(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6258293号
(P6258293)
(45)発行日　平成30年1月10日(2018.1.10)　(24)登録日　平成29年12月15日(2017.12.15)

(51)Int.Cl.　F I
G06T 15/00 (2011.01)　G06T 15/00 501

請求項の数 15 (全 73 頁)

(21)出願番号	特願2015-504596 (P2015-504596)
(86)(22)出願日	平成25年3月15日 (2013.3.15)
(65)公表番号	特表2015-524092 (P2015-524092A)
(43)公表日	平成27年8月20日 (2015.8.20)
(86)国際出願番号	PCT/US2013/032098
(87)国際公開番号	WO2013/151748
(87)国際公開日	平成25年10月10日 (2013.10.10)
審査請求日	平成28年2月19日 (2016.2.19)
(31)優先権主張番号	61/620,358
(32)優先日	平成24年4月4日 (2012.4.4)
(33)優先権主張国	米国 (US)
(31)優先権主張番号	61/620,340
(32)優先日	平成24年4月4日 (2012.4.4)
(33)優先権主張国	米国 (US)

(73)特許権者　595020643
クゥアルコム・インコーポレイテッド
ＱＵＡＬＣＯＭＭ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100109830
弁理士　福原　淑弘
(74)代理人　100103034
弁理士　野河　信久
(74)代理人　100075672
弁理士　峰　隆司

最終頁に続く

(54)【発明の名称】グラフィックス処理におけるパッチされたシェーディング

(57)【特許請求の範囲】
【請求項１】
グラフィックスをレンダリングする方法であって、
少なくとも１つのプリミティブをシェーディングするためのドローコールに基づいて、動作モードを決定することと、
前記決定された動作モードに基づいて、レンダリングパイプラインの第１のシェーダステージと関連付けられるセットの第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定することと、
前記決定された動作モードに基づいて、前記レンダリングパイプラインの第２の異なるシェーダステージと関連付けられるセットの第２のシェーディング操作を前記第１のシェーディング操作に付加することと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記第１のシェーディング操作に後続する前記第２のシェーディング操作を実行することと
を備える、方法。
【請求項２】
前記ドローコールは、ドローコールの第１のサブドローコールを備え、前記方法は、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、第３のシェーディング操作を実行することをさらに備える、請求項１に記載の方法。

【請求項３】
前記決定された第２の動作モードに基づいて、レンダリングパイプラインの第４のシェーダステージと関連付けられる第２のセットの第３のシェーディング操作を実行するように、グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを指定することと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、第４のシェーディング操作を実行することと
をさらに備える、請求項２に記載の方法。
【請求項４】
前記第２のシェーダステージと関連付けられる前記第２のシェーディング操作を実行することは、前記第１のシェーダステージと関連付けられる入力／出力インターフェースを維持することを備える、
請求項１に記載の方法。
【請求項５】
前記第２のシェーディング操作を実行する前に、前記第２のシェーディング操作のためにプログラムカウンタと１つまたは複数のリソースポインタとを切り替えることをさらに備える、
請求項１に記載の方法。
【請求項６】
前記第１のシェーディング操作と関連付けられる第１の命令は、前記第２のシェーディング操作と関連付けられる第２の命令に依存しないように、前記第１の命令が前記第２の命令とは独立にコンパイルされる、
請求項１に記載の方法。
【請求項７】
１つまたは複数のシステムにより生成される値のために、ローカルメモリ中の１つまたは複数の所定の位置を確保することをさらに備え、前記システムにより生成される値は、前記第１のシェーディング操作および前記第２のシェーディング操作において使用される、
請求項６に記載の方法。
【請求項８】
前記第１のシェーディング操作の結果をローカルメモリに記憶することをさらに備え、前記第２のシェーディング操作を実行することは、前記グラフィックスプロセシングユニットの外部に位置するオフチップメモリにアクセスすることなく、前記第１のシェーディング操作の前記結果に対して前記第２のシェーディング操作を実行することを備える、
請求項１に記載の方法。
【請求項９】
前記第１のシェーディング操作を実行することは、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することを備え、前記第２のシェーディング操作を実行することは、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するジオメトリシェーディング操作を実行することを備える、
請求項１に記載の方法。
【請求項１０】
前記第１のシェーディング操作を実行することは、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することを備え、前記第２のシェーディング操作を実行することは、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するハルシェーディング操作を実行することを備える、
請求項１に記載の方法。

【請求項１１】
前記第１のシェーディング操作を実行することは、頂点を生成するためにドメインシェーディング操作を実行することを備え、前記第２のシェーディング操作を実行することは、前記ドメインシェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するためにジオメトリシェーディング操作を実行することを備える、
請求項１に記載の方法。
【請求項１２】
グラフィックスをレンダリングするための装置であって、
少なくとも１つのプリミティブをシェーディングするためのドローコールに基づいて、前記少なくとも１つのプリミティブをシェーディングするための関連付けられるシェーディング動作のセットを有する動作モードを決定するための手段と、
前記決定された動作モードに基づいて、レンダリングパイプラインの第１のシェーダステージと関連付けられるセットの第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定するための手段と、
前記決定された動作モードに基づいて、前記レンダリングパイプラインの第２の異なるシェーダステージと関連付けられるセットの第２のシェーディング操作を前記第１のシェーディング操作に付加するための手段と、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記第１のシェーディング操作に後続する前記第２のシェーディング操作を実行するための手段と
を備える、装置。
【請求項１３】
前記装置は、グラフィックスをレンダリングするためのグラフィックスプロセシングユニットである、
請求項１２に記載の装置。
【請求項１４】
前記ドローコールは、ドローコールの第１のサブドローコールを備え、前記装置は、
前記ドローコールの第２のサブドローコールに基づいて、前記少なくとも１つのプリミティブをシェーディングするための関連するシェーディング操作の第２のセットを有する第２の動作モードを決定するための手段と、
前記決定された第２の動作モードに基づいて、レンダリングパイプラインの第３のシェーダステージと関連付けられる第２のセットの第３のシェーディング操作を実行するように、グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを指定するための手段と、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記第３のシェーディング操作を実行するための手段と
をさらに備える、請求項１２に記載の装置。
【請求項１５】
命令を記憶した非一時的コンピュータ可読媒体であって、前記命令は、実行されると、１つまたは複数のプロセッサに請求項１－１１のうちのいずれか一項に従った方法を実行することを行わせる、
非一時的コンピュータ可読媒体。
【発明の詳細な説明】
【関連出願】
【０００１】
[0001]本出願は、そのすべての内容全体が参照により本明細書に組み込まれる、２０１２年４月４日に出願された米国仮出願第６１／６２０，３４０号、２０１２年４月４日に出願された米国仮出願第６１／６２０，３５８号、および２０１２年４月４日に出願され

た米国仮出願第６１／６２０，３３３号の利益を主張する。
【技術分野】
【０００２】
[0002]本開示は、コンピュータグラフィックスに関する。
【背景技術】
【０００３】
[0003]視覚的提示のためのコンテンツを提供するデバイスは、一般にグラフィックスプロセシングユニット（ＧＰＵ）を含む。ＧＰＵは、コンテンツを表すピクセルをディスプレイ上にレンダリングする。ＧＰＵは、提示のための各ピクセルをレンダリングするために、ディスプレイ上の各ピクセルに対する１つまたは複数のピクセル値を生成する。
【０００４】
[0004]いくつかの例では、ＧＰＵは、グラフィックスをレンダリングするための統一されたシェーダアーキテクチャを実装することができる。そのような例では、ＧＰＵは、異なるシェーディング操作のパイプラインを実行するように、複数の同様のコンピューティングユニットを構成することができる。コンピューティングユニットは、統一されたシェーディングユニットまたは統一されたシェーダプロセッサと呼ばれ得る。
【発明の概要】
【０００５】
[0005]本開示の技法は一般に、グラフィックスレンダリングパイプラインのシェーダステージと関連付けられるシェーディング操作を実行することに関する。たとえば、グラフィックスプロセシングユニット（ＧＰＵ）は、グラフィックスレンダリングパイプラインのシェーダステージと関連付けられるシェーディング操作を実行するために１つまたは複数のシェーディングユニットを呼び出すことができる。本開示の態様によれば、ＧＰＵは次いで、第１のシェーディング操作を実行するために指定されたシェーディングユニットを用いて、グラフィックスレンダリングパイプラインの第２の異なるシェーダステージと関連付けられるシェーディング操作を実行することができる。たとえば、ＧＰＵは、第１のシェーダステージと関連付けられる入力／出力インターフェースを堅持しながら、第２のステージと関連付けられるシェーディング操作を実行することができる。このようにして、ＧＰＵは、同じシェーディングユニットを用いて複数のシェーディング操作を実行することによって、より多くのシェーディングリソースを有するＧＰＵをエミュレートすることができる。
【０００６】
[0006]ある例では、本開示の態様は、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することと、ここにおいて、ハードウェアユニットが、入力として単一の頂点を受け取り、出力として単一の頂点を生成するように構成される、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するためにジオメトリシェーディング操作を実行することとを含み、ジオメトリシェーディング操作が、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて１つまたは複数の新たな頂点を出力する、グラフィックスをレンダリングする方法に関する。
【０００７】
[0007]別の例では、本開示の態様は、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することと、ここにおいて、ハードウェアユニットが、入力として単一の頂点を受け取り、出力として単一の頂点を生成する、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、頂点シェーディングされた頂

点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するために、ジオメトリシェーディング操作を実行することとを行うように構成され、ジオメトリシェーディング操作が、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて１つまたは複数の新たな頂点を出力する、１つまたは複数のプロセッサを含むグラフィックスをレンダリングするためのグラフィックスプロセシングユニットに関する。

【０００８】

[0008]別の例では、本開示の態様は、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行するための手段と、ここにおいて、ハードウェアユニットが、入力として単一の頂点を受け取り、出力として単一の頂点を生成するように構成される、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するために、ジオメトリシェーディング操作を実行するための手段とを含み、ジオメトリシェーディング操作が、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて１つまたは複数の新たな頂点を出力する、グラフィックスをレンダリングするための装置に関する。

【０００９】

[0009]別の例では、本開示の態様は、命令を記憶した非一時的コンピュータ可読媒体に関し、命令は、実行されると、１つまたは複数のプロセッサに、頂点シェーディングのために指定されたシェーディングユニットを用いて、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングするために頂点シェーディング操作を実行することと、ここにおいて、ハードウェアユニットが、入力として単一の頂点を受け取り、出力として単一の頂点を生成するように構成される、頂点シェーディングのために指定されたハードウェアシェーディングユニットを用いて、頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するために、ジオメトリシェーディング操作を実行することとを行わせ、ジオメトリシェーディング操作が、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて１つまたは複数の新たな頂点を出力する。

【００１０】

[0010]別の例では、本開示の態様は、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアユニットを用いて、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することと、ここにおいて、ハードウェアユニットが、入力として単一の頂点を受け取り、出力として単一の頂点を生成するインターフェースを堅持する、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアユニットを用いて、頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するハルシェーディング操作を実行することとを含み、１つまたは複数のハルシェーディング操作が、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて１つまたは複数の制御ポイントを出力する、グラフィックスをレンダリングするための方法に関する。

【００１１】

[0011]別の例では、本開示の態様は、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアユニットを用いて、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することと、ここにおいて、ハードウェアユニットが、入力として単一の頂点を受け取り、出力として単一の頂点を生成するインターフェースを堅持する、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアユニットを用いて、頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するハルシェーディング操作を実行することとを行うように構成される

１つまたは複数のプロセッサを含み、１つまたは複数のハルシェーディング操作が、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて１つまたは複数の制御ポイントを出力する、グラフィックスをレンダリングするためのグラフィックスプロセシングユニットに関する。

【００１２】

[0012]別の例では、本開示の態様は、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアユニットを用いて、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行するための手段と、ここにおいて、ハードウェアユニットが、入力として単一の頂点を受け取り、出力として単一の頂点を生成するインターフェースを堅持する、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアユニットを用いて、頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するハルシェーディング操作を実行するための手段とを含み、１つまたは複数のハルシェーディング操作が、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて１つまたは複数の制御ポイントを出力する、グラフィックスをレンダリングするための装置に関する。

【００１３】

[0013]別の例では、本開示の態様は、命令を記憶した非一時的コンピュータ可読媒体に関し、命令は、実行されると、１つまたは複数のプロセッサに、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアユニットを用いて、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することと、ここにおいて、ハードウェアユニットが、入力として単一の頂点を受け取り、出力として単一の頂点を生成するインターフェースを堅持する、頂点シェーディングのために指定されたグラフィックスプロセシングユニットのハードウェアユニットを用いて、頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するために、ハルシェーディング操作を実行することとを行わせ、１つまたは複数のハルシェーディング操作が、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて１つまたは複数の制御ポイントを出力する。

【００１４】

[0014]ある例では、本開示の態様は、レンダリングパイプラインの第１のシェーダステージと関連付けられる第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定することと、第１のシェーディング操作が完了すると、ハードウェアシェーディングユニットの動作モードを切り替えることと、第１のシェーディング操作を実行するように指定されたグラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行することとを含む、グラフィックスをレンダリングする方法に関する。

