(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5379810号
(P5379810)
(45)発行日　平成25年12月25日(2013.12.25)　(24)登録日　平成25年10月4日(2013.10.4)

(51)Int.Cl.　F I
G06T 15/00 (2011.01)　G06T 15/00 100A

請求項の数 13 (全 16 頁)

(21)出願番号	特願2010-540662 (P2010-540662)
(86)(22)出願日	平成20年12月23日 (2008.12.23)
(65)公表番号	特表2011-508338 (P2011-508338A)
(43)公表日	平成23年3月10日 (2011.3.10)
(86)国際出願番号	PCT/US2008/014011
(87)国際公開番号	WO2009/085264
(87)国際公開日	平成21年7月9日 (2009.7.9)
審査請求日	平成23年12月22日 (2011.12.22)
(31)優先権主張番号	61/016,746
(32)優先日	平成19年12月26日 (2007.12.26)
(33)優先権主張国	米国 (US)
(31)優先権主張番号	12/341,028
(32)優先日	平成20年12月22日 (2008.12.22)
(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】グラフィックスパイプラインのための効率的な状態管理

(57)【特許請求の範囲】
【請求項１】
状態依存処理を実行するためのコンピュータベースのシステムであって、
コマンドストリームにおける１以上のコマンドを受け取り、前記コマンドストリームにおける１以上のグローバルコンテキストイベントに応答する１以上のグローバル状態を管理するように構成されたコマンドプロセッサと、
前記コマンドストリームにおける前記１以上のコマンドを受け取り、前記コマンドストリームにおける１以上のブロックコンテキストイベントに応答する１以上のブロック状態のそれぞれを管理する複数の処理ブロックであって、それぞれが前記１以上のグローバル状態および当該１以上の処理ブロックのブロック状態に基づいてデータストリームにおけるデータに対して処理を実行する複数の処理ブロックと、
少なくとも一つの前記処理ブロックに記憶された１以上のブロック状態の数を示すブロック状態値を含む、少なくとも一つの処理ブロックのためのカウンタを管理するとともに、前記ブロック状態値が閾値を超えたときに前記コマンドストリームにおける前記１以上のコマンドを待機させるように構成されたインタフェースブロックとを備える、
コンピュータベースのシステム。
【請求項２】
前記複数の処理ブロックのうち第１の処理ブロックは、前記１以上のブロック状態の第１の数を管理するように構成され、前記複数の処理ブロックのうち第２の処理ブロックは、前記１以上のブロック状態の第２の数を管理するように構成され、前記第１の数は前記

第２の数と異なる、請求項１のコンピュータベースのシステム。
【請求項３】
前記コマンドプロセッサは、前記１以上のグローバル状態の数を示すグローバル状態値を用いてグローバル状態カウンタを管理するとともに、前記グローバル状態値が他の閾値を超えたときに前記コマンドストリームにおける前記１以上のコマンドを待機させるように構成されている、請求項１のコンピュータベースのシステム。
【請求項４】
前記コマンドプロセッサは、前記複数の処理ブロックにおける最後の処理ブロックからの空き信号を受け取った後に、前記１以上のグローバル状態のうち第１のグローバル状態を再割り当てするように構成されている、請求項１のコンピュータベースのシステム。
【請求項５】
特定用途向け集積回路（ＡＳＩＣ）における状態依存処理を実行するための方法であって、
コマンドストリームにおける１以上のグローバルコンテキストイベントに応答する、前記ＡＳＩＣの１以上のグローバル状態を管理するステップと、
前記コマンドストリームにおける１以上のブロックコンテキストイベントに応答する、複数の処理ブロックにおける各処理ブロックの１以上のブロック状態を個別に管理するステップと、
前記１以上のグローバル状態およびそれぞれの処理ブロックの前記１以上のブロック状態に基づいて、前記複数の処理ブロックの各処理ブロックにおいてデータストリームにおけるデータに対する処理を実行するステップと、
前記１以上のグローバル状態の数を示すグローバル状態値が第１の閾値を超えた場合、前記１以上のブロック状態の数を示すブロック状態値が第２の閾値を超えた場合、またはこれらの組み合わせの場合のうち少なくとも一つが生じたときに、前記コマンドストリームにおける１以上のコマンドを待機させるステップとを含む、
方法。
【請求項６】
前記１以上のブロック状態を管理するステップは、
前記複数の処理ブロックのうち第１の処理ブロックにおける前記１以上のブロック状態の第１の数と、前記複数の処理ブロックのうち第２の処理ブロックにおける前記１以上のブロック状態の第２の数とを管理するステップを含み、
前記第１の数は前記第２の数と異なる、請求項５の方法。
【請求項７】
前記１以上のグローバル状態を管理するステップは、
前記複数の処理ブロックにおける最後の処理ブロックからの空き信号を受け取った後に、前記１以上のグローバル状態のうち第１のグローバル状態を再割り当てするステップを含む、請求項５の方法。
【請求項８】
ソフトウエアにおいて具現化された特定用途向け集積回路（ＡＳＩＣ）を含む有形的コンピュータ可読記録媒体であって、前記ＡＳＩＣは状態依存処理を実行し、
前記ＡＳＩＣは、
コマンドストリームにおける１以上のコマンドを受け取り、前記コマンドストリームにおける１以上のグローバルコンテキストイベントに応答する１以上のグローバル状態を管理するように構成されたコマンドプロセッサと、
前記コマンドストリームにおける前記１以上のコマンドを受け取り、前記コマンドストリームにおける１以上のブロックコンテキストイベントに応答する１以上のブロック状態のそれぞれを管理する複数の処理ブロックであって、それぞれが前記１以上のグローバル状態および当該１以上の処理ブロックのブロック状態に基づいてデータストリームにおけるデータに対して処理を実行する複数の処理ブロックと、
少なくとも一つの前記処理ブロックに記憶された１以上のブロック状態の数を示すブロ

