172
CP入力0
124a
CP入力1
124b
CP入力n
124c
HWSトップレベル制御
104
124
ホスト
128

【図２】


200
ワークアイテム204
ウェーブフロント136
WI 1
WI 2
WI 3
WI 4
WI 5
WI 6
WI 7
WIn
シェーダコア122
シスコールデータ構造
SIMDベクトル206
SC1
SC2
SC3
SC4
SC5
SC6
SC7
SCn
SIMDエレメント208
SIMDベクトル206
SIMDベクトル206
SIMDベクトル206
キュー210
CPU102

【図３】


300
SIMDベクトルを初期化する
302
ウェーブフロント内の各ワークアイテムから、SIMDベクトルをシスコールと共にロードする
304
公的高優先度キュー上のSIMDベクトルを、CPU上の処理対象としてキューに入れる
306
SIMDベクトルをキューから取り出し、シスコールをCPU上において処理する
308
シスコール処理結果をSIMDベクトルに保存する
310
シスコールが処理された旨をAPDに通知する
312
シスコールからの結果を用いてウェーブフロントを処理する
314

フロントページの続き

早期審査対象出願

(74)代理人 100162156
弁理士 村雨 圭介
(72)発明者 ベンジャミン トーマス サンダー
アメリカ合衆国 ７８７３５ テキサス州、オースティン、メディスン クリーク ５７０１
(72)発明者 マイケル ヒューストン
アメリカ合衆国 ９５０１４ カリフォルニア州、クパチーノ、コロンバス アベニュー ２１３３０
(72)発明者 ニュートン チェン
アメリカ合衆国 ９４０８８ カリフォルニア州、サニーベイル、ワン・エイ・エム・ディ・プレイス
(72)発明者 キース ローリー
アメリカ合衆国 ９８０１１ ワシントン州、ボセル、ノースイースト １９７番 ストリート １０９１０

合議体
審判長 高木 進
審判官 辻本 泰隆
審判官 石井 茂和

(56)参考文献 特開２００９－１４０４９１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl., ＤＢ名)
G06F9/46-9/54