【００１５】

[0015]別の例では、本開示の態様は、レンダリングパイプラインの第１のシェーダステージと関連付けられる第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定し、第１のシェーディング操作が完了すると、ハードウェアシェーディングユニットの動作モードを切り替え、第１のシェーディング操作を実行するように指定されたグラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行するように構成される、１つまたは複数のプロセッサを備える、グラフィックスをレンダリングするためのグラフィックスプロセシングユニットに関する。

【００１６】

[0016]別の例では、本開示の態様は、レンダリングパイプラインの第１のシェーダステ

ージと関連付けられる第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定するための手段と、第１のシェーディング操作が完了すると、ハードウェアシェーディングユニットの動作モードを切り替えるための手段と、第１のシェーディング操作を実行するように指定されたグラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行するための手段とを含む、グラフィックスをレンダリングするための装置に関する。

【００１７】

[0017]別の例では、本開示の態様は、命令を記憶した非一時的コンピュータ可読媒体に関し、命令は、実行されると、１つまたは複数のプロセッサに、レンダリングパイプラインの第１のシェーダステージと関連付けられる第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定することと、第１のシェーディング操作が完了すると、ハードウェアシェーディングユニットの動作モードを切り替えることと、第１のシェーディング操作を実行するように指定されたグラフィックスプロセシングユニットのハードウェアシェーディングユニットを用いて、レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行することとを行わせる。

【００１８】

[0018]本開示の１つまたは複数の例の詳細が、添付の図面および以下の説明に記載される。他の特徴、目的、および利点は、説明および図面、ならびに特許請求の範囲から明らかになるであろう。

【図面の簡単な説明】

【００１９】

【図１】[0019]本開示で説明される技法を実装し得るコンピューティングデバイスを示すブロック図。

【図２】[0020]例示的なグラフィックスプロセシングパイプライン８０を示すブロック図。

【図３Ａ】[0021]本開示の態様による、グラフィックスレンダリングパイプラインにおけるデータフローの概念図。

【図３Ｂ】本開示の態様による、グラフィックスレンダリングパイプラインにおけるデータフローの概念図。

【図４】[0022]本開示で説明される技法を実施して頂点シェーディング操作とジオメトリシェーディング操作とを実行する、ハードウェアシェーディングユニットの例示的な動作を示す図。

【図５Ａ】[0023]頂点シェーディング操作とジオメトリシェーディング操作とを実行するときにマージされた頂点シェーダ／ジオメトリシェーダハードウェアによって実行される動作のフローを示す図。

【図５Ｂ】[0024]マージされた頂点シェーダ／ジオメトリシェーダハードウェアシェーディングユニットによって実行され得る、図５Ａに示される動作のフローに対応する擬似コードを示す図。

【図６】[0025]本開示の態様による、マージされた頂点シェーディング操作とジオメトリシェーディング操作とを実行するためのグラフィックスプロセシングユニットの例示的なコンポーネントを示す図。

【図７】[0026]本開示の態様による、頂点シェーディング操作とジオメトリシェーディング操作とを実行するための例示的なプロセスを示すフローチャート。

【図８】[0027]テッセレーションステージを含む例示的なグラフィックスプロセシングパイプラインを示すブロック図。

【図９】[0028]テッセレーションを示す概念図。

【図１０Ａ】[0029]本開示の態様による、グラフィックスレンダリングパイプラインにお

けるデータフローの概念図。
【図１０Ｂ】本開示の態様による、グラフィックスレンダリングパイプラインにおけるデータフローの概念図。
【図１１】[0030]本開示で説明される技法を実施して頂点シェーディング操作とハルシェーディング操作とを実行する、ハードウェアシェーディングユニットの例示的な動作を示す図。
【図１２Ａ】[0031]頂点シェーディング操作とハルシェーディング操作とを実行するときにマージされた頂点シェーダ／ハルシェーダハードウェアシェーディングユニットによって実行される動作のフローを示す図。
【図１２Ｂ】[0032]マージされた頂点シェーダ／ハルシェーダハードウェアシェーディングユニットによって実行され得る、図１２Ａに示される動作のフローに対応する擬似コードを一般に示す図。
【図１３Ａ】[0033]ドメインシェーディング操作とジオメトリシェーディング操作とを実行するときにマージされたドメインシェーダ／ジオメトリシェーダハードウェアシェーディングユニットによって実行される動作のフローを一般に示す図。
【図１３Ｂ】[0034]マージされたドメインシェーダ／ジオメトリシェーダハードウェアシェーディングユニットによって実行され得る、図１３Ａに示される動作のフローに対応する擬似コードを一般に示す図。
【図１４】[0035]本開示の態様による、マージされた頂点シェーディング操作と、ハルシェーディング操作と、ドメインシェーディング操作と、ジオメトリシェーディング操作とを実行するためのグラフィックスプロセシングユニットの例示的なコンポーネントを示す図。
【図１５】[0036]本開示の態様による、同じハードウェアシェーディングユニットを使用して２つのレンダリングパスでグラフィックスレンダリングを実行することを示すフロー図。
【図１６】[0037]本開示の態様による、２つのパスのグラフィックスレンダリングプロセスの第１のパスと関連付けられる、グラフィックスレンダリング操作を実行することを示すフロー図。
【図１７】[0038]本開示の態様による、２つのパスのグラフィックスレンダリングプロセスの第２のパスと関連付けられる、グラフィックスレンダリング操作を実行することを示すフロー図。
【図１８】[0039]本開示の態様による、同じハードウェアシェーディングユニットによる実行のために２つ以上のシェーダステージを一緒にパッチすること（patching）を示すフロー図。
【発明を実施するための形態】
【００２０】
[0040]本開示の技法は一般に、グラフィックスレンダリングパイプラインのシェーダステージと関連付けられるシェーディング操作を実行することに関する。たとえば、グラフィックスプロセシングユニット（ＧＰＵ）は、グラフィックスレンダリングパイプラインのシェーダステージと関連付けられるシェーディング操作を実行するために、１つまたは複数のシェーディングユニットを呼び出すことができる。本開示の態様によれば、ＧＰＵは次いで、第１のシェーディング操作を実行するために指定されたシェーディングユニットを用いて、グラフィックスレンダリングパイプラインの第２の異なるシェーダステージと関連付けられるシェーディング操作を実行することができる。たとえば、ＧＰＵは、第１のシェーダステージと関連付けられる入力／出力インターフェースを堅持しながら、第２のステージと関連付けられるシェーディング操作を実行することができる。このようにして、ＧＰＵは、同じシェーディングユニットを用いて複数のシェーディング操作を実行することによって、より多くのシェーディングリソースを有するＧＰＵをエミュレートすることができる。
【００２１】

[0041]図１は、本開示で説明される技法を実装し得るコンピューティングデバイス３０を示すブロック図である。コンピューティングデバイス３０の例には、限定はされないが、ワイヤレスデバイス、いわゆるスマートフォンを含む携帯電話またはセルラー電話、携帯情報端末（ＰＤＡ）、ビデオディスプレイを含むビデオゲームコンソール、モバイルビデオゲームデバイス、モバイルビデオ会議ユニット、ラップトップコンピュータ、デスクトップコンピュータ、テレビジョンセットトップボックス、タブレットコンピューティングデバイス、電子ブックリーダー、固定式または移動式のメディアプレーヤーなどがある。

【００２２】

[0042]図１の例では、コンピューティングデバイス３０は、ＣＰＵメモリ３４を有する中央処理装置（ＣＰＵ）３２と、グラフィックスプロセシングユニット（ＧＰＵ）メモリ３８と１つまたは複数のシェーディングユニット４０とを有するＧＰＵ　３６と、ディスプレイユニット４２と、ディスプレイバッファユニット４４と、ユーザインターフェースユニット４６と、ストレージユニット４８とを含む。加えて、ストレージユニット４８は、コンパイラ５４と、ＧＰＵプログラム５２と、ローカルにコンパイルされたＧＰＵプログラム５６とを有する、ＧＰＵドライバ５０を記憶することができる。

【００２３】

[0043]ＣＰＵ　３２の例には、限定はされないが、デジタル信号プロセッサ（ＤＳＰ）、汎用マイクロプロセッサ、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブル論理アレイ（ＦＰＧＡ）、あるいは他の等価な集積回路またはディスクリート論理回路がある。ＣＰＵ　３２およびＧＰＵ　３６は図１の例では別個のユニットとして示されるが、いくつかの例では、ＣＰＵ　３２およびＧＰＵ　３６は単一のユニットへとマージされ得る。ＣＰＵ　３２は１つまたは複数のアプリケーションを実行し得る。アプリケーションの例には、ウェブブラウザ、電子メールアプリケーション、スプレッドシート、ビデオゲーム、オーディオキャプチャおよび／またはビデオキャプチャ、再生または編集アプリケーション、あるいは、ディスプレイユニット４２を介して提示されるべき画像データの生成を開始する他のアプリケーションがあり得る。

【００２４】

[0044]図１に示される例では、ＣＰＵ　３２はＣＰＵメモリ３４を含む。ＣＰＵメモリ３４は、機械コードまたはオブジェクトコードを実行する際に使用されるオンチップストレージまたはメモリを表し得る。ＣＰＵメモリ３４は各々、一定の数のデジタルビットを記憶することが可能なハードウェアメモリレジスタを備え得る。ＣＰＵ　３２は、たとえばシステムバスを通じてアクセスされ得るストレージユニット４８から値を読み取ること、またはそれに値を書き込むことよりも迅速に、ローカルＣＰＵメモリ３４から値を読み取り、またはそれに値を書き込むことが可能であり得る。

【００２５】

[0045]ＧＰＵ　３６は、グラフィカルな操作を実行するための１つまたは複数の専用プロセッサを表す。すなわち、たとえば、ＧＰＵ　３６は、グラフィックスをレンダリングしＧＰＵアプリケーションを実行するための、固定機能のコンポーネントとプログラマブルコンポーネントとを有する専用ハードウェアユニットであり得る。ＧＰＵ　３６はまた、ＤＳＰ、汎用マイクロプロセッサ、ＡＳＩＣ、ＦＰＧＡ、あるいは他の等価な集積回路またはディスクリート論理回路を含み得る。

【００２６】

[0046]ＧＰＵ　３６はまた、機械コードまたはオブジェクトコードを実行する際に使用されるオンチップストレージまたはメモリを表し得る、ＧＰＵ　メモリ３８を含む。ＧＰＵメモリ３８は各々、一定の数のデジタルビットを記憶することが可能なハードウェアメモリレジスタを備え得る。ＧＰＵ　３６は、たとえばシステムバスを通じてアクセスされ得るストレージユニット４８から値を読み取ること、またはそれに値を書き込むことよりも迅速に、ローカルＧＰＵメモリ３８から値を読み取り、またはそれに値を書き込むことが可能であり得る。

【００２７】
[0047]ＧＰＵ　３６は、シェーディングユニット４０を含む。以下でより詳細に説明されるように、シェーディングユニット４０は、プロセシングコンポーネントのプログラム可能なパイプラインとして構成され得る。いくつかの例では、シェーディングユニット４０は、「シェーダプロセッサ」または「統一されたシェーダ」と呼ばれることがあり、グラフィックスをレンダリングするために、ジオメトリシェーディング操作、頂点シェーディング操作、ピクセルシェーディング操作、または他のシェーディング操作を実行することができる。シェーディングユニット４０は、命令をフェッチして復号するためのコンポーネント、算術計算を実行するための１つまたは複数の算術論理ユニット（「ＡＬＵ」）、および１つまたは複数のメモリ、キャッシュ、もしくはレジスタのような、わかりやすくするために図１には特に示されていない１つまたは複数のコンポーネントを含み得る。
【００２８】
[0048]ディスプレイユニット４２は、閲覧者により使用される、ビデオデータ、画像、テキストまたは他のタイプのデータを表示することが可能なユニットを表す。ディスプレイユニット４２は、液晶ディスプレイ（ＬＣＤ）、発光ダイオード（ＬＥＤ）ディスプレイ、有機ＬＥＤ（ＯＬＥＤ）、アクティブマトリックスＯＬＥＤ（ＡＭＯＬＥＤ）ディスプレイなどを含み得る。
【００２９】
[0049]ディスプレイバッファユニット４４は、ディスプレイユニット４２のための、写真またはビデオフレームのような像の提示のためのデータを記憶することに専用のメモリまたはストレージデバイスを表す。ディスプレイバッファユニット４４は、複数の記憶位置を含む２次元バッファを表し得る。ディスプレイバッファユニット４４内の記憶位置の数は、ディスプレイユニット４２上に表示されるべきピクセルの数と実質的に同様であり得る。たとえば、ディスプレイユニット４２が６４０×４８０のピクセルを含むように構成される場合、ディスプレイバッファユニット４４は６４０×４８０の記憶位置を含み得る。ディスプレイバッファユニット４４は、ＧＰＵ　３６によって処理されるピクセルの各々に対する最終的なピクセル値を記憶し得る。ディスプレイユニット４２は、ディスプレイバッファユニット４４から最終的なピクセル値を取り出し、ディスプレイバッファユニット４４に記憶されたピクセル値に基づいて最終的な画像を表示し得る。
【００３０】
[0050]ユーザインターフェースユニット４６は、ユーザが、ＣＰＵ　３２のような、コンピューティングデバイス３０の他のユニットと対話し得るときに用いる、または別様にそれらのユニットと通信するためにインターフェースし得るときに用いるユニットを表す。ユーザインターフェースユニット４６の例には、限定はされないが、トラックボール、マウス、キーボード、および他のタイプの入力デバイスがある。ユーザインターフェースユニット４６はまた、タッチスクリーンであってよく、ディスプレイユニット４２の一部として組み込まれ得る。
【００３１】
[0051]ストレージユニット４８は１つまたは複数のコンピュータ可読記憶媒体を備え得る。ストレージユニット４８の例には、限定はされないが、ランダムアクセスメモリ（ＲＡＭ）、読取り専用メモリ（ＲＯＭ）、電気消去可能プログラマブル読取り専用メモリ（ＥＥＰＲＯＭ（登録商標））、ＣＤ－ＲＯＭもしくは他の光ディスクストレージ、磁気ディスクストレージもしくは他の磁気ストレージデバイス、フラッシュメモリ、または、命令もしくはデータ構造の形態の所望のプログラムコードを記憶するために使用されコンピュータまたはプロセッサによってアクセスされ得る、任意の他の媒体がある。
【００３２】
[0052]いくつかの例示的な実装形態では、ストレージユニット４８は、本開示においてＣＰＵ　３２およびＧＰＵ　３６に起因する機能をＣＰＵ　３２および／またはＧＰＵ　３６に実行させる命令を含み得る。ストレージユニット４８は、いくつかの例では、非一時的記憶媒体と見なされ得る。「非一時的」という用語は、記憶媒体が、搬送波または伝