ック状態値を含む、少なくとも一つの処理ブロックのためのカウンタを管理するとともに、前記ブロック状態値が閾値を超えたときに前記コマンドストリームにおける前記１以上のコマンドを待機させるように構成されたインタフェースブロックとを備える、
有形的コンピュータ可読記録媒体。
【請求項９】
前記複数の処理ブロックのうち第１の処理ブロックは、前記１以上のブロック状態の第１の数を管理するように構成され、前記複数の処理ブロックのうち第２の処理ブロックは、前記１以上のブロック状態の第２の数を管理するように構成され、前記第１の数は前記第２の数と異なる、請求項８の有形的コンピュータ可読記録媒体。
【請求項１０】
前記コマンドは、前記１以上のグローバル状態の数を示すグローバル状態値を用いてグローバル状態カウンタを管理するとともに、前記グローバル状態カウンタが他の閾値を超えたときに前記コマンドストリームにおける前記１以上のコマンドを待機させるように構成されている、請求項８の有形的コンピュータ可読記録媒体。
【請求項１１】
前記コマンドプロセッサは、前記複数の処理ブロックにおける最後の処理ブロックからの空き信号を受け取った後に、前記１以上のグローバル状態のうち第１のグローバル状態を再割り当てするように構成されている、請求項８の有形的コンピュータ可読記録媒体。
【請求項１２】
前記ＡＳＩＣは、ハードウエア記述言語ソフトウエアにおいて具現化される、請求項８の有形的コンピュータ可読記録媒体。
【請求項１３】
前記ＡＳＩＣは、ベリログハードウエア記述言語ソフトウエアおよびＶＨＤＬハードウエア記述言語ソフトウエアの一方において具現化される、請求項８の有形的コンピュータ可読記録媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は一般的にコンピュータシステムにおいて実行されるコンピュータの動作に関するものである。
【背景技術】
【０００２】
グラフィックス処理ユニット（ＧＰＵ）は、グラフィックス処理タスクを実行するために特別に設計された特定用途向け集積回路（ＡＳＩＣ）である。ＧＰＵは、例えば、ビデオゲームアプリケーション等のエンドユーザアプリケーションによって要求されるグラフィックス処理タスクを実行することができる。このような例においては、エンドユーザアプリケーションとＧＰＵの間にいくつものソフトウエアのレイヤが存在する。エンドユーザアプリケーションはアプリケーションプログラミングインタフェース（ＡＰＩ）と交信する。ＡＰＩは、ＧＰＵに依存するフォーマットでよりはむしろ標準化されたフォーマットでエンドユーザアプリケーションがグラフィックスデータおよびコマンドを出力することを可能にする。マイクロソフトコープ(Microsoft Corp.)により開発されたＤｉｒｅｃｔＸ（登録商標）およびシリコングラフィックスインク(Silicon Graphics, Inc.)により開発されたＯｐｅｎＧＬ（登録商標）を含め、何種類かのＡＰＩが商業的に入手可能である。ＡＰＩはドライバと交信する。ドライバはＡＰＩから受け取った標準コードをＧＰＵが理解する固有のフォーマットの命令に変換する。ドライバは典型的にはＧＰＵの製造業者によって書かれる。ＧＰＵはそうしてドライバからの命令を実行する。
【０００３】
多くのＧＰＵは命令を実行するためのグラフィックスパイプラインを含む。グラフィックスパイプラインは、ある命令の異なるステップに同時に作用する複数の処理ブロックを含む。パイプライン方式は、その命令を実行するのに必要となる複数ステップ間での並列

処理をＧＰＵが上手く利用することを可能にする。結果として、ＧＰＵは短時間により多くの命令を実行することができる。

【０００４】

グラフィックスパイプラインの出力はグラフィックスパイプラインの状態に依存する。グラフィックスパイプラインの状態は、状態パッケージ－即ちグラフィックスパイプラインによりローカルで記憶されるコンテキスト固有定数(context-specific constants)（例えばテクスチュアハンドル(texture handles)、シェーダ定数(shader constants)、変換マトリクス等）に基いて更新される。コンテキスト固有定数はローカルで維持されるので、ブラフィックスパイプラインはそれらに素早くアクセスすることができる。

【発明の概要】

【発明が解決しようとする課題】

【０００５】

グラフィックスパイプラインにより維持される状態パッケージの数は、ＧＰＵが接続されるＡＰＩに依存する。従来の標準的なＡＰＩに付随する状態パッケージは、比較的少数のレジスタ、例えば８レジスタに記憶することができる。しかしながら、標準的なＡＰＩと異なり、より新しいＡＰＩ（例えばＤｉｒｅｃｔＸ（登録商標）１０）は、パイプラインの特定の局面に関して比較的多数の頻繁なコンテキスト切り替えを必要とする。これら頻繁なコンテキスト切り替えに関連する状態パッケージの数は、標準的なグラフィックスパイプラインにより維持される比較的少数のレジスタではサポートすることができない。

【０００６】

より新しいＡＰＩに付随する多数の状態パッケージに対処するための見込みのある解決方法は、グラフィックスパイプラインによってサポートされる状態パッケージの数を単純に増やすことである。しかし、この解決方法では、付加的な状態パッケージに対処するために付加的なレジスタが必要になるであろうから、ダイ(die)領域を著しく増大させることとなろう。また、もし状態パッケージの数がパイプラインの記憶容量を超えれば、グラフィックスパイプラインは待機させられるであろうから、タイミングの問題を新たに生み出すこととなろう。

【０００７】

もう一つの見込みのある解決方法は、ソフトウエアを用いて状態パッケージの数の増加を補償しようという試みであろう。例えば、ドライバまたはエンドユーザアプリケーションは、ＧＰＵに送られる作業(work)を並べ替えて状態変更の数を減らす（状態変更毎に送られる作業を増やす）ことを試みるかもしれない。しかし、この解決方法には少なくとも２つの欠点がある。第１に、この解決方法は、幾らかの作業付加を伴うだけである（幾つかは余分な状態変更を本質的に有する）。第２に、この解決方法は、入力トランザクション(transactions)を検索し並び替えるＣＰＵの作業負荷を著しく増大する。

【０００８】

以上に鑑みて、効率的な状態管理システムを提供する方法およびシステムが必要となっている。

【課題を解決するための手段】

【０００９】

本発明は、効率的な状態管理システムおよびその応用を提供することにより、上述した必要性を満たす。

【００１０】

本発明のある態様によると、状態依存処理を実行するためのコンピュータベースのシステムが提供される。コンピュータベースのシステムはコマンドプロセッサおよび複数の処理ブロックを含む。コマンドプロセッサはコマンドストリームにおけるコマンドを受け取りコマンドストリームにおけるグローバルコンテキストイベントに応答するグローバル状態を管理する。複数の処理ブロックはコマンドストリームにおけるコマンドを受け取り、コマンドストリームにおけるブロックコンテキストイベントに応答するそれぞれのブロック状態を管理する。複数の処理ブロックはグローバル状態およびそれぞれの処理ブロック

のブロック状態に基づいてデータストリームにおけるデータに対する処理を実行する。
【００１１】
本発明の他の態様によると、ＡＳＩＣにおける状態依存処理を実行するための方法が提供される。方法は、コマンドストリームにおけるグローバルコンテキストイベントに応答する、ＡＳＩＣのグローバル状態を管理するステップと、コマンドストリームにおけるブロックコンテキストイベントに応答する、複数の処理ブロックにおける各処理ブロックのブロック状態を個別に管理するステップとを含む。複数の処理ブロックは、グローバル状態およびそれぞれの処理ブロックのブロック状態に基づいてデータストリームにおけるデータに対する処理を実行する。
【００１２】
本発明の更なる態様によると、ソフトウエア内で具現化されたＡＳＩＣを含む有形的コンピュータ可読記録媒体であって、ＡＳＩＣは状態依存処理を実行する有形的コンピュータ可読記録媒体が提供される。ＡＳＩＣはコマンドプロセッサおよび複数の処理ブロックを含む。コマンドプロセッサはコマンドストリームにおけるコマンドを受け取り、コマンドストリームにおけるグローバルコンテキストイベントに応答するグローバル状態を管理する。複数の処理ブロックはコマンドストリームにおけるコマンドを受け取り、コマンドストリームにおけるブロックコンテキストイベントに応答するそれぞれのブロック状態を管理する。複数の処理ブロックはグローバル状態およびそれぞれの処理ブロックのブロック状態に基づいてデータストリームにおけるデータに対する処理を実行する。
【００１３】
本発明の更なる特徴および利点は、本発明の種々の実施形態の構成および動作と同様に、添付図面を参照して以下に詳細に説明される。尚、本発明はここで説明される特定の実施形態に限定されない。かかる実施形態は例示目的のみのために提示される。更なる実施形態は、ここに含まれる教示に基いて、関連技術をも含めた当業者にとって明白なものとなろう。
【００１４】
添付図面は、ここに組み込まれると共に明細書の一部をなし、本発明を例示しそして、その記述と共に、本発明の原理を説明し且つ関連技術を含めた当業者が本発明を作り且つ使用することを可能にするために更に役立つ。
【図面の簡単な説明】
【００１５】
【図１】本発明の実施形態に従ってＧＰＵを含むシステム例を示すブロック図。
【図２】図１のＧＰＵの特徴を示すブロック図。
【図３】図１のＧＰＵに含まれるグラフィックスパイプラインの詳細を示すブロック図。
【図４】本発明の実施形態に従ってグローバル状態およびブロック状態の管理を示す表を表す。
【発明を実施するための形態】
【００１６】
本発明の特徴および利点は、図面と併せて以下の詳細な説明からより明らかなものとなろう。全図を通して同様の参照文字は対応する要素を特定する。図面において、同様な参照番号は一般的に、同一な、機能的に類似な、および／または構造的に類似な要素を示す。ある要素が最初に現れる図面は、その要素に対応する参照番号の最も左の桁数で示される。
【００１７】
Ｉ．　概要
本発明は複雑なＡＳＩＣのための効率的な状態管理システムおよびその応用を提供する。以下の詳細な説明において、「１つの実施形態」、「実施形態」、「実施形態例」等への言及は、記述された実施形態が特定の特徴、構造、または特性を含んでよいことを示すが、必ずしもすべての実施形態がその特定の特徴、構造、または特性を含まなくてよい。また、そのような語句は必ずしも同一の実施形態を参照しているとは限らない。更に、あ

る実施形態に関連して特定の特徴、構造、または特性が記述されている場合、明確な説明の有無にかかわらず他の実施形態に関連してそのような特徴、構造、または特性をとることは、当業者が知り得ると言うことができる。

【００１８】

限定ではなく例示を目的として、本発明の実施形態に従う効率的な状態管理システムを、ＧＰＵのグラフィックスパイプラインのグローバル状態およびブロック状態に基づいて描画コマンドのストリームを実行するＧＰＵに関してここに説明する。しかしながら、そのような効率的な状態管理システムは、それぞれのシステムの状態に基いてコマンドのストリームを実行する他の種類のシステムにも適用可能であることが理解される。ここに含まれる記述に基いて、関連技術を含めた当業者であれば、そのような他の種類のシステムにおいて本発明の実施形態をどのように実施するかを理解するであろう。

【００１９】

上述したように、本発明の実施形態に従う効率的な状態制御は、グラフィックスパイプラインのグローバル状態およびブロック状態を独立して管理する。グローバル状態は所定数の可能なグローバル状態、例えば８つの可能なグローバル状態を含む。ブロック状態は各々所定数の可能なブロック状態、例えば２、４、８、１６、３２、６４、または他のなんらかの数の可能なブロック状態を含む。可能なブロック状態の数は各ブロックで異なっていてよい。グローバル状態は、例えば場面における太陽光の方向のような頻繁には変化しない場面内の特徴に関する描画定数に対応していてよい。ブロック状態は、例えば画面に描かれた草が風になびく方向のような頻繁に変化する場面内の特徴に関する描画定数に対応していてよい。

【００２０】

ここで用いられる「ブロック」という用語は、ＡＳＩＣに含まれる処理モジュール、ＣＰＵの実行パイプライン、および／またはＧＰＵのグラフィックスパイプラインを参照する。そのような処理モジュールは、限定はされないが、算術論理ユニット、乗算／除算ユニット、浮動小数点ユニット、色バッファ、頂点シェーダ(vertex shader)、画素シェーダ、ｚ－バッファ、または関連技術を含めた当業者に明らかであろうなんらかの他の処理モジュールを含んでよい。

【００２１】

本発明の実施形態に従うグラフィックスパイプラインはグローバル状態およびブロック状態を独立して管理するので、そのようなグラフィックスパイプラインは、従来のグラフィックスパイプラインと比較してより多くの異なる値に基づく描画コマンドを実行することができる。例えば、従来のグラフィックスパイプラインは典型的には最大で８つの異なる値に基く描画コマンドを実行することができる。対照的に、本発明の実施形態に従うグラフィックスパイプラインは最大で例えば２５６の異なる値に基づく描画コマンドを実行することができる。

【００２２】

グローバル状態はコマンドプロセッサ（ＣＰ）により管理される。ＣＰはグローバル状態ダーティービットおよびグローバル状態識別（ＩＤ）ビットを保持する。グローバル状態ダーティービットはグラフィックスパイプラインのグローバル状態が更新されるべきかどうかを示す。グローバル状態ＩＤビットはグラフィックスパイプラインのグローバル状態を示す。グローバル状態ＩＤビットはグローバルコンテキストイベントの後にインクリメントされる。グローバルコンテキストイベントは、グローバル状態が更新される予定であることを示す入力（データ）ストリーム（例えばデータパケット）におけるイベントである。グローバル状態ダーティービットは描画要求(draw call)でクリアされる。

【００２３】

ブロック状態はグラフィックスパイプラインのそれぞれのブロックにより管理される。グラフィックスパイプラインの各ブロックはブロック状態ダーティービットおよびブロック状態ＩＤビットを保持する。ブロック状態ダーティービットはあるブロックのブロック状態が更新されるべきかどうかを示す。あるブロックに対応するブロック状態ＩＤビット

はそのブロックのブロック状態を示す。あるブロックのブロック状態ＩＤビットは、次のブロック状態更新がそのブロックに対応していれば、ブロックコンテキストイベントでインクリメントされる。ブロックコンテキストイベントは、ブロック状態が更新される予定であることを示す入力（データ）ストリーム（例えばデータパケット）におけるイベントである。ある実施形態においては、グラフィックスパイプライン内のあるブロックは、次のブロック状態更新がそのブロックに対応していることを、次のブロック状態更新のアドレス範囲に基いて決定する。グローバル状態ダーティービットと同様に、ブロック状態ダーティービットは描画要求でクリアされる。

【００２４】

効率的な状態管理システムはＧＰＵを含むコンピュータシステム内で表現され得る。そのような効率的な状態管理システムのハードウエア実装例およびその動作について、更に詳細に以下説明する。そのような効率的な状態管理システムのソフトウエア実装についても説明する。

【００２５】

II.　効率的な状態管理システムのハードウエア実装例

上述したように、本発明の実施形態に従う効率的な状態管理システムは、コマンドのストリームを実行するＡＳＩＣ内に実装されてよい。限定でなく例示を目的として、以下更に詳細にそのような効率的な状態管理システムをコンピュータシステムに含まれるグラフィックスパイプライン例に関して説明する。