搬信号では実施されないことを示し得る。しかしながら、「非一時的」という用語は、ストレージユニット４８が可動ではないことを意味するものと解釈されるべきでない。一例として、ストレージユニット４８は、コンピューティングデバイス３０から取り外され、別のデバイスに移され得る。別の例として、ストレージユニット４８と実質的に同様のストレージユニットが、コンピューティングデバイス３０に挿入され得る。いくつかの例では、非一時的記憶媒体は、時間経過に伴って変動し得るデータを（たとえば、ＲＡＭに）記憶し得る。

【００３３】

[0053]図２の例に示されるように、ストレージユニット４８は、ＧＰＵドライバ５０と、コンパイラ５４と、ＧＰＵプログラム５２と、ローカルにコンパイルされるＧＰＵプログラム５６とを記憶する。ＧＰＵドライバ５０は、ＧＰＵ　３６にアクセスするためのインターフェースを与えるコンピュータプログラムまたは実行可能コードを表す。ＣＰＵ　３２は、ＧＰＵ　３６とインターフェースするために、ＧＰＵドライバ５０またはその一部を実行し、これが理由で、ＧＰＵドライバ５０は、図１の例では、ＣＰＵ　３２内に「ＧＰＵドライバ５０」と標識された破線ボックスとして示されている。ＧＰＵドライバ５０は、ＧＰＵプログラム５２を含む、ＣＰＵ　３２によって実行されるプログラムまたは他の実行ファイルにとってアクセス可能である。

【００３４】

[0054]ＧＰＵプログラム５２は、たとえば、アプリケーションプログラミングインターフェース（ＡＰＩ）を使用する、高水準（ＨＬ）プログラミング言語で書かれたコードを含み得る。ＡＰＩの例には、Ｏｐｅｎ－Ｃｏｍｐｕｔｉｎｇ　Ｌａｎｇｕａｇｅ（「ＯｐｅｎＣＬ」）、Ｏｐｅｎ　Ｇｒａｐｈｉｃｓ　Ｌｉｂｒａｒｙ（「ＯｐｅｎＧＬ」）、およびＭｉｃｒｏｓｏｆｔ社により開発されたＤｉｒｅｃｔＸがある。一般に、ＡＰＩは、関連するハードウェアによって実行される、所定の標準化されたコマンドのセットを含む。ＡＰＩコマンドは、ユーザが、ハードウェアコンポーネントの仕様についてのユーザの知識を伴わずに、コマンドを実行するようにＧＰＵのハードウェアコンポーネントに命令することを可能にする。

【００３５】

[0055]ＧＰＵプログラム５２は、ＧＰＵドライバ５０によって与えられる１つまたは複数の機能を呼び出すか、またはさもなければ含み得る。ＣＰＵ　３２は一般に、ＧＰＵプログラム５２が埋め込まれたプログラムを実行し、ＧＰＵプログラム５２に遭遇すると、ＧＰＵプログラム５２をＧＰＵドライバ５０に（たとえば、コマンドストリームの形式で）渡す。ＣＰＵ　３２は、この状況では、ＧＰＵプログラム５２を処理するために、ＧＰＵドライバ５０を実行する。すなわち、たとえば、ＧＰＵドライバ５０は、ＧＰＵプログラム５２を、ＧＰＵ　３６によって実行可能なオブジェクトコードまたは機械コードへとコンパイルすることによって、ＧＰＵプログラム５２を処理し得る。このオブジェクトコードは、ローカルにコンパイルされたＧＰＵプログラム５６として図１の例では示されている。

【００３６】

[0056]いくつかの例では、コンパイラ５４は、リアルタイムまたは準リアルタイムで動作して、ＧＰＵプログラム５２が埋め込まれたプログラムの実行の間に、ＧＰＵプログラム５２をコンパイルすることができる。たとえば、コンパイラ５４は一般に、ＨＬプログラミング言語に従って定義されたＨＬ命令を低水準（ＬＬ）プログラミング言語のＬＬ命令へと縮小するモジュールを表す。コンパイルの後に、これらのＬＬ命令は、ＦＰＧＡ、ＡＳＩＣなど（たとえば、ＣＰＵ　３２およびＧＰＵ　３６を含む）のような、特定のタイプのプロセッサまたは他のタイプのハードウェアによって実行されることが可能である。

【００３７】

[0057]ＬＬプログラミング言語は、それらが、プロセッサまたは他のタイプのハードウェアの命令セットアーキテクチャからの抽象化をほとんど行わず、またはより低水準の抽

象化を行うという意味において、低水準と見なされ得る。ＬＬ言語は一般に、アセンブリ言語および／または機械語を指す。アセンブリ言語は、機械語よりもわずかに高度なＬＬ言語であるが、一般に、アセンブリ言語は、コンパイラまたは他の変換モジュールを使用せずに機械語に変換され得る。機械語は、ｘ８６機械コードのような、基礎をなすハードウェア、たとえば、プロセッサによってネイティブに実行されるものと、同じではないとしても同様である命令を定義する任意の言語を表す（ｘ８６は、Ｉｎｔｅｌ　Ｃｏｒｐｏｒａｔｉｏｎによって開発されたｘ８６プロセッサの命令セットアーキテクチャを指す）。

【００３８】

[0058]いずれの場合でも、コンパイラ５４は、ＨＬプログラミング言語に従って定義されたＨＬ命令を、基礎をなすハードウェアによってサポートされるＬＬ命令へと変換することができる。これらのＨＬプログラミング言語に従って定義されたソフトウェアが、実際の基礎をなすハードウェアによってより直接的に実行されることが可能なように、コンパイラ５４は、ＨＬプログラミング言語（およびＡＰＩ）と関連付けられる抽象性を除去する。

【００３９】

[0059]図１の例では、コンパイラ５４は、ＧＰＵプログラム５２を含むＨＬコードを実行するとき、ＣＰＵ　３２からＧＰＵプログラム５２を受け取り得る。コンパイラ５４は、ＬＬプログラミング言語に準拠するローカルでコンパイルされたＧＰＵプログラム５６を生成するために、ＧＰＵプログラム５２をコンパイルし得る。コンパイラ５４は、次いで、ＬＬ命令を含むローカルにコンパイルされたＧＰＵプログラム５６を出力する。

【００４０】

[0060]ＧＰＵ　３６は一般に、（ＧＰＵ　３６内で「ローカルにコンパイルされたＧＰＵプログラム５６」と標識された破線ボックスによって示されるように）ローカルにコンパイルされたＧＰＵプログラム５６を受け取り、その後、いくつかの例では、ＧＰＵ　３６は、画像をレンダリングし、画像のレンダリングされた部分をディスプレイバッファユニット４４に出力する。たとえば、ＧＰＵ　３６は、ディスプレイユニット４２において表示されるべき多数のプリミティブを生成することができる。プリミティブは、線（曲線、スプラインなどを含む）、点、円、楕円、多角形（通常、多角形は１つまたは複数の三角形の集合体として定義される）、または任意の他の２次元（２Ｄ）プリミティブの１つまたは複数を含み得る。「プリミティブ」という用語は、立方体、円柱、球体、円錐、三角錐、トーラスなどのような、３次元（３Ｄ）プリミティブも指し得る。一般に、「プリミティブ」という用語は、ディスプレイユニット４２を介して画像（またはビデオデータの状況ではフレーム）として表示するための、ＧＰＵ　３６によってレンダリングされることが可能な任意の基本的な幾何学的形状または要素を指す。

【００４１】

[0061]ＧＰＵ　３６は、１つまたは複数のモデル変換（これは状態データにおいても規定され得る）を適用することによって、プリミティブとプリミティブの他の状態データ（たとえば、色、テクスチャ、照明、カメラ構成、または他の様相）とを、いわゆる「ワールド空間」へと変換することができる。変換されると、ＧＰＵ　３６は、アクティブなカメラに対するビュー変換を適用して（これも、カメラを定義する状態データにおいて規定され得る）、プリミティブおよび照明の座標を、カメラ空間またはアイ空間に変換することができる。ＧＰＵ　３６はまた、任意のアクティブな照明のもとでの、プリミティブの外観をレンダリングするために頂点シェーディングを実行することができる。ＧＰＵ　３６は、上のモデル、ワールド空間またはビュー空間の１つまたは複数において頂点シェーディングを実行することができる（しかし、頂点シェーディングは通常、ワールド空間において実行される）。

【００４２】

[0062]プリミティブがシェーディングされると、ＧＰＵ　３６は投影を実行して、画像を、一例では（－１，－１，－１）および（１，１，１）において端点を伴う単位立方体

へと投影することができる。この単位立方体は通常、標準ビューボリュームと呼ばれる。モデルをアイ空間から標準ビューボリュームへと変換した後で、ＧＰＵ　３６は、ビューボリューム内に少なくとも部分的にも存在しないあらゆるプリミティブを除去するために、クリッピングを実行することができる。言い換えると、ＧＰＵ　３６は、カメラのフレーム内にないあらゆるプリミティブを除去することができる。ＧＰＵ　３６は次いで、プリミティブの３Ｄ座標をスクリーンの２Ｄ座標へと実質的に縮小する、プリミティブの座標をビューボリュームからスクリーン空間へとマッピングすることができる。

【００４３】

[0063]関連するシェーディングデータを伴うプリミティブを定義する変換され投影された頂点が与えられると、ＧＰＵ　３６は次いで、プリミティブをラスタライズすることができる。たとえば、ＧＰＵ　３６は、プリミティブによって覆われるスクリーンのピクセルに対する色を計算し設定することができる。ラスタライズの間、ＧＰＵ　３６は、プリミティブと関連付けられる任意のテクスチャを適用することができる（テクスチャは状態データを備え得る）。ＧＰＵ　３６はまた、ラスタライズの間に、深度テストとも呼ばれるＺバッファアルゴリズムを実行して、プリミティブおよび／またはオブジェクトのいずれかが任意の他のオブジェクトによって塞がれるかどうかを判定することができる。ＧＰＵ　３６が各プリミティブをスクリーンに描く順序を知るように、Ｚバッファアルゴリズムは、プリミティブの深度に従ってプリミティブを分類する。ＧＰＵ　３６は、レンダリングされたピクセルをディスプレイバッファユニット４４に出力する。

【００４４】

[0064]ディスプレイバッファユニット４４は、画像全体がレンダリングされるまで、レンダリングされた画像のレンダリングされたピクセルを一時的に記憶し得る。ディスプレイバッファユニット４４は、この状況では画像フレームバッファと見なされ得る。ディスプレイバッファユニット４４は、次いで、ディスプレイユニット４２上に表示されるべきレンダリングされた画像を送信し得る。いくつかの代替的な例では、ＧＰＵ　３６は、画像をディスプレイバッファユニット４４に一時的に記憶するのではなく、画像のレンダリングされた部分を表示のためにディスプレイユニット４２に直接出力し得る。ディスプレイユニット４２は、次いで、ディスプレイバッファユニット７８に記憶された画像を表示し得る。