【００２６】

Ａ．コンピュータシステム例の概要

図１は実施形態に従うコンピュータシステム１００のブロック図である。システム１００は、中央処理ユニット（ＣＰＵ）１０２およびグラフィックス処理ユニット（ＧＰＵ）１１０を含み、随意的にコプロセッサ１１２を含んでよい。加えて、コンピュータシステム１００は、ＣＰＵ１０２、ＧＰＵ１１０、およびコプロセッサ１１２によりアクセスされてよいシステムメモリ１０４を含む。ＧＰＵ１１０およびコプロセッサ１１２は、バス１１４を介してＣＰＵ１０２およびシステムメモリと交信する。バス１１４は、コンピュータシステムにおいて用いられるどのような種類のバスであってもよく、その種類は、限定はされないが、周辺部品(peripheral component)インタフェース（ＰＣＩ）バス、加速化(accelerated)グラフィックスポート（ＡＧＰ）バス、およびＰＣＩ高速(Express)（ＰＣＩＥ）バスを含む。ＧＰＵ１１０およびコプロセッサ１１２は、ある種の特別な機能を、ＣＰＵ１０２がそれらをソフトウエアにおいて実行し得るよりは通常は高速に実行することにより、ＣＰＵ１０２を支援する。コプロセッサ１１２は、限定はされないが、浮動小数点コプロセッサ、ＧＰＵ、ネットワーキングコプロセッサ、その他の種類のコプロセッサや、関連技術をも含めた当業者にとって明らかであろうようなプロセッサから構成されてよい。

【００２７】

システム１００は、第１のローカルメモリ１０６および第２のローカルメモリ１０８を更に含む。第１のローカルメモリ１０６は、ＧＰＵ１１０に接続されるとともにバス１１４にも接続されている。第２のローカルメモリ１０８は、コプロセッサ１１２に接続されるとともにバス１１４にも接続されている。第１および第２のローカルメモリ１０６および１０８は、それぞれＧＰＵ１１０およびコプロセッサ１１２によって利用可能であり、特定のデータ（例えば頻繁に使用されるようなデータ）への、そのデータがシステムメモリ１０４に記憶されているとした場合に可能なよりも高速なアクセスを提供する。

【００２８】

ある実施形態では、ＧＰＵ１１０およびコプロセッサ１１２は、ＣＰＵ１０２とともに命令を並行して復号化し、それらのために意図されている当該命令のみを実行する。もう一つの実施形態では、ＣＰＵ１０２は、ＧＰＵ１１０およびコプロセッサ１１２のために企図された命令をそれぞれのコマンドバッファに送る。

【００２９】

例えば、図２は、ＧＰＵ１１０のために企図された命令をＣＰＵ１０２がコマンドバッファ２０２に送る実施形態を説明しているブロック図を示す。コマンドバッファ２０２は、例えばシステムメモリ１０４内に位置していてよく、あるいはバス１１４に接続された別個のメモリであってよい。図２に示されるように、ＧＰＵ１１０はスケジューラ２０４およびグラフィックスパイプライン２０６を含む。スケジューラ２０４はコマンドバッファ２０２から命令を読み出す(retrieves)。スケジューラ２０４は、後ほど更に詳細に説明されるように、それら命令をグラフィックスパイプライン２０６内のコマンドプロセッサに転送する。

【００３０】

Ｂ．グラフィックスパイプライン例

図３は、本発明の実施形態に従うグラフィックスパイプライン２０６の詳細を説明しているブロック図である。図３に示されるように、グラフィックスパイプライン２０６は、コマンドプロセッサ（ＣＰ）３１０、レジスタバスマネージャ（ＲＢＭ）３２０、データバストランスファー（ＤＢＴ）ブロック３３０、ブロックＡ３４０ａ、ブロックＢ３４０ｂ、ブロックＣ３４０ｃ、およびブロックＤ３４０ｄを含む。これら要素の各々については、後ほど更に詳細に説明する。

【００３１】

ＣＰ３１０は、コマンドストリームにおける複数コマンドを例えばスケジューラ２０４から受け取る。それらコマンドは、描画コマンド、グローバル状態更新、およびブロック状態更新を含む。ある実施形態では、各コマンドはレジスタアドレスを含む。この実施形態では、あるコマンドに含まれるアドレスが所定のアドレス範囲に含まれるかどうかに基づいて、そのコマンドは描画コマンド、グローバル状態更新、またはブロック状態更新として構文解析(parsed)が可能である。本発明は、しかし、この実施形態に限定されない。関連技術をも含めた当業者であれば、本発明の精神および範囲から逸脱することなしに、コマンドを構文解析する他のメカニズムが使用され得ることを理解するであろう。

【００３２】

ＣＰ３１０はまた、２つのダーティビット(dirty bits)：（１）グローバル状態更新が既に受信されているかどうかを示すグローバリーマネージドダーティー(globally managed dirty)（ＧＭＤ）ビット３１２；および（２）ブロック状態更新が既に受信されているかどうかを示すブロックマネージドダーティー(block managed dirty)（ＢＭＤ）ビット３１４を保持する。ＧＭＤビット３１２およびＢＭＤビット３１４は、例えば、論理値１としてオンすることによりダーティー（セット）をオンにする。これらビットの両方は描画コマンドによりクリアされる。

【００３３】

ＣＰ３１０はまた、グラフィックスパイプライン２０６の複数ブロック３４０のレジスタに書かれたグローバル状態の数を管理する。グローバル状態の数を管理するために、ＣＰ３１０は、グラフィックスパイプライン２０６のグローバル状態を示すグローバル状態識別(identification)（ＩＤ）を管理する。グローバル状態ＩＤは、ゼロからグラフィックスパイプライン２０６によって保持されるグローバル状態の最大数、例えば８グローバル状態までの範囲を取る。ＧＰＵ１１０の電源が入るとグローバル状態ＩＤはゼロにセットされる。後で更に詳細に説明されるように、グローバル状態ＩＤは、グローバルコンテキストイベントが受け取られ且つ新たなグローバル状態がグラフィックスパイプライン２０６に書き込まれた後にインクリメントされる。グローバル状態ＩＤは、グラフィックスパイプライン２０６内の最後のブロック（例えばブロックＤ３４０ｄ）がＣＰ３１０にパルスを送り返した後に解除される（そして次いで再生されてよい）。例えば、ＣＰ３１０はグローバル状態ＩＤをゼロから１にインクリメントしてよいが、グローバル状態ＩＤゼロに対応するコンテキストが完了したことを示すパルスを最後のブロックがＣＰ３１０に送り返すまでは、グローバル状態ＩＤゼロは再生することができない。

【００３４】

ＣＰ３１０はＲＢＭ３２０に接続されている。ＣＰ３１０は、描画コマンド、グローバ

ル状態更新およびブロック状態更新をＲＢＭ３２０に書き込む。後で更に詳細に説明されるように、グラフィックスパイプライン２０６に割り当てられたグローバル状態の数は、ＣＰ３１０によって管理される。

【００３５】

ＲＢＭ３２０はＤＢＴブロック３３０に接続されている。ＲＢＭ３２０は、描画コマンド、グローバル状態更新およびブロック状態更新をＤＢＴブロック３３０に書き込む。後で更に詳細に説明されるように、グラフィックスパイプライン２０６に割り当てられたブロック状態の数は、ＲＢＭ３２０およびそれぞれのブロック３４０によって集合的に管理される。