【００４５】

[0065]上で説明された方式でピクセルをレンダリングするために、ＧＰＵ　３６は、（たとえば、図２および図８に関してより詳細に説明されるように）種々のシェーディング操作を実行するように、シェーディングユニット４０を指定することができる。しかしながら、比較的短いレンダリングパイプラインをサポートするように設計されたいくつかのＧＰＵ（ＧＰＵ　３６のような）は、拡大されたレンダリングパイプラインを有するＡＰＩをサポートすることが不可能であり得る。たとえば、いくつかのＧＰＵは、３つ以上の異なるタイプのシェーディング操作を実行するようにシェーディングユニット４０を指定することを妨げられ得る。

【００４６】

[0066]ある例では、ＧＰＵ　３６は、頂点シェーディング操作とピクセルシェーディング操作とを実行するようにシェーディングユニット４０を指定することができる。この例では、ＧＰＵ　３６は、ハルシェーダ、ドメインシェーダ、および／またはジオメトリシェーダと関連付けられる操作を実行するようにシェーディングユニット４０を指定するためのリソースを欠いていることがある。すなわち、ハードウェアおよび／ソフトウェアの制約は、ＧＰＵ　３６が、ハルシェーディング操作、ドメインシェーディング操作、および／またはジオメトリシェーディング操作を実行するようにシェーディングユニット４０を指定することを妨げ得る。したがって、ＧＰＵ　３６は、そのような機能を含むＡＰＩと関連付けられるシェーダステージをサポートすることが不可能であり得る。

【００４７】

[0067]たとえば、以前のＤｉｒｅｃｔＸ　９　ＡＰＩ（Ｄｉｒｅｃｔ３Ｄ　９　ＡＰＩ

を含み得る、Ｍｉｃｒｏｓｏｆｔによって開発された）をサポートしていた以前のＧＰＵは、ＤｉｒｅｃｔＸ　１０　ＡＰＩ（Ｄｉｒｅｃｔ３Ｄ　１０　ＡＰＩを含み得る）をサポートすることが不可能であり得る。すなわち、ＤｉｒｅｃｔＸ　１０　ＡＰＩの特徴の少なくともいくつか（たとえば、いくつかのシェーダステージのような）は、以前のＧＰＵを使用して実行されることが不可能であり得る。その上、以前のＤｉｒｅｃｔＸ　９　ＡＰＩとＤｉｒｅｃｔＸ　１０　ＡＰＩとをサポートしていたＧＰＵは、ＤｉｒｅｃｔＸ　１１　ＡＰＩのすべての機能をサポートすることが不可能であり得る。そのような非互換性は、ＤｉｒｅｃｔＸ　１０またはＤｉｒｅｃｔＸ　１１を利用するソフトウェアまたは他のアプリケーションを実行することに対するサポートをもはや提供し得ない、現在展開されている大量のＧＰＵをもたらし得る。上の例はＡＰＩのＭｉｃｒｏｓｏｆｔのＤｉｒｅｃｔＸのファミリーに関して説明されるが、同様の互換性の問題は、他のＡＰＩおよびレガシーのＧＰＵ　３６について存在し得る。

【００４８】

[0068]加えて、比較的長いグラフィックスプロセシングパイプライン（たとえば、追加のシェーダステージを有するレンダリングパイプライン）をサポートすることは、より複雑なハードウェア構成を必要とし得る。たとえば、シェーディングユニット４０の専用の１つによって実施されるときに、ジオメトリシェーダステージをレンダリングパイプラインに導入して、ジオメトリシェーディングを実行することは、オフチップメモリに対する追加の読取りと書込みとをもたらし得る。すなわち、ＧＰＵ　３６は最初に、シェーディングユニット４０の１つを用いて頂点シェーディングを実行し、頂点をストレージユニット４８に記憶することができる。ＧＰＵ　３６はまた、頂点シェーダによって出力される頂点を読み取り、シェーディングユニット４０の１つによってジオメトリシェーディングを実行するときに生成される新たな頂点を書き込むことができる。テッセレーションステージ（たとえば、ハルシェーダステージおよびドメインシェーダステージ）をレンダリングパイプラインに含めることは、以下で説明されるように、同様の複雑さをもたらし得る。

【００４９】

[0069]オフチップメモリに対する追加の読取りおよび書込みは、メモリバスの帯域幅（たとえば、ＧＰＵ　３６をストレージユニット４８に接続する通信チャネル）を消費しながら、また、読取りおよび書込みが各々、メモリバスとストレージユニット４８とに電力供給することを必要とすることを考えると、消費される電力の量を潜在的に増やし得る。この意味で、各シェーダステージに対して専用のシェーディングユニット４０を使用する、多くのステージを伴うグラフィックスパイプラインを実装することは、より電力効率の低いＧＰＵをもたらし得る。加えて、そのようなＧＰＵ　３６はまた、ストレージユニット４８からのデータの取り出しの遅延により、レンダリングされる画像の出力に関して実行がより低速であり得る。

【００５０】

[0070]本開示の態様は一般に、シェーディングユニット４０の１つが２つ以上のシェーディング機能を実行できるように、シェーディングユニット４０の１つまたは複数の機能をマージすることに関する。たとえば、通常、ＧＰＵ　３６は、特定のシェーディング操作を実行するようにシェーディングユニット４０を指定することによって、レンダリングプロセス（シェーダステージを有するレンダリングパイプラインと呼ばれ得る）を実行することができ、シェーディングユニット４０の各々は、同じシェーダの複数のインスタンスを同時に実装することができる。すなわち、ＧＰＵ　３６は、たとえば、頂点シェーダの最大で２５６個の同時のインスタンスをサポートする、頂点シェーディング操作を実行するように、シェーディングユニット４０の１つまたは複数を指定することができる。ＧＰＵ　３６はまた、たとえば、ピクセルシェーダの最大で２５６個の同時のインスタンスをサポートする、ピクセルシェーディング操作を実行するように、シェーディングユニット４０の１つまたは複数を指定することができる。これらのハードウェアユニットは、次の指定されたハードウェアユニットがグラフィックスプロセシングパイプラインにおいて

以前のハードウェアユニットの出力を処理することに利用可能となるまで、ストレージユニット４８のようなオフチップメモリに、３つのシェーダのうちの実行されている１つからの出力を記憶することができる。

【００５１】

[0071]本開示の態様は、単数形（たとえば、１つのハードウェアシェーディングユニット）で特定のハードウェアシェーディングユニットに言及することがあるが、そのようなユニットは実際には、１つまたは複数のシェーディングユニット４０（２つ以上のシェーダプロセッサ）、さらには、シェーディング操作を実行するためのＧＰＵ　３６の１つまたは複数の他のコンポーネントを備え得ることを理解されたい。たとえば、上で述べられたように、ＧＰＵ　３６は、複数の関連するシェーディングユニット４０を有し得る。ＧＰＵ　３６は、同じシェーディング操作を実行するように、シェーディングユニット４０のうちの２つ以上を指定することができ、シェーディングユニット４０の各々は、シェーディング操作をマージするための、本開示の技法を実行するように構成される。一般に、ハードウェアシェーディングユニットは、特定のシェーディング操作を実行するための、ＧＰＵ　３６のようなＧＰＵによって呼び出されるハードウェアコンポーネントのセットを指し得る。

【００５２】

[0072]一例では、本開示の態様は、単一のハードウェアシェーディングユニットを用いて、頂点シェーディング操作とジオメトリシェーディング操作とを実行することを含む。別の例では、本開示の態様は、単一のハードウェアシェーディングユニットを用いて、頂点シェーディング操作とハルシェーディング操作とを実行することを含む。さらに別の例では、本開示の態様は、単一のハードウェアシェーディングユニットを用いて、ドメインシェーディング操作とジオメトリシェーディング操作とを実行することを含む。本開示の態様はまた、ハードウェアシェーディングユニットが複数のシェーディング操作の間を移行する方式に関する。すなわち、本開示の態様は、ハードウェアシェーディングユニットを用いて第１のシェーディング操作を実行することと、同じハードウェアシェーディングユニットを用いて第２のシェーディング操作を実行することとの間を移行することに関する。

【００５３】

[0073]たとえば、本開示の態様によれば、ＧＰＵ　３６は、頂点シェーディング操作を実行するように指定されたシェーディングユニット４０を用いて、頂点シェーディングされた頂点を出力するために、頂点シェーディング操作を実行して、入力された頂点をシェーディングすることができる。この例では、シェーディングユニット４０は、入力として単一の頂点を受け取り出力として単一の頂点を生成するインターフェースを用いて構成され得る。加えて、ＧＰＵ　３６は、同じシェーディングユニット４０を用いて、ジオメトリシェーディング操作を実行して、頂点シェーディングされた頂点の１つまたは複数に基づいて、１つまたは複数の新たな頂点を生成することができる。ジオメトリシェーディング操作は、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて、１つまたは複数の新たな頂点を出力することができる。再び、単一のシェーディングユニット４０に関して説明されるが、これらの技法は、ＧＰＵ　３６の複数のシェーディングユニット４０によって同時に実施され得る。

【００５４】

[0074]いくつかのＡＰＩは、頂点シェーディング操作を実行するように指定されたシェーディングユニット４０が１：１インターフェースを実装または堅持することを求めることがあり、１：１インターフェースは、入力として単一の頂点を受け取り出力として単一の頂点を生成する。対照的に、ジオメトリシェーディング操作を実行するように指定されたシェーディングユニット４０は、１：Ｎインターフェースを実装または堅持することができ、１：Ｎインターフェースは、入力として１つまたは複数の頂点を受け取り、出力として１つまたは複数の（かつしばしば多数の、したがって上で「Ｎ」が使用される）頂点を生成する。

【００５５】
[0075]本開示の態様によれば、ＧＰＵ　３６は、頂点シェーディング操作を実行するように指定されたシェーディングユニット４０の１：１インターフェースを利用して、ジオメトリシェーダプログラムの複数のインスタンスを呼び出すことによって、この１：Ｎジオメトリシェーダインターフェースをエミュレートすることができる。ＧＰＵ　３６は、これらのジオメトリシェーダプログラムの各々を同時に実行して、ジオメトリシェーダ操作を実行することから得られる新たな頂点の１つを生成することができる。すなわち、シェーディングユニット４０が一般に「シェーダプログラム」と呼ばれるものの複数のインスタンスを同時に実行できるように、シェーディングユニット４０は、ＨＬＳＬ（たとえば、グラフィックスレンダリングＡＰＩを伴う）を使用してプログラム可能であり得る。これらのシェーダプログラムは、「ファイバー」または「スレッド」（これらの両方が、プログラムまたは実行のスレッドを形成する命令のストリームを指し得る）と呼ばれ得る。本開示の態様によれば、かつ以下でより詳細に説明されるように、ＧＰＵ　３６は、頂点シェーディング操作のために指定されるハードウェアシェーディングユニットを使用して、ジオメトリシェーダプログラムの複数のインスタンスを実行することができる。同じシェーディングユニット４０が両方のシェーダ、たとえば、頂点シェーダとジオメトリシェーダとを順番に実行するように、ＧＰＵ　３６は、ジオメトリシェーダ命令を頂点シェーダ命令に付加することができる。

【００５６】
[0076]別の例では、本開示の態様によれば、ＧＰＵ　３６は、頂点シェーディング操作を実行するように指定されたハードウェアシェーディングユニットを用いて、頂点シェーディングされた頂点を出力するために、頂点シェーディング操作を実行して、入力された頂点をシェーディングすることができる。ハードウェアシェーディングユニットは、入力として単一の頂点を受け取り出力として単一の頂点を生成するインターフェースを堅持し得る。加えて、ＧＰＵは、頂点シェーディング操作を実行するために指定された同じハードウェアシェーディングユニットを用いて、１つまたは複数のテッセレーション操作（たとえば、ハルシェーディング操作および／またはドメインシェーディング操作）を実行して、頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成することができる。１つまたは複数のテッセレーション操作は、１つまたは複数の頂点シェーディングされた頂点の少なくとも１つに対して行われて、１つまたは複数の新たな頂点を出力することができる。

【００５７】
[0077]たとえば、上で説明されたシェーダステージに加えて、いくつかのグラフィックスレンダリングパイプラインはまた、ハルシェーダステージと、テッセレータステージと、ドメインシェーダステージとを含み得る。一般に、ハルシェーダステージ、テッセレータステージ、およびドメインシェーダステージが、ハードウェアテッセレーションに対応するために含まれる。すなわち、ハルシェーダステージ、テッセレータステージ、およびドメインシェーダステージが、たとえば、ＣＰＵ　３２によって実行されているソフトウェアアプリケーションによる実行ではなく、ＧＰＵ　３６によるテッセレーションに対応するために含まれる。

【００５８】
[0078]本開示の態様によれば、ＧＰＵ　３６は、同じシェーディングユニット４０を用いて、頂点シェーディング操作とテッセレーション操作とを実行することができる。たとえば、ＧＰＵ　３６は、２つのパスで頂点シェーディング操作とテッセレーション操作とを実行することができる。本開示の態様によれば、かつ以下でより詳細に説明されるように、ＧＰＵ　３６は、異なるシェーディング操作の間の移行を可能にするための種々の値を記憶することができる。

【００５９】
[0079]ある例では、第１のパスでは、ＧＰＵ　３６は、頂点シェーディング操作とハルシェーディング操作とを実行するように１つまたは複数のシェーディングユニット４０を

指定することができる。この例では、ＧＰＵ　３６は、ハルシェーダ命令を頂点シェーダ命令に付加することができる。したがって、同じシェーディングユニット４０が、頂点シェーディング命令とハルシェーダ命令とを順番に実行する。

【００６０】

[0080]第２のパスでは、ＧＰＵ　３６は、ドメインシェーディング操作とジオメトリシェーディング操作とを実行するように１つまたは複数のシェーディングユニット４０を指定することができる。この例では、ＧＰＵ　３６は、ドメインシェーダ命令をジオメトリシェーダ命令に付加することができる。したがって、同じシェーディングユニット４０が、ドメインシェーディング操作とジオメトリシェーディング操作とを順番に実行する。複数のパスで複数のシェーディング操作を実行することによって、ＧＰＵ　３６は、同じシェーディングハードウェアを使用して、追加のシェーディング能力を有するＧＰＵをエミュレートすることができる。

【００６１】

[0081]本開示の態様はまた、ＧＰＵ　３６が複数のシェーディング操作の間を移行する方式に関する。たとえば、本開示の態様は、操作が同じハードウェアシェーディングユニットによって順番に実行されるように、シェーディング操作が一緒にパッチされる方式に関する。

【００６２】

[0082]ある例では、本開示の態様によれば、ＧＰＵ　３６は、レンダリングパイプラインの第１のシェーダステージと関連付けられる第１のシェーディング操作を実行するように、１つまたは複数のシェーディングユニット４０を指定することができる。ＧＰＵ　３６は、第１のシェーディング操作が完了すると、シェーディングユニット４０の動作モードを切り替えることができる。ＧＰＵ　３６は次いで、第１のシェーディング操作を実行するように指定された同じシェーディングユニット４０を用いて、レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行することができる。

【００６３】

[0083]いくつかの例によれば、ＧＰＵ　３６は、複数のモードを使用するシェーディング操作を一緒にパッチされることができ、各モードは、関連するシェーディング操作の特定のセットを有する。たとえば、第１のモードは、ドローコールが頂点シェーディング操作のみを含むことを示し得る。この例では、ドローコールを実行すると、ＧＰＵ　３６は、モード情報に従って頂点シェーディング操作を実行するように、１つまたは複数のシェーディングユニット４０を指定することができる。加えて、第２のモードは、ドローコールが頂点シェーディング操作とジオメトリシェーディング操作の両方を含むことを示し得る。この例では、ドローコールを実行すると、ＧＰＵ　３６は、頂点シェーディング操作を実行するように、１つまたは複数のシェーディングユニット４０を指定することができる。加えて、本開示の態様によれば、同じシェーディングユニットが頂点シェーディング操作とジオメトリシェーディング操作の両方を実行するように、ＧＰＵ　３６は、ジオメトリシェーダ命令を頂点シェーダ命令に付加することができる。以下でより詳細に説明されるように、追加のモードが、シェーダの他の組合せを示すために使用され得る。

【００６４】

[0084]いくつかの例では、ＧＰＵドライバ５０は、ＧＰＵ　３６によって使用されるモード情報を生成することができる。本開示の態様によれば、異なるシェーダ（たとえば、頂点シェーディング操作、ジオメトリシェーディング操作、ハルシェーディング操作、ドメインシェーディング操作など）は、同じシェーディングユニット４０によって順番に実行されるように、特定の方式でコンパイルされる必要はない。むしろ、各シェーダは、ＧＰＵ　３６によって、ドローのときに独立にコンパイルされ（任意の他のシェーダを参照することなく）、一緒にパッチされ得る。すなわち、ドローコールを実行すると、ＧＰＵ　３６は、ドローコールと関連付けられるモードを決定し、それに従ってコンパイルされたシェーダを一緒にパッチされることができる。

【００６５】
[0085]本開示の技法は、シェーディング操作を実行するための限られた数のシェーディングユニット４０を有するＧＰＵ（ＧＰＵ　３６のような）が、より多数のシェーディングユニット４０を有するＧＰＵをエミュレートすることが可能になり得る。たとえば、ＧＰＵ　３６は、３つ以上のシェーディング操作（たとえば、頂点シェーディング操作およびピクセルシェーディング操作）を実行するようにシェーディングユニット４０を指定することを妨げられ得るが、本開示の技法は、ＧＰＵ　３６が、シェーディングユニット４０を再構成することなく、追加のシェーディング操作（たとえば、ジオメトリシェーディング操作、ハルシェーディング操作、および／またはドメインシェーディング操作）を実行することを可能にし得る。すなわち、本技法は、シェーディングユニット４０が、他のシェーディング操作を実行しながら、いくつかのシェーダステージの入力／出力の制約を堅持することを可能にし得る。
【００６６】
[0086]その上、同じシェーディングユニット４０を用いて複数のシェーディング操作を実行することによって、本技法は、メモリのバス帯域幅の消費を減らすことができる。たとえば、他のシェーディング操作（たとえば、ジオメトリシェーディング）とともに頂点シェーディングが実行される場合、頂点シェーディングのために使用されるシェーディングユニット４０は、他のシェーダ操作を実行する前に、頂点シェーディングの結果をオフチップメモリ（ストレージユニット４８のような）に記憶する必要はない。むしろ、頂点シェーディングの結果は、ＧＰＵメモリ３８に記憶され、ジオメトリシェーディング操作のために直ちに使用され得る。
【００６７】
[0087]このようにして、本技法は、追加のシェーディングユニット４０を有するＧＰＵと比較して、メモリのバス帯域幅の消費を減らすことができ、これにより電力消費が減り得る。したがって、本技法は、追加のハードウェアシェーダユニットを有するＧＰＵよりも電力消費が少ない、より電力効率の高いＧＰＵを促進し得る。したがって、いくつかの例では、本技法は、モバイルデバイス、ラップトップコンピュータ、および一定の専用の電力供給を有さない他のタイプのデバイスのような、電力が限られているデバイスにおいて展開され得る。
【００６８】
[0088]コンピューティングデバイス３０は、明快のために図１に示されていない追加のモジュールまたはユニットを含み得ることを理解されたい。たとえば、コンピューティングデバイス３０は、データを送信し受信するための送受信機モジュールを含んでよく、コンピューティングデバイス３０と別のデバイスまたはネットワークとの間のワイヤレス通信または有線通信を可能にするための回路を含み得る。コンピューティングデバイス３０はまた、コンピューティングデバイス３０がモバイルワイヤレス電話である例において電話通信を実現するために、そのいずれも図１に示されていないスピーカーとマイクロフォンとを含んでよく、または、コンピューティングデバイス３０がメディアプレーヤーである例においてスピーカーを含んでよい。いくつかの例では、ユーザインターフェースユニット４６およびディスプレイユニット４２は、コンピューティングデバイス３０が、外部ユーザインターフェースまたはディスプレイとインターフェースする能力があるデスクトップコンピュータまたは他のデバイスである例において、コンピューティングデバイス３０の外部にあり得る。
【００６９】
[0089]図２は、例示的なグラフィックスプロセシングパイプライン８０を示すブロック図である。例示的なパイプライン８０は、入力アセンブラステージ８２と、頂点シェーダステージ８４と、ジオメトリシェーダステージ８６と、ラスタライザステージ８８と、ピクセルシェーダステージ９０と、出力マージャステージ（merger stage）９２とを含む。いくつかの例では、ＤｉｒｅｃｔＸ　１０（またはＤｉｒｅｃｔ３Ｄ　１０）のようなＡＰＩは、図２に示されるステージの各々を使用するように構成され得る。グラフィックス

プロセシングパイプライン８０は、ＧＰＵ　３６によって実行されるものとして以下で説明されるが、種々の他のグラフィックスプロセッサによって実行され得る。

【００７０】

[0090]グラフィックスプロセシングパイプライン８０は一般に、プログラム可能なステージ（たとえば、丸い角によって示される）と固定された機能のステージ（たとえば、四角形の角によって示される）とを含む。たとえば、グラフィックスレンダリングパイプライン８０のいくつかのステージと関連付けられるグラフィックスレンダリング操作は一般に、シェーディングユニット４０の１つのようなプログラム可能なシェーダプロセッサによって実行され、一方、グラフィックスレンダリングパイプライン８０の他のステージと関連付けられる他のグラフィックスレンダリング操作は一般に、ＧＰＵ３６と関連付けられるプログラム可能ではない固定された機能のハードウェアユニットによって実行される。シェーディングユニット４０によって実行されるグラフィックスレンダリングステージは一般に、「プログラム可能」ステージと呼ばれることがあり、一方、固定された機能のユニットによって実行されるステージは一般に、固定された機能ステージと呼ばれることがある。

【００７１】

[0091]入力アセンブラステージ８２は、固定された機能ステージとして図２の例では示され、一般に、グラフィックスプロセシングパイプライン８０にグラフィックスデータ（三角形、線、および点）を供給することを担う。たとえば、入力アセンブラステージ８２は、高次の表面、プリミティブなどに対する頂点データを収集し、頂点データと属性とを頂点シェーダステージ８４に出力することができる。したがって、入力アセンブラステージ８０は、固定された機能の操作を使用して、ストレージユニット４８のようなオフチップメモリから頂点を読み取ることができる。入力アセンブラステージ８０は次いで、これらの頂点からパイプラインのワークアイテムを作成することができ、一方、また、頂点識別子（「ＶｅｒｔｅｘＩＤ」）と、インスタンス識別子（「ＩｎｓｔａｎｃｅＩＤ」、これは頂点シェーダに対して利用可能にされる）と、プリミティブ識別子（「ＰｒｉｍｉｔｉｖｅＩＤ」、これはジオメトリシェーダおよびピクセルシェーダに対して利用可能にされる）とを生成する。入力アセンブラステージ８０は、頂点を読み取ると、ＶｅｒｔｅｘＩＤと、ＩｎｓｔａｎｃｅＩＤと、ＰｒｉｍｉｔｉｖｅＩＤとを自動的に生成することができる。

【００７２】

[0092]頂点シェーダステージ８４は、受信された頂点データと属性とを処理することができる。たとえば、頂点シェーダステージ８４は、変換、スキニング、頂点変位、および頂点ごとのマテリアルの属性の計算のような、頂点ごとの処理を実行することができる。いくつかの例では、頂点シェーダステージ８４は、テクスチャの座標、頂点の色、頂点の照明、フォグファクタなどを生成することができる。頂点シェーダステージ８４は一般に、単一の入力された頂点を取り込み、単一の処理された出力される頂点を出力する。

【００７３】

[0093]ジオメトリシェーダステージ８６は、頂点データ（たとえば、三角形に対しては３つの頂点、線に対しては２つの頂点、または点に対しては単一の頂点）によって定義されるプリミティブを受け取り、このプリミティブをさらに処理することができる。たとえば、ジオメトリシェーダステージ８６は、他のあり得るプロセシング操作の中でもとりわけ、シルエット－エッジの検出およびシャドウボリュームの突出のような、プリミティブごとの処理を実行することができる。したがって、ジオメトリシェーダステージ８６は、入力（１つまたは複数の頂点を含み得る）として１つのプリミティブを受け取ることができ、０個、１個、または複数個のプリミティブ（やはり１つまたは複数の頂点を含み得る）を出力する。出力プリミティブは、ジオメトリシェーダステージ８６を伴わずに可能であり得るものよりも多くのデータを含み得る。出力データの総量は、頂点の数を乗算された頂点のサイズに等しくてよく、呼出しごとに制限され得る。ジオメトリシェーダステージ８６からのストリーム出力は、このステージに達したプリミティブが、メモリユニット

４８のようなオフチップメモリに記憶されることを可能にし得る。ストリーム出力は通常、ジオメトリシェーダステージ８６と結び付けられ、（たとえば、ＡＰＩを使用して）両方が一緒にプログラムされ得る。

【００７４】

[0094]ラスタライザステージ８８は通常、プリミティブをクリッピングして、ピクセルシェーダステージ９０のためにプリミティブを準備することを担う、固定された機能のステージである。たとえば、ラスタライザステージ８８は、（カスタムクリップ境界を含む）クリッピングと、パースペクティブ分割と、ビューポート／シザーの選択および実装と、レンダリング対象の選択と、プリミティブのセットアップとを実行することができる。このようにして、ラスタライザステージ８８は、ピクセルシェーダステージ９０によるシェーディングのために多数のフラグメントを生成することができる。

【００７５】

[0095]ピクセルシェーダステージ９０は、ラスタライザステージ８８からフラグメントを受け取り、色のようなピクセルごとのデータを生成する。ピクセルシェーダステージ９６はまた、テクスチャ混合および照明モデル計算のような、ピクセルごとの処理を実行することができる。したがって、ピクセルシェーダステージ９０は、入力として１つのピクセルを受け取ることができ、同じ相対的な位置において１つのピクセル（またはそのピクセルに対して０の値）を出力することができる。