【００３６】

ＤＢＴブロック３３０は、データバス３８０およびレジスタバス（コントロールバス）３９０を介してブロックＡ３４０ａ、ブロックＢ３４０ｂ、ブロックＣ３４０ｃ、およびブロックＤ３４０ｄに接続されている。データバス３８０は、それぞれのブロック３４０に入力（データ）ストリームを供給する。ブロックＡ３４０ａ乃至ブロックＤ３４０ｄは、デイジーチェーン方式(daisy chain manner)によりデータバス３８０に接続されている。つまり、ブロックＡ３４０ａは、入力ストリームのデータに対する処理を実行し、次いでデータをブロックＢ３４０ｂに送り、ブロックＢ３４０ｂは、入力ストリームのデータに対する処理を実行し、次いでデータをブロックＣ３４０ｃに送り、ブロックＣ３４０ｃは、入力ストリームのデータに対する処理を実行し、次いでデータをブロックＤ３４０ｄに送り、ブロックＤ３４０ｄは、入力ストリームのデータに対する処理を実行し、次いで当該データをグラフィックスパイプライン２０６の出力として出力する。与えられたグローバルコンテキストのためのデータに対する処理をブロックＤ３４０ｄが実行した後、ブロックＤ３４０ｄは、その与えられたグローバルテキストに付随するグローバル状態ＩＤが再割り当てされてよいことを示すパルスをＣＰ３１０に送り返す。データバス３８０は、入力（データ）ストリームを供給することに加えて、グローバルコンテキストイベントおよびブロックコンテキストイベントをそれぞれのブロック３４０に供給する。ある実施形態では、データバス３８０は１００ビット幅である。

【００３７】

レジスタバス３９０は、描画コマンド、グローバル状態更新およびブロック状態更新を含むコマンドをそれぞれのブロック３４０に供給する。描画コマンドは、その描画コマンドのために割り当てられたグローバル状態およびブロック状態に基づいて、データバス３８０から受け取ったデータに対する描画を実行する旨をそれぞれのブロック３４０に指示する。

【００３８】

グローバル状態更新は、それぞれのブロックによって維持されているローカルレジスタに書き込まれる。図３に示されるように、ブロックＡ３４０ａはグローバル状態更新をレジスタファイルＡ３４２内のレジスタに書き込み、ブロックＢ３４０ｂはグローバル状態更新をレジスタファイルＢ３５２内のレジスタに書き込み、ブロックＣ３４０ｃはグローバル状態更新をレジスタファイルＣ３６２内のレジスタに書き込み、そしてブロックＤ３４０ｄはグローバル状態更新をレジスタファイルＤ３７２内のレジスタに書き込む。

【００３９】

ブロック状態更新は、グラフィックスパイプライン２０６内の特定の一つまたは複数のブロックにのみ書き込まれる。例えば、レジスタバス３９０上のある一つのブロック状態更新は、ブロックＢ３４０ｂのみを対象とされていてよい。この例では、ブロックＢ３４０ｂのみがそのブロック状態更新をレジスタファイルＢ３５２のレジスタに書き込むであろう。他のすべてのブロックは、それぞれが保持しているローカルレジスタファイル内のレジスタに当該ブロック状態更新を書き込まないであろう。ある実施形態では、ブロック状態更新はアドレスを含む。この実施形態では、各ブロック３４０は、そのローカルレジスタファイルに当該ブロック状態更新を書き込むべきかどうかを当該ブロック状態更新のアドレスに基いて決定する。しかしながら、関連技術を含めた当業者であれば、ある特定

のブロックがそのローカルレジスタにブロック状態更新を書き込むべきかどうかを決定するための他のメカニズムが本発明の精神および範囲から逸脱することなしに使用され得ることを理解するであろう。

【００４０】

グラフィックスパイプライン２０６の構成を考慮しつつ、その動作を更に詳細に以下に説明する。

【００４１】

III. 効率的な状態管理システムの動作例

上述したように、グラフィックスパイプライン２０６は、グローバル状態更新およびブロック状態更新を独立して管理し、それによりグラフィックスパイプライン２０６が比較的多数の異なる描画定数に基いて描画コマンドを実行することを可能にしている。先ず、グローバル管理状態およびブロック管理状態の概要が提示される。次いで、グラフィックスパイプライン２０６に割り当てられたグローバル状態およびブロック状態の数を限定するための方法例が提示される。最後に、本発明の実施形態に従う効率的な状態管理の動作を説明するために、イベントのシーケンス例が提示される。

【００４２】

A. グローバルおよびブロック管理状態移行

ＣＰ３１０はグローバル状態更新を管理する。もし、ＣＰ３１０がコマンドストリームにおいてグローバル状態更新を検出し且つグラフィックスパイプライン２０６内で書き込みに利用可能なグローバル状態レジスタがあるならば、ＣＰ３１０は以下の機能を実行するであろう：

（i）ＧＭＤビット３１２をセット（もし未セットであれば）；

（ii）グローバルコンテキストイベント（即ち「Ｇｌｏｂａｌ＿Ｃｏｎｔｅｘｔ＿Ｄｏｎｅ」）をＲＢＭ３２０に送り、ＲＢＭ３２０はそのグローバルコンテキストイベントをデータバス３８０上に供給；

（iii）グローバル状態ＩＤビットをインクリメント；

（iv）ＣＰ３１０のＧＬＢ＿ＣＯＰＹ＿ＳＴＡＴＥレジスタに書き込み；そして

（v）グローバル状態更新をＲＢＭ３２０に送り、ＲＢＭ３２０はそのグローバル状態更新をレジスタバス３９０上に供給。

しかしながら、もし、ＣＰ３１０がコマンドストリームにおいてグローバル状態更新を検出していても、グラフィックスパイプライン２０６内に利用可能なグローバル状態レジスタが無ければ、ＧＬＢ＿ＣＯＰＹ＿ＳＴＡＴＥレジスタへの書き込みは、コンテキストが利用可能になるまで待機させられる。ＧＭＤビット３１２は、コンテキストが状態の当該セットのために既に展開されている(rolled)ことを示すために用いられる。

【００４３】

もし、ＢＭＤビット３１４がセットされていなくてＣＰ３１０がコマンドストリームにおいてブロック状態更新を検出すれば、ＣＰ３１０は、ＢＭＤビット３１４をセットし、Ｇｌｏｂａｌ＿Ｃｏｎｔｅｘｔ＿Ｄｏｎｅイベントを送り、ブロック状態更新が続くであろう。ＣＰ３１０および個々のブロック３４０により保持されたダーティービットは、状態の当該セットのためにコンテキストが既に展開されていることを示すようにセットされる。もし、利用可能なブロック状態レジスタが無ければ、目標ブロック（例えばブロックＢ３４０ｂ）は、たとえ目標ブロック（例えばブロックＢ３４０ｂ）がレジスタ更新をもはや許容し得なくとも、空き状態に戻らないように関与する。