【００７６】

[0096]出力マージャステージ９２は一般に、様々なタイプの出力データ（ピクセルシェーダ値、深度およびステンシル情報のような）を組み合わせて最終的な結果を生成することを担う。たとえば、出力マージャステージ９２は、レンダリング対象（ピクセル位置）に対して、固定された機能である混合、深度、および／またはステンシル操作を実行することができる。頂点シェーダステージ８４、ジオメトリシェーダステージ８６、およびピクセルシェーダステージ９０に一般に関して上では説明されたが、前述の説明の各々は、それぞれのシェーディング操作を実行するようにＧＰＵによって指定された１つまたは複数のシェーディングユニット（シェーディングユニット４０のような）を指し得る。

【００７７】

[0097]一部のＧＰＵは、図２に示されたシェーダステージのすべてをサポートすることが不可能であり得る。たとえば、一部のＧＰＵは、ハードウェアおよび／またはソフトウェアの制約（たとえば、限られた数のシェーディングユニット４０および関連するコンポーネント）により、３つ以上のシェーディング操作を実行するようにシェーディングユニットを指定することが不可能であり得る。ある例では、一部のＧＰＵは、ジオメトリシェーディングステージ８６と関連付けられる操作をサポートすることができない。むしろ、ＧＰＵは、頂点シェーダステージ８４とピクセルシェーダステージ９０とを実行するようにシェーディングユニットに指定することに対するサポートのみを含み得る。したがって、シェーディングユニットによって実行される操作は、頂点シェーダステージ８４およびピクセルシェーダステージ９０と関連付けられる入力／出力インターフェースを堅持しなければならない。

【００７８】

[0098]加えて、いくつかの例では、ジオメトリシェーダステージ８６をパイプラインに導入することで、ジオメトリシェーダステージ８６を含まないグラフィックスプロセシングパイプラインと比較して、ストレージユニット４８に対する追加の読取りと書込みとが発生し得る。たとえば、上で述べられたように、頂点シェーダステージ８６は、ストレージユニット４８のようなオフチップメモリに頂点を書き出すことができる。ジオメトリシェーダステージ８６は、これらの頂点（頂点シェーダステージ８４によって出力された頂点）を読み取り、新たな頂点を書き込むことができ、新たな頂点は次いでピクセルシェーディングされる。ストレージユニット４８に対するこれらの追加の読取りおよび書込みは、メモリのバス帯域幅を消費しつつ、消費される電力の量も増やす可能性がある。この意味で、頂点シェーダステージ８４、ジオメトリシェーダステージ８６、およびピクセルシ

ェーダステージ９０の各々を含むグラフィックスプロセシングパイプラインを実装することは、ストレージユニット４８からデータを取り出す際の遅延が原因で、レンダリングされた画像を出力するのがより遅い可能性もある、電力効率の低いＧＰＵをもたらし得る。

【００７９】

[0099]上で述べられたように、本開示の態様は一般に、ある特定のシェーディング操作のために指定された１つのシェーディングユニット４０が２つ以上のシェーディング機能を実行できるように、シェーディングユニット４０の１つまたは複数の機能をマージすることに関する。以下でより詳細に説明されるように、いくつかの例では、１つのシェーディングユニット４０は、頂点シェーダステージ８４と関連付けられる頂点シェーディング操作を実行することを指定され得る。本開示の態様によれば、同じシェーディングユニット４０はまた、ジオメトリシェーダステージ８６と関連付けられるジオメトリシェーディング操作を実行するように実装され得る。すなわち、ＧＰＵ　３６は、頂点シェーディング操作を実行するためにシェーディングユニット４０を呼び出すことができるが、ジオメトリシェーディングのタスクを実行するようにシェーディングユニット４０を再指定することなく、ジオメトリシェーディング操作を実行するようにシェーディングユニット４０を実装することもできる。

【００８０】

[0100]図３Ａおよび図３Ｂは、本開示の態様による、グラフィックスレンダリングパイプラインにおけるデータフローの概念図である。たとえば、図３Ａは、頂点シェーダステージ１００と、ジオメトリシェーダステージ１０２、ストリームアウト１０４と、ピクセルシェーダステージ１０６とを示す。一般に、図３Ａに示される頂点シェーダステージ１００、ジオメトリシェーダステージ１０２、およびピクセルシェーダステージ１０６は各々、シェーディング操作を実行するための関連するハードウェアを表す。すなわち、たとえば、頂点シェーダステージ１００、ジオメトリシェーダステージ１０２、およびピクセルシェーダステージ１０６の各々は、それぞれのタスクを実行するように指定されたシェーディングユニット４０のような、別々に指定されたプロセシングユニットと関連付けられ得る。

【００８１】

[0101]たとえば、頂点シェーダステージ１００は、頂点シェーディング操作を実行する（シェーディングユニット４０のような）１つまたは複数のユニットを表す。すなわち、頂点シェーダステージ１００は、頂点シェーディング操作を実行するためにＧＰＵ　３６によって呼び出されたコンポーネントを含み得る。たとえば、頂点シェーダステージ１００は、入力として頂点を受け取り、３次元（３Ｄ）モデル空間からスクリーン空間中の２次元（２Ｄ）座標へと、入力された頂点を変換することができる。頂点シェーダステージ１００は次いで、頂点の変換されたバージョン（これは「変換された頂点」と呼ばれ得る）を出力することができる。頂点シェーダステージ１００は普通は新たな頂点を生成しないが、一度に１つの頂点に対して行われる。結果として、頂点シェーダステージ１００は、１対１（１：１）ステージと呼ばれることがあり、その頂点シェーダステージ１００は、単一の入力される頂点を受け取り、単一の出力される頂点を出力する。

【００８２】

[0102]ジオメトリシェーダステージ１０２は、ジオメトリシェーディング操作を実行する（シェーディングユニット４０のような）１つまたは複数のユニットを表す。すなわち、ジオメトリシェーダステージ１０２は、ジオメトリシェーディング操作を実行するためにＧＰＵ　３６によって呼び出されたコンポーネントを含み得る。たとえば、ジオメトリシェーダステージ１０２は、キューブマップに対する単一パスのレンダリング、ポイントスプライトの生成などのような、多種多様な操作を実行するのに有用であり得る。通常、ジオメトリシェーダステージ１０２は、頂点シェーダステージ１００によって頂点シェーディングされた１つまたは複数の変換された頂点からなるプリミティブを受け取る。ジオメトリシェーダステージ１０２は、ジオメトリシェーディング操作を実行して、新たなプリミティブを形成し得る新たな頂点を作成する（または場合によっては、追加の新たな頂

点を有する新たなタイプのプリミティブへと入力プリミティブを変換する）。

【００８３】

[0103]たとえば、ジオメトリシェーダステージ１０２は通常、１つまたは複数の変換された頂点によって定義されるプリミティブを受け取り、受け取られたプリミティブに基づいて１つまたは複数の新たな頂点を生成する。ジオメトリシェーダステージ１０２は次いで、新たな頂点を出力する（１つまたは複数の新たなプリミティブを形成し得る）。結果として、ジオメトリシェーダステージ１０２は、ジオメトリシェーダステージ１０２が１つまたは複数の変換された頂点を受け取り多数の新たな頂点を生成するという点で、１対多数（１：Ｎ）ステージまたはさらには多数対多数（Ｎ：Ｎ）ステージと呼ばれることがある。

【００８４】

[0104]１対多数またはさらには多数対多数であると説明されるが、ジオメトリシェーダステージ１０２はまた、いくつかの例では、新たな頂点を何ら出力しなくてよく、または、単一の新たな頂点のみを出力してよい。この点で、本技法は、すべてのインスタンスにおいて多数の頂点を出力するジオメトリシェーダのみに限定されるべきではなく、以下でより詳細に説明されるように、０個、１個、または多数の新たな頂点を出力し得る任意のジオメトリシェーダステージ１０２に関して一般に実装され得る。

【００８５】

[0105]ジオメトリシェーダステージ１０２の出力は、（たとえば、ストリームアウト１０４の間の）追加のジオメトリシェーディングのために記憶され得る。ジオメトリシェーダステージ１０２の出力はまた、新たな頂点（および変換された頂点）をラスタライズしてピクセルからなるラスター画像を生成する、ラスタライザに出力され得る。

【００８６】

[0106]ジオメトリシェーダステージ１０２からのピクセルはまた、ピクセルシェーダステージ１０６にも渡され得る。ピクセルシェーダステージ１０６（フラグメントシェーダとも呼ばれ得る）は、各ピクセルの色と他の属性とを計算し、多種多様な操作を実行してシェーディングされたピクセルを生成することができる。シェーディングされたピクセルは、深度マップとマージされてよく、他のシェーディング後操作が、コンピュータモニタ、テレビジョン、または他のタイプのディスプレイデバイスなどのディスプレイデバイスを介して表示するための出力画像を生成するために実行され得る。

【００８７】

[0107]図３Ａに示されるシェーダステージは、１つまたは複数のグラフィックスＡＰＩをサポートすることができる。説明のための例では、頂点シェーダステージ１００、ジオメトリシェーダステージ１０２、およびピクセルシェーダステージ１０６は、ＤｉｒｅｃｔＸ　１０　ＡＰＩをサポートすることができる。すなわち、ＤｉｒｅｃｔＸ　１０　ＡＰＩを使用して作成されたコードが、グラフィックスデータをレンダリングするために、頂点シェーダステージ１００、ジオメトリシェーダステージ１０２、およびピクセルシェーダステージ１０６によって実行され得る。しかしながら、ジオメトリシェーダステージ１０２は、すべてのグラフィックスレンダリングパイプラインに含まれなくてよく、すべてのＧＰＵによって実行可能でなくてよい。たとえば、ＤｉｒｅｃｔＸ　１０　ＡＰＩはジオメトリシェーダステージ１０２に対するサポートを含むが、いくつかのより以前の改訂（たとえば、ＤｉｒｅｃｔＸ　９）はそのようなサポートを含まない。したがって、ＤｉｒｅｃｔＸ　ＡＰＩのより以前の改訂によって作成されたコードを実行するように設計されるＧＰＵ（または他のＡＰＩのために設計されたＧＰＵ）は、ジオメトリシェーダステージ１０２を実行するようにシェーディングユニット４０を指定することが不可能であり得る。

【００８８】

[0108]図３Ｂは、本開示の技法による、（図３Ａに示される例に対する）グラフィックスレンダリングパイプラインにおけるデータフローの修正された概念図を示す。図３Ｂに示される例は、マージされた頂点シェーダ／ジオメトリシェーダ（ＶＳ／ＧＳ）ステージ

１１０と、ストリームアウト１１２と、ピクセルシェーダステージ１１４とを含む。本開示の態様によれば、マージされたＶＳ／ＧＳステージ１１０は、頂点シェーダステージ１００およびジオメトリシェーダステージ１０２に関して上で説明された機能を実行するための１つまたは複数のプロセシングユニットを含み得る。すなわち、頂点シェーダステージ１００およびジオメトリシェーダステージ１０２は、頂点シェーディング操作およびジオメトリシェーディング操作をそれぞれ実行するための、ＧＰＵ（ＧＰＵ　３６のような）によって呼び出される別個のユニットを表すが、本開示の態様によれば、そのような機能は、実質的に同一のハードウェア（たとえば、シェーディングユニット４０）によって実行され得る。

【００８９】

[0109]たとえば、頂点シェーディング操作がＧＰＵ　３６によって呼び出されると、ＶＳ／ＧＳステージ１１０は、頂点シェーディング操作とジオメトリシェーディング操作の両方を実行することができる。すなわち、マージされたＶＳ／ＧＳステージ１１０は、頂点シェーダステージ１００に関して上で説明された操作を実行し、ジオメトリシェーダステージ１０２に関して上で説明された操作を実行するための、シェーディングユニット４０の同じセットを含み得る。

【００９０】

[0110]しかしながら、ＧＰＵ　３６は最初は、頂点シェーディングユニットとして各シェーディングユニット４０を呼び出すので、ＧＰＵ　３６のコンポーネントは、特定のフォーマットで、たとえば、１：１の入力／出力インターフェースを堅持して、頂点シェーディングユニットからデータを受け取るように構成され得る。たとえば、ＧＰＵ　３６は、単一のエントリーをキャッシュ（たとえば、以下でより詳細に説明されるような頂点パラメータキャッシュ）へと割り振り、シェーディングされた頂点に対するシェーディングユニット４０からの出力を記憶することができる。ＧＰＵ　３６はまた、シェーディングユニット４０が呼び出される方式に基づいて、何らかのラスタライズ操作を実行することができる。以下でより詳細に説明されるように、本開示の態様は、ＧＰＵ　３６が、頂点シェーディング操作と同じシェーディングユニットによってジオメトリシェーディング操作を実行しつつ、適切なインターフェースを依然として堅持することを可能にする。