グローバルおよびブロック管理状態移行の動作については、セクションIII. C. において提供されるシーケンス例により更に説明する。

【００４４】

B. 本発明の実施形態に従うグラフィックスパイプラインに割り当てられたグローバル状態およびブロック状態の数を管理

グラフィックスパイプライン２０６に割り当てられたグローバル状態の数は、ＣＰ３１０により管理される一方、グラフィックスパイプライン２０６に割り当てられたブロック

状態の数はＲＢＭ３２０およびそれぞれのブロック３４０により集合的に管理される。グローバル状態およびブロック状態の数を管理するためのメカニズムを更に詳細に説明する。

【００４５】

ＣＰ３１０は、コマンドおよびブロック状態更新をＲＢＭ３２０に継続的に書き込む。しかしながら、ＣＰ３１０は、割り当てられたグローバル状態の数がグラフィックスパイプライン２０６により維持されているグローバル状態の最大数よりも小さい場合にのみ、グローバル状態更新をＲＢＭ３２０に書き込む。即ち、グラフィックスパイプライン２０６に割り当てられたグローバル状態の数は、ＣＰ３１０によって管理される。

【００４６】

ＲＢＭ３２０は、コマンドおよびブロック状態更新をＤＢＴブロック３３０に継続的に書き込む。しかしながら、ＲＢＭ３２０は、割り当てられたブロック状態の数がグラフィックスパイプライン２０６のそれぞれのブロック３４０により維持されているブロック状態の最大数より小さい場合にのみブロック状態更新をＤＢＴ３３０に書き込む。ＤＢＴブロック３３０は、ブロック状態の数が最大数より小さいかどうかをそれぞれのブロック３４０から受け取った信号に基いて決定する。即ち、ＲＢＭ３２０およびそれぞれのブロック３４０が、グラフィックスパイプライン２０６に割り当てられたブロック状態更新の数を管理する。

【００４７】

上述したように、ＲＢＭ３２０およびそれぞれのブロック３４０は、グラフィックスパイプライン２０６に書き込まれたブロック状態更新の数を絞る(throttle)ために機能する。図３に示されるように、ブロック３４０は、それぞれの配線(trace)３４４、配線３５４、配線３６４、および配線３７４を介してＲＢＭ３２０に接続されている。それぞれのブロックに割り当てられているブロック状態の数が各ブロックのためのブロック状態の最大数よりも小さい限り、ＲＢＭ３２０はブロック状態更新をＤＢＴブロック３３０に書き込み、ＤＢＴブロック３３０は次いでブロック状態更新をレジスタバス３９０に書き出す。しかしながら、もし、ある一つのブロック３４０に対するブロック更新の数がそのブロックのためのブロック状態の最大数に達したならば、そのブロックは「空き(free)」信号をＲＢＭ３２０へは返さない。結果として、先に出された描画コマンドの実行が完了し且つその描画コマンドに割り当てられたブロック状態の割り当てが解除されるまで、ＲＢＭ３２０は待機することとなる。

【００４８】

Ｃ．イベントのシーケンス例

図４はイベントのシーケンス例に対してグラフィックスパイプライン２０６がどのように応答するのかを表す表４００を示している。図４に示されるように、表４００は８列を含む。第１列４１０は、受けたイベントのシーケンスを数え上げる行番号を列挙している。第２列４２０は、ＣＰ３１０から受け取りレジスタバス３９０に書き出されたイベント例を列挙している。第３列４３０は、データバス３８０上に供給された入力（データ）ストリームを列挙している。第４列４４０は、ＣＰ３１０のＧＭＤビット３１２の状態を列挙している。第５列４５０は、ＣＰ３１０のＢＭＤビット３１４の状態を列挙している。第６列４６０はグローバル状態ＩＤを列挙しており、グローバル状態ＩＤはグラフィックスパイプライン２０６のグローバル状態を示す。第７列４７０は、ブロックＡ３４０ａのブロックダーティービットおよびブロックコンテキストインジケータビットの状態を示す。第８列４８０は、ブロックＡ３４０ａのブロック状態ＩＤを示す。第７列４７０および第８列４８０は、イベントのシーケンス例にブロックＡ３４０ａがどのように応答するかを説明している。関連技術を含む当業者であれば、ブロックＢ３４０ｂ、ブロックＣ３４０ｃ、およびブロックＤ３４０ｄがブロックＡ３４０ａと同様に動作することは理解するであろう。第１列４１０内の行番号により数え上げられたイベントのシーケンスを以下では更に詳細に説明する。

【００４９】

表４００の第１列４１０を参照すると、１行目は、ＧＭＤビット３１２およびＢＭＤビット３１４がそれぞれ第４および第５列４４０および４５０に示されるようにダーティーをオン（例えば論理値１）にされることを説明している。また、第７列４７０は、ブロックＡ３４０ａのダーティービットもまたダーティーをオン（例えば論理値１）にされることを示している。２行目および３行目では、ブロック状態(Block State)およびグローバル状態(Global State)がそれぞれ受け取られる。これらの状態の書き込みは状態になんら変化をもたらさない。

【００５０】

４行目では、Ｄｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｏｎｔｅｘｔ＿（Ｃｔｘ）＿０命令が受信される。この描画コマンドは、第４列４４０、第５列４５０、および第７列４７０にそれぞれ示されるように、ＧＭＤビット３１２、ＢＭＤビット３１４、およびブロックＡ３４０ａにより管理されるダーティービットをクリアする。描画コマンドは、データバス３８０上で供給されたデータに対して実行される。第３列４３０は、当該データが頂点／画素ベクトル(vertex/pixel (vtx/pix) vectors)を含むことを示している。

【００５１】

５行目では、Ｇｌｏｂａｌ＿Ｃｏｎｔｅｘｔ＿Ｄｏｎｅコマンドが受け取られる。このコマンドは、グローバルコンテキスト０が完了したことを示している。結果として、グローバルコンテキスト０が完了したことをそれぞれのブロック３４０に知らせるために、Ｃｔｘ＿Ｄｏｎｅ＿Ｅｖｅｎｔがデータバス３８０上に供給される。ある実施形態では、Ｃｔｘ＿Ｄｏｎｅ＿Ｅｖｅｎｔはデータパケットである。

【００５２】

６行目では、Ｗｒｉｔｅ　ＧＢＬ＿ＣＯＰＹ＿ＳＴＡＴＥコマンドが受け取られる。このコマンドは、新たなグローバル状態がグラフィックスパイプライン２０６のそれぞれのレジスタに書き込まれる予定であることを示している。このコマンドの結果として、第４列４４０に示されるようにＧＭＤビット３１２がセットされる。また、第６列４６０に示されるように、グローバル状態ＩＤが１にインクリメントされ、グラフィックスパイプライン２０６が新たなグローバル状態に入っていることを表示する。

【００５３】

７行目では、グローバルコンテキスト１に対応するグローバル状態(Global State)更新が受け取られ、グラフィックスパイプライン２０６のそれぞれのレジスタに書き込まれる。

【００５４】

８行目では、Ｄｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿１コマンドが受け取られる。この描画コマンドは、前もって設定された唯一のダーティービット－即ち、第４列４４０に示されるＧＭＤビット３１２をクリアする。４行目で受け取ったＤｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿０と同様に、Ｄｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿１コマンドはデータバス３８０上で伝えられた頂点／画素ベクトルに対して実行される。