【００９１】

[0111]いくつかの例では、ジオメトリシェーダステージ１０２は、基本的に、データの小さな増幅（たとえば、ポイントスプライトの生成）のために使用され得る。そのような操作は、ジオメトリシェーダの呼出しごとに、比較的低いＡＬＵ使用量しか必要としないことがある。したがって、シェーディングユニット４０のＡＬＵは、ジオメトリシェーダステージ１０２の間は完全には利用されないことがある。本開示の態様によれば、ジオメトリシェーダステージ１０２は、マージされたＶＳ／ＧＳステージ１１０を形成するために頂点シェーダステージ１００に付加されてよく、マージされたＶＳ／ＧＳステージ１１０は、ＧＰＵアーキテクチャにおいて頂点シェーダステージ１００として呼び出され得る。上で説明された方式でマージされたＶＳ／ＧＳステージ１１０を呼び出すことで、頂点シェーディング操作とジオメトリシェーディング操作の両方が同じプロセシングユニットによって実行されることを可能にすることによって、ＡＬＵの利用率を上げることができる。

【００９２】

[0112]マージされたＶＳ／ＧＳステージ１１０を可能にするために、図４に示される例に関してより詳細に説明されるように、ＧＰＵ　３６は、頂点シェーディング操作（１：１のステージ）とジオメトリシェーディング操作（１：Ｎのステージ）との間の移行のための機能を実行することができる。このようにして、本開示の技法は、限られたリソース（たとえば、このことは、ＧＰＵが３つ以上のシェーディング操作を実行するようにシェーディングユニット４０を指定するのを妨げ得る）を有するＧＰＵが、追加のリソースを有するＧＰＵをエミュレートすることを可能にする。

【００９３】

[0113]図４は、本開示で説明される技法を実施して頂点シェーディング操作とジオメトリシェーディング操作とを実行する、ハードウェアシェーディングユニットの例示的な動作を示す図である。ＧＰＵ　３６（図１）に関して説明されるが、本開示の態様は、種々の他のコンポーネントを有する多種多様な他のＧＰＵによって実行され得る。

【００９４】

[0114]図４の例では、ＧＰＵ　３６は、頂点シェーディング操作を実行するようにシェーディングユニット４０を指定することができる。したがって、ＧＰＵ　３６のコンポーネントは、頂点のデータをシェーディングユニット４０に送り、シェーディングされた頂点のデータをシェーディングユニット４０から受け取るように構成され得る（たとえば、１：１のインターフェース）。シェーディングユニット４０は、頂点シェーダを実行して頂点シェーディング操作を行うことができ、これによって、プリミティブの第１のセット１２０を生成する。図４の例では、プリミティブの第１のセット１２０は、点ｐ０～ｐ３として図示される４つの頂点を有する、隣接した三角形を含む。

【００９５】

[0115]頂点シェーディング操作を実行した後で、ＧＰＵ　３６は、シェーディングされた頂点をローカルのメモリリソースに記憶することができる。たとえば、ＧＰＵ　３６は、（もしあれば）「切断」情報およびｓｔｒｅａｍｉｄとともに、（たとえば、ＧＰＵメモリ３８の）位置キャッシュに頂点シェーダ出力をエクスポートすることができる。頂点シェーディング操作およびジオメトリシェーディング操作は、ＶＳ　ＥＮＤ命令によって分離され得る。したがって、ＶＳ　ＥＮＤ命令を実行し頂点シェーディング操作を完了した後、頂点シェーディング操作を実行するように指定された１つまたは複数のシェーディングユニット４０は、各々、ジオメトリシェーディング操作の実行を開始する。

【００９６】

[0116]すなわち、本開示の態様によれば、頂点シェーディング操作を実行するように指定された同じシェーディングユニット４０はまた、ジオメトリシェーディング操作を実行する。たとえば、ＧＰＵ　３６は、１つまたは複数のリソースポインタを変更することによって、状態をジオメトリシェーダ固有のリソース（たとえば、ジオメトリシェーダ定数、テクスチャオフセットなど）へと変更することができる。ＧＰＵ　３６は、シェーディング操作に割り当てられたモード（ドローモード）に従って、この状態変更を実行することができる。

【００９７】

[0117]いくつかの例では、ＧＰＵ　３６は、ドローコールを実行するときにドローモードを設定することができる。ドローモードは、どのシェーディング操作がドローコールと関連付けられるかを示し得る。説明のための例では、０というドローモードは、ドローコールが頂点シェーディング操作のみを含むことを示し得る。１というドローモードは、ドローコールが頂点シェーディング操作とジオメトリシェーディング操作の両方を含むことを示し得る。以下でより詳細に説明されるように、他のドローモードも可能である。表１は、２つのモードを有する例示的なモードの表を与える。

【表１】

表1:モード情報 マージされたVS／GS

モード	モード0 GS:オフ	モード1 GS:オン
フロー	VS->PS	VS| GS->PS
インデックス(32ビット)	頂点インデックス(VS)	頂点インデックス(VS)
PrimitiveID(32ビット)	使用されない	*PrimitiveID* *(GS)*
Misc(25ビット)	使用されない	misc->rel_primID (4:0)
		misc->rel_vertex (9:5)
		misc->GsInstance (14:10)
		misc->Gsoutvertex (24:15)
Vs__valid (1ビット)		
Gshs__valid (1ビット)		
モード (2:0)	モード= モード_0	モード= モード_1

【００９８】

[0118]上の表１の例では、「フロー」は、それぞれのモードと関連付けられる操作（ＧＰＵ　３６によって実行されるような）のフローを示す。たとえば、モード０は、頂点シェーディング（ＶＳ）操作とピクセルシェーディング（ＰＳ）操作とを含む。したがって、ＧＰＵ　３６は、モード０のドローコールを実行すると、頂点シェーディング操作とピクセルシェーディング操作とを実行するようにシェーディングユニット４０を指定することができる。表１のモード１は、頂点シェーディング操作およびピクセルシェーディング操作、さらには、ジオメトリシェーディング（ＧＳ）操作を含む。

【００９９】

[0119]したがって、ＧＰＵ　３６は、頂点シェーディング操作とピクセルシェーディング操作とを実行するようにシェーディングユニット４０を指定することができる。しかしながら、ＧＰＵ　３６はまた、頂点シェーダ操作を実行することを担う同じシェーディングユニット４０によってジオメトリシェーダ操作が実行されるように、頂点シェーダ命令にジオメトリシェーダ命令を付加することができる。「misc」ビットは、同じシェーディングユニット４０が連続して複数の異なるシェーダを実行することを可能にするために使用される変数（たとえば、rel_primID、rel_vertex、GsInstance、Gsoutvertex）のために確保される。

【０１００】

[0120]図４の例では、同じシェーディングユニット４０はまた、プリミティブの第１の

セット１２０を入力として使用して、頂点Ｖ０～Ｖ５を有するプリミティブの第２のセット１２４（トライアングルストリップと呼ばれ得る）を生成する。頂点Ｖ０～Ｖ５を生成するために、頂点シェーディングのために指定されたシェーディングユニット４０は、メモリシェーダ操作の複数のインスタンス（たとえば、出力識別子（outID）によって図示され、同じジオメトリシェーダプログラムの異なるインスタンスとしても言及され得る）を実行する。ジオメトリシェーダ操作の各インスタンスは、同じアルゴリズムを実行して、同じジオメトリシェーディング操作を実行し、１つまたは複数の新たな頂点Ｖ０～Ｖ５のそれぞれのインスタンスを生成する。

【０１０１】

[0121]図４に示される表の８個の列は、ジオメトリシェーダ操作（またはプログラム）の８個の別個のインスタンスに対応し、各列は左から右へ、０～７のジオメトリシェーダ操作outIDによって識別され得る。入力されたプリミティブごとのマージされたＶＳ／ＧＳの出力の数は、dcl_maxoutputvertexcount*GSInstancecountに等しくてよく、各ＶＳ／ＧＳの出力は、ジオメトリシェーダステージから放出される１つの頂点である。ジオメトリシェーダステージの出力される頂点の数がdcl_maxoutputvertexcountより少ない例では、その出力される頂点は、以下でより詳細に説明されるように、条件に応じて廃棄または省略され得る（「消滅させられる」と呼ばれ得る）。したがって、各ファイバーは、MaxVertexOutputによって規定されるジオメトリシェーダの出力される頂点ごとの、頂点シェーダの１回の呼出しおよびそれに続くジオメトリシェーダの１回の呼出しに対応する。

【０１０２】

[0122]図４に示される例では、ジオメトリシェーダ操作の８個のインスタンスの各々は、頂点シェーディング操作のために指定されたのと同じシェーディングユニット４０によって、しばしば同時に付加され実行されて、１つまたは複数の新たな頂点の別個のインスタンスを生成する。したがって、ジオメトリシェーダ操作のインスタンスの各々は、６個すべての頂点（Ｖ０～Ｖ５）を生成するが、６個の新たな頂点の対応する１つのみを出力する。ジオメトリシェーダ操作の各インスタンスは、頂点シェーディング操作を実行するようにシェーディングユニット４０を呼び出すことと関連付けられる１：１のインターフェースを堅持するために、６個の新たな頂点の対応する１つのみを出力する。

【０１０３】

[0123]図４の例で示されるように、ジオメトリシェーダ操作の各々は、そのoutIDと一致する６個の新たな頂点の１つを出力する。したがって、outID = 0を有するジオメトリシェーダ操作の第１のインスタンスは、６個の新たな頂点のうちの第１の頂点、Ｖ０を出力する。outID = 1を有するジオメトリシェーダ操作の第２のインスタンスは、６個の新たな頂点のうちの第２の頂点、Ｖ１を出力する。outID = 2を有するジオメトリシェーダ操作の第３のインスタンスは、６個の新たな頂点のうちの第３の頂点、Ｖ２を出力する。outID = 3を有するジオメトリシェーダ操作の第４のインスタンスは、６個の新たな頂点のうちの第４の頂点、Ｖ３を出力する。outID = 4を有するジオメトリシェーダ操作の第５のインスタンスは、６個の新たな頂点のうちの第２の頂点、Ｖ４を出力する。outID = 5を有するジオメトリシェーダ操作の第６のインスタンスは、６個の新たな頂点のうちの第６の頂点、Ｖ５を出力する。

【０１０４】

[0124]ジオメトリシェーダ操作は６個の新たな頂点のみを生成し、ジオメトリシェーダ操作の第７および第８のインスタンスのｏｕｔＩＤは６個の新たな頂点のいずれにも対応しないので、ジオメトリシェーダ操作の第７および第８のインスタンスは、「消滅させられ」、または終了させられる。したがって、シェーディングユニット４０は、ジオメトリシェーダ操作のこれらのインスタンスと関連付けられる対応する頂点がないと判定すると、ジオメトリシェーダ操作の第７および第８のインスタンスの実行を終了する。

【０１０５】

[0125]以下に示される表２は、頂点シェーディング操作とジオメトリシェーディング操作とを実行するためにＧＰＵ　３６によって保持され得るいくつかのパラメータを示す。

【表２】

表2:VS／GSのためのパラメータ

フロー	VS| GS->PS
インデックス(32ビット)	頂点インデックス(VS)
uv__msb(2ビット)	使用されない
PrimitiveID(32ビット)	*PrimitiveID(GS)*
Rel__patchid(32ビット)	使用されない
Misc(25ビット)	misc-> rel_primID (4:0)
	misc-> rel_vertex (9:5)
	misc-> GsInstance (14:10)
	misc-> Gsoutvertex (24:15)
Vs__valid(1ビット)	
Gshs__valid(1ビット)	
モード(2:0)	モード= モード_1
Instance__cmd(2ビット))	

【０１０６】

[0126]表２に示されるいくつかのパラメータ（たとえば、ｕｖ＿ｍｓｂ、Ｒｅｌ＿ｐａｔｃｈｉｄ）は、ＶＳ／ＧＳ操作のために使用されず、以下でより詳細に説明される。表２の例では、インデックスは、頂点の相対的なインデックスを示す。ＰｒｉｍｉｔｉｖｅＩＤは、関連する頂点のプリミティブを識別するために、ジオメトリシェーディング操作の間に使用されるプリミティブＩＤを示し、システムにより生成される値（たとえば、ＧＰＵ　３６の１つまたは複数のハードウェアコンポーネントによって生成される）であり得る。上で述べられるように、Ｍｉｓｃは、ＶＳ操作の後にＧＳ操作を実行するための、確保されたキャッシュの値を示す。たとえば、以下で示される表３は、図４に関して上で説明された頂点シェーディング操作とジオメトリシェーディング操作とを実行するときのパラメータ値を示す。

【表３】

表3:VS／GS操作のためのパラメータ値

モード1 GS:オン	ファイバー0	ファイバー1	ファイバー2	ファイバー3	ファイバー4	ファイバー5	ファイバー6	ファイバー7
Valid_as_input	1	1	1	0	0	0	0	0
頂点インデックス(VS)	V0	V1	V2	0	0	0	0	0
primitiveID (GS)	5	5	5	5	5	5	5	5
Valid_as_output	1	1	1	1	1	1	1	1
misc-> rel_primID (4:0)	2	2	2	2	2	2	2	2
misc-> rel_vertex (9:5)	0	1	2	0	0	0	0	0
misc-> GsInstance (14:10)	0	0	0	0	0	0	0	0
misc-> Gsoutvertex (24:15)	0	1	2	3	4	5	6	7