【００５５】

９行目では、Ｂｌｏｃｋ＿Ｃｏｎｔｅｘｔ＿Ｄｏｎｅイベントが受け取られる。このコマンドは、ブロックコンテキストの一つが完了したことを示す。結果として、ＢＭＤビット３１４がセットされ（第５列４５０に示されるように）、ブロックコンテキストイベント（Ｂｌｋ＿Ｃｔｘ＿Ｄｏｎｅ＿Ｅｖｅｎｔ）がデータバス３８０上に供給される。第７列４７０に示されるように、ブロックＡ３４０ａはこのイベントを検出するが、そのダーティービットを当初はセットしない。ブロックＡ３４０ａがそのダーティービットを当初セットしないのは、このブロックイベントがブロックＡ３４０ａに対応するものであるかどうかがまだ確定していないからである。このブロックイベントは、グラフィックスパイプライン２０６内の任意のブロック３４０に対応し得る。

【００５６】

１０行目では、Ｂｌｏｃｋ＿Ｓｔａｔｅ＿（Ｉｎｃｌｕｓｅｓ　ＢｌｋＡ　Ｓｔａｔｅ

）コマンドが受け取られ、このコマンドは、当該ブロックイベントがブロックＡ３４０ａに対応するものであることを示している。結果として、ブロックＡ３４０ａは、それぞれ第７列４７０および第８列４８０に示されるように、そのダーティービットをセットすると共にその状態ＩＤをインクリメントする。上述したように、ある実施形態では、コマンドストリームにおけるコマンドはアドレスを含むので、ブロックＡ３４０ａは、ブロック状態(Block_State)のアドレスがブロックＡ３４０ａに対応する所定のアドレス範囲に属するという理由で、そのブロック状態(Block_State)がレジスタファイルＡ３４２内のレジスタに書き込まれるべきことを決定する。

【００５７】

１１行目では、Ｇｌｏｂａｌ＿Ｃｏｎｔｅｘｔ＿Ｄｏｎｅ命令が受け取られる。このコマンドは、グローバルコンテキスト１が完了したことを示している。結果として、グローバルコンテキストイベント（Ｃｔｘ＿Ｄｏｎｅ＿Ｅｖｅｎｔ）がデータバス３８０上に供給され、グローバルコンテキスト１が完了したことをグラフィックスパイプライン２０６のそれぞれのブロック３４０に知らせる。

【００５８】

１２行目では、Ｗｒｉｔｅ　ＧＢＬ＿ＣＯＰＹ＿ＳＴＡＴＥコマンドが受け取られる。６行目でＷｒｉｔｅ　ＧＢＬ＿ＣＯＰＹ＿ＳＴＡＴＥコマンドが受け取られたのと同様にして、１２行目で受け取られたコマンドは、新たなグローバル状態がグラフィックスパイプライン２０６のそれぞれのレジスタに書き込まれる予定であることを示している。このコマンドの結果として、ＧＭＤビット３１２が第４列４４０に示されるようにセットされる。また、第６列４６０に示されるように、グローバル状態ＩＤが２にインクリメントされ、グラフィックスパイプライン２０６が新たなグローバル状態に入っていることを表示する。

【００５９】

１３行目では、グローバルコンテキスト２に対応するグローバル状態(Global State)更新が受け取られ、グラフィックスパイプライン２０６のそれぞれのレジスタに書き込まれる。

【００６０】

１４行目では、Ｄｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿２コマンドが受け取られる。この描画コマンドは、前もって設定されたダーティービット－即ち、それぞれ第４列４４０、第５列４５０、および第７列４７０に示されるＧＭＤビット３１２、ＢＭＤビット３１４、およびブロックＡ３４０ａにより管理されるダーティービットをクリアする。４行目で受け取ったＤｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿０および８行目で受け取ったＤｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿１と同様に、Ｄｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿２コマンドはデータバス３８０上で供給された頂点／画素ベクトルに対して実行される。

【００６１】

１５行目では、Ｂｌｏｃｋ＿Ｃｏｎｔｅｘｔ＿Ｄｏｎｅイベントが受け取られる。このコマンドは、ブロックコンテキストの一つが完了したことを示す。結果として、ＢＭＤビット３１４がセットされ（第５列４５０に示されるように）、ブロックコンテキストイベント（Ｂｌｋ＿Ｃｔｘ＿Ｄｏｎｅ＿Ｅｖｅｎｔ）がデータバス３８０上に書き出される。第７列４７０に示されるように、ブロックＡ３４０ａはこのイベントを検出するが、そのダーティービットを当初はセットしない。

【００６２】

１６行目では、Ｂｌｏｃｋ＿Ｓｔａｔｅ　（Ｎｏｔ　ａ　ＢｌｋＡ　Ｓｔａｔｅ）コマンドが受け取られ、このコマンドは、このブロックイベントがブロックＡ３４０ａに対応していないことを示している。結果として、ブロックＡ３４０ａは、それぞれ第７列４７０および第８列４８０に示されるように、そのダーティービットをセットせず且つその状態ＩＤをインクリメントしない。ある実施形態では、ブロックＡ３４０ａは、このブロック状態(Block_State)のアドレスがブロックＡ３４０ａに対応する所定のアドレス範囲に

属さないという理由で、このブロック状態(Block_State)がレジスタファイルＡ３４２内のレジスタに書き込まれるべきではないことを決定する。

【００６３】

１７行目では、別のＤｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿２コマンドが受け取られる。この描画コマンドは、前もって設定されたダーティービット－即ち、それぞれ第４列４４０、第５列４５０、および第７列４７０に示されるＧＭＤビット３１２、ＢＭＤビット３１４、およびブロックＡ３４０ａにより管理されるダーティービットをクリアする。先行して受け取った他のＤｒａｗ＿Ｉｎｉｔｉａｔｏｒコマンドと同様に、このＤｒａｗ＿Ｉｎｉｔｉａｔｏｒ＿ｉｎ＿Ｃｔｘ＿２コマンドは、データバス３８０上で伝えられた頂点／画素ベクトルに対して実行される。

【００６４】

上述したのと同様にして、追加的なイベントが受け取られ処理される。上述したイベントのシーケンス例は、例示のみを目的としており、限定のためのものではないことが理解されるべきである。

【００６５】

IV.　効率的な状態管理システムのソフトウエア実装例

ＧＰＵ１１０のハードウエア実装に加えて、ＧＰＵ１１０は、例えば、ソフトウエア（例えばコンピュータ可読プログラムコードのような）を記憶するように構成されたコンピュータ利用可能な（例えば可読な）媒体内に配置されたソフトウエアにおいて具現化されてよい。プログラムコードは、以下のような実施形態：（i）システムの機能およびここに開示された技術（例えばグローバル状態更新およびブロック状態更新を管理すること）；（ii）システムの製造およびここに開示された技術（例えばＧＰＵ１１０の製造）；（iii）システムの機能および製造並びにここに開示された技術の組み合わせを含め、本発明の実施形態の実施可能性をもたらす。例えば、このことは、一般的なプログラミング言語（例えばＣまたはＣ＋＋）、ベリログ(Verilog)ＨＤＬ、ＶＨＤＬ、アルテラ(Altera)ＨＤＬ（ＡＨＤＬ）等を含むハードウエア記述言語(hardware description languages)（ＨＤＬ）、あるいは他の利用可能なプログラミングおよび／または回路図等（schematic）キャプチャツール(capture tools)（例えば回路キャプチャツール）の使用を通じて達成され得る。

【００６６】

プログラムコードは、半導体、磁気ディスク、光学ディスク（例えばＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ）を含むコンピュータが使用可能なあらゆる既知の記録媒体内に配置されてよく、またコンピュータが使用可能な（例えば可読な）伝送媒体（例えばデジタル、光学的、またはアナログ系の媒体を含む搬送波または他のあらゆる媒体）において具現化されるコンピュータデータ信号として処置されてよい。従って、コードは、インターネットおよび将来出現するであろう同等のもの(the Internet and internets)を含む通信網を介して伝送されてよい。上述したシステムおよび技術により達成された機能および／または提供された構造は、プログラムコードにおいて具現化されるコア(core)（例えばＧＰＵコア）内に表現することができ、また集積回路の生産の一部としてハードウエアに変換されてよいことが理解される。

【００６７】

V.　結論

概要および要約の欄ではなく、詳細な説明の欄は、特許請求の範囲を解釈するために用いられることが意図されることは理解されるべきである。概要および要約の欄は、本発明者によって意図されるような本発明の１つまたは２つ以上であるがすべてではない例示的な実施形態を記述していてよく、従って、いかなる方法においても本発明および添付の特許請求の範囲を限定することを意図するものではない。

【図１】

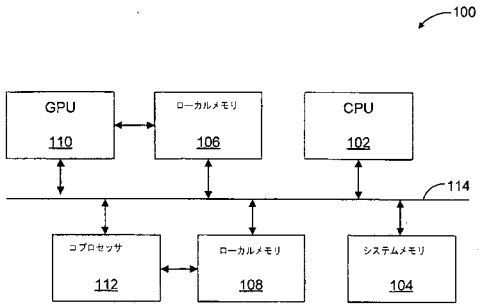

FIG. 1

【図２】

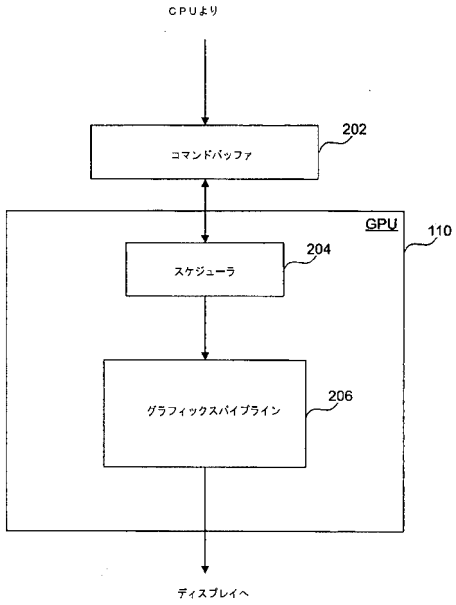

FIG. 2

【図３】

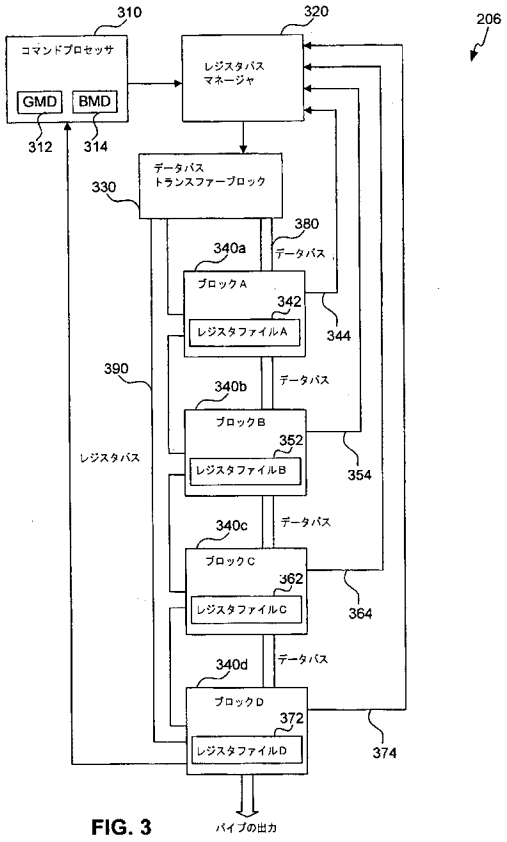

FIG. 3

【図４】

400

	410 状態書き込み	420 パイプライン（ストリーム）入力	430 GPU GMD ビット	440 GPU BMD ビット	450 グローバル状態 ID	460 ブロックA ダーティーイベント	470 ブロックA 状態 ID	480
1			1	1	0			
2	Block State		1	1	0			
3	Global State		1	1	0			
4	Draw Initiator in Ctx 0	vtx/pix vectors	0	0	0			
5	Global_Context_Done	Ctx Done Event	0	0	0			
6	Write GBL_COPY_STATE		1	0	1			
7	Global State		1	0	1			
8	Draw Initiator in Ctx 1	vtx/pix vectors	0	0	1			
9	Block_Context_Done	Blk Ctx Done Event	0	1	1			
10	Block State (includes BlkA State)			1	1			
11	Global Context_Done	Ctx Done Event	0	1	1			
12	Write GBL_COPY_STATE		1	1	2			
13	Global State		1	1	2			
14	Draw Initiator in Ctx 2	vtx/pix vectors	0	0	2			
15	Block_Context_Done	Blk Ctx Done Event	0	1	2			
16	Block State (No BlkA State)		0	1	2			
17	Draw Initiator in Ctx 2	vtx/pix vectors	0	0	2			

FIG. 4

フロントページの続き

(74)代理人 100162156
弁理士 村雨 圭介
(72)発明者 マイケル マントール
アメリカ合衆国、フロリダ州 ３２８２５、オーランド ピナールドライブ １６２０
(72)発明者 レックス エルドン マックラリー
アメリカ合衆国、フロリダ州 ３２７６５、オビエド サマーオークスコート ７５８

審査官 千葉 久博

(56)参考文献 特開２００１－２３６２２１（ＪＰ，Ａ）
特開２００１－０７６１７９（ＪＰ，Ａ）
特開平１１－３４５３４９（ＪＰ，Ａ）
特開平１０－１１１８５８（ＪＰ，Ａ）
特開平０８－３２８９５１（ＪＰ，Ａ）
米国特許出願公開第２００７／０１０３４７５（ＵＳ，Ａ１）
米国特許第０５９２０３２６（ＵＳ，Ａ）
米国特許第０６４６２７４３（ＵＳ，Ｂ１）
国際公開第２００６／０６８９８５（ＷＯ，Ａ１）
国際公開第２００６／０５１３３０（ＷＯ，Ａ１）
国際公開第０２／１０１６４９（ＷＯ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｔ １５／００－１５／８７
Ｇ０６Ｔ １３／００－１３／８０，１９／００，１９／２０
Ｇ０６Ｔ １／００－１／６０