【０１０７】

[0127]多数のファイバー（たとえば、命令）が頂点シェーディング操作とジオメトリシェーディング操作とを実行するために割り振られるが、いくつかの例では、ＧＰＵ　３６は、ファイバーのサブセットのみを実行し得る。たとえば、ＧＰＵ　３６は、シェーディングユニット４０を用いて命令を実行する前に、命令が有効かどうかを判定することができる（上の表３に示されるvalid_as_input）。割り振られたファイバーのうちの３つのみがシェーディングされた頂点を生成するために使用されるので、ＧＰＵ　３６は、頂点シェーディング操作を実行するときに残りのファイバー（上の表３のファイバー３～７）を実行しなくてよく、このことは電力を節減し得る。以下でより詳細に説明されるように、ＧＰＵ　３６は、マスク（たとえば、以下の図５Ｂのcov_mask_1）に基づいて、どのファイバーが実行されるべきかを判定することができる。

【０１０８】

[0128]いくつかのＡＰＩ（たとえば、ＤｉｒｅｃｔＸ　１０　ＡＰＩ）は、ジオメトリシェーダステージからのいわゆる「ストリームアウト」を提供し、ここでストリームアウトは、新たな頂点がジオメトリシェーダに再び入力され得るように、ストレージユニット４８のようなメモリへとジオメトリシェーダからこれらの新たな頂点を出力することを指す。

【０１０９】

[0129]本技法は、ハードウェアユニットが、ジオメトリシェーダ操作を実行することで得られる新たな頂点をストレージユニット４８に出力することを可能にすることによって、このストリームアウトの機能に対するサポートを提供することができる。このストリー

ムアウトを介して出力される新たな頂点は、ラスタライザによって予想されるフォーマットではなく、予想されるジオメトリシェーダフォーマットで規定される。ハードウェアユニットは、これらの新たな頂点を取り出し、既存のジオメトリシェーダ操作を実施し続け、または、この状況では「ストリームアウト頂点」と呼ばれ得る、これらの頂点に対する新たなジオメトリシェーダ操作を実施し続けることができる。このようにして、本技法は、比較的限られた数のシェーディングユニット４０を有する、ＧＰＵ　３６のようなＧＰＵが、より多くのシェーディングユニットを有するＧＰＵをエミュレートすることを可能にし得る。

【０１１０】

[0130]図５Ａおよび図５Ｂは、本開示の技法を実施するハードウェアシェーディングユニットによって実行され得る例示的な動作を示す。たとえば、図５Ａは一般に、頂点シェーディング操作とジオメトリシェーディング操作とを実行するときにマージされたＶＳ／ＧＳハードウェアシェーディングユニットによって実行される動作のフローを示す。マージされたＶＳ／ＧＳハードウェアシェーディングユニットは、いくつかの例では、頂点シェーディング操作を実行するようにＧＰＵ　３６によって指定されるが、本開示の技法に従って頂点シェーディング操作とハードウェアシェーディング操作の両方を実行する、シェーディングユニット４０を含み得る。

【０１１１】

[0131]図５Ｂは、マージされたＶＳ／ＧＳハードウェアシェーディングユニットによって実行され得る、図５Ａに示される動作のフローに対応する擬似コードを一般に示す。図５Ａおよび図５Ｂのいくつかの態様はＧＰＵ　３６（図１）に関して説明され得るが、本開示の態様は、種々の他のコンポーネントを有する多種多様な他のＧＰＵによって実行され得る。

【０１１２】

[0132]図５Ａに示される例では、マージされたＶＳ／ＧＳハードウェアシェーディングユニットは、頂点の属性、vertex_id、nstance_id、primitive_id、miscのようなシステム値を、一連のレジスタＲ０、Ｒ１、およびＲ２に書き込む（１４０）。通常、システム値は、ＧＰＵの任意のそれ以外には割り振られないメモリに記憶され得る。システムにより生成された値を所定の位置にある一連のレジスタに記憶することによって、ＧＰＵ　３６は、ＶＳステージおよびＧＳステージの各々に対する、システムにより生成された値にアクセスすることができる。したがって、ＧＳステージは、システムにより生成された値がどこに記憶されたかを判定するために、ＶＳステージに基づいてコンパイルされる必要がない。むしろ、ＧＰＵ　３６は、ステージの各々を実行するときに所定のメモリ位置にアクセスして、システムにより生成された必要とされる値にアクセスすることができる。

【０１１３】

[0133]マージされたＶＳ／ＧＳハードウェアユニットが次いで、頂点シェーディング操作を実行する（１４２）。頂点シェーディング操作に続いて、マージされたＶＳ／ＧＳハードウェアシェーディングユニットが、汎用レジスタ（ＧＰＲ）の内容（たとえば、頂点シェーディング操作からのプリミティブの頂点）を、ＧＰＵメモリ３８などのローカルメモリに書き込むことができる。図５Ｂに関して以下でより詳細に説明されるように、マージされたＶＳ／ＧＳハードウェアシェーディングユニットは次いで、ＧＳのテクスチャおよび定数のオフセットに（１４６）、かつＧＳプログラムカウンタに（１４８）切り替えることができる。

【０１１４】

[0134]マージされたＶＳ／ＧＳハードウェアシェーディングユニットは、頂点シェーディング操作からのプリミティブの頂点のような、ローカルメモリの内容を読み取り、ジオメトリシェーディング操作を実行することができる（１５０）。マージされたＶＳ／ＧＳハードウェアシェーディングユニットは、１つの頂点の属性を頂点パラメータキャッシュ（ＶＰＣ）に出力し、さらに、ジオメトリシェーディングされた頂点の位置の指示、ｓｔｒｅａｍ＿ｉｄ、任意の切断の指示、および任意の変換された値を、位置キャッシュに出

力することができる。

【０１１５】

[0135]図５Ｂは、マージされたＶＳ／ＧＳハードウェアシェーディングユニットによって実行され得る、図５Ａに示される動作のフローに対応する擬似コードを一般に示す。各シェーダステージは、（たとえば、特定のステージが別のステージとどのようにリンクされるかの知識を伴わずに）、別々に、かつ独立にコンパイルされ得る。単一のハードウェアシェーディングユニットが複数のシェーディング操作を実行することを可能にするために、ハードウェアシェーディングユニットは、ローカルメモリにおいていくつかの位置を確保することができる。たとえば、ハードウェアシェーディングユニットは、シェーダステージ（ＶＳまたはＧＳ）の両方によってアクセスされ得るローカルメモリ中の位置を確保することができる。いくつかの変数（たとえば、PrimitiveID、misc、およびrel_patch id）は、２つ以上のシェーダステージによって使用され得る。したがって、ローカルメモリにおいて確保された位置は、２つ以上のシェーダステージによってアクセスされ得る、一般に使用される変数に対する標準化された位置を提供する。

【０１１６】

[0136]図５Ｂに示される例では、ハードウェアシェーディングユニットは最初に、頂点シェーディング操作（ＶＳ）（図５Ａの例ではステップ１４０～１４２に対応し得る、上から下への第１の破線のボックスに含まれる）を実行することができる。本開示の態様によれば、ハードウェアシェーディングユニット（またはＧＰＵの別のコンポーネント）は次いで、いわゆる「パッチコード」を実行して、頂点シェーディング操作からジオメトリシェーディング操作（図５Ａの例ではステップ１４４～１４８に対応し得る、上から下への第２の破線のボックスに含まれる）への切り替えを開始することができる。より具体的には、コマンドＣＨＭＳＫおよびＣＨＳＨは、ハードウェアシェーディングユニットに、（上で説明されたような）実行されているドローコールのモードに従って、動作モードを切り替えさせることができる。

【０１１７】

[0137]たとえば、ハードウェアシェーディングユニットは、頂点シェーディング操作からローカルＧＰＵメモリに頂点データを書き込めるので、シェーディングされる頂点は、ジオメトリシェーディング操作を実行するときに利用可能である。ハードウェアシェーディングユニット（またはＧＰＵの別のコンポーネント）は次いで、ジオメトリシェーディング操作のためのハードウェアシェーディングユニットのリソースを切り替えるマスク変更（ＣＨＭＳＫ）命令を実行する。たとえば、ＣＨＭＳＫ命令を実行することは、ハードウェアシェーディングユニットに、どのモードが現在実行されているかを判定させ得る。

【０１１８】

[0138]上の表２に関して、ＣＨＭＳＫを実行することはまた、ハードウェアシェーディングユニットに、どのシェーダステージが有効か（たとえば、vs_valid、gs_validなど）を判定させ得る。上で述べられたように、ＧＰＵ　３６は、頂点シェーディング操作とジオメトリシェーディング操作とを実行するための多数のファイバーを割り振ることができる。しかしながら、ＣＨＭＳＫを実行すると、ＧＰＵ　３６は、ファイバーのサブセットのみを実行することができる。たとえば、ＧＰＵ　３６は、シェーディングユニット４０を用いて命令を実行する前に、命令が有効かどうかを判定することができる。ＧＰＵ　３６は、有効ではないファイバーを実行しなくてよく（たとえば、シェーディングされた頂点を生成しない）、このことは電力を節減し得る。

【０１１９】

[0139]ハードウェアシェーディングユニットはまた、シェーダ変更（ＣＨＳＨ）命令を実行して、ジオメトリシェーディング操作を実行するためにプログラムカウンタ（ＰＣ）を適切な状態オフセットへと切り替える。以下でより詳細に説明されるように、このパッチコード（図５Ａの例ではステップ１４４～１４８に対応し得る、上から下への第２の破線のボックスに含まれる）は、どのシェーダステージがマージされているかどうかに関係なく同一であり得る。

【０１２０】
[0140]パッチコードを実行した後で、ハードウェアシェーダユニットは、頂点シェーディング操作を停止し、ジオメトリシェーディング操作（図５Ａの例ではステップ１５０に対応する、上から下への第３の破線のボックスに含まれる）を実行する。通常、複数のシェーディング操作を実行するハードウェアシェーディングユニットによって実行されるシェーダ（シェーディング操作を実行するためのコード）は、シェーダの依存関係に基づく再コンパイルを必要とし得る。たとえば、primitiveID（システムにより生成される値）がＧＳステージによって使用される場合、ＶＳステージは、ＧＳステージがそこから値を選ぶことができる位置に、primitiveID値を置くように（たとえば、コンパイラ５４によって）コンパイルされ得る。したがって、ＶＳステージのコンパイルは、ＧＳステージの必要性に依存し得る。

【０１２１】
[0141]本開示の態様によれば、シェーダの各々は、他のシェーダとは関係なく独立にコンパイルされ得る。たとえば、シェーダは、他のシェーダがいつ実行されかの知識を伴わずに、独立にコンパイルされ得る。コンパイルの後、ＧＰＵ　３６は、ドローのときに実行されているドローコールと関連付けられるモード情報に基づいて、図５Ｂに示されるパッチコードを使用して、シェーダを一緒にパッチされることができる。システムにより生成される値であるvertexIDおよびinstanceIDのみが、頂点シェーダにおいて使用されてよく、ＶＳステージをコンパイルすることによって計算されるような規定された汎用レジスタスロット（ＧＰＲ）にロードされ得る。しかしながら、primitiveIDおよび、miscおよびrel_patchidのような、プリミティブコントローラ（ＰＣ）からの他のマージシェーダ関連の値（たとえば、図６に示されるような）は、シェーダステージのいずれかによって使用され得る。

【０１２２】
[0142]上で説明されたパッチコードは、ＧＰＵドライバ５０のようなＧＰＵ　３６のためのドライバによって、コンパイルされたシェーダに追加され得る。たとえば、ＧＰＵドライバ５０は、どのシェーダが各ドローコールに対して必要とされるかを判定する。ＧＰＵドライバ５０は、図５Ｂに示されるパッチコードを、いわゆるドライバ時間またはリンク時間において適切なシェーダ（マージされているシェーダ）に付加することができ、これによって、シェーダが同じハードウェアシェーディングユニットによって実行されるように、コンパイルされたシェーダをリンクする。ＧＰＵドライバ５０は、シェーダ全体を再コンパイルする必要がなく、これによって、計算リソースを節約する。

【０１２３】
[0143]このようにして、ＧＰＵ　３６は、複数のモードを使用するシェーディング操作を一緒にパッチされることができ、各モードは、関連するシェーディング操作の特定のセットを有する。そのような技法は、ＧＰＵ　３６が、シェーディングユニット４０を再構成することなく、追加のシェーディング操作（たとえば、ジオメトリシェーディング操作、ハルシェーディング操作、および／またはドメインシェーディング操作）を実行することを可能にし得る。すなわち、本技法は、シェーディングユニット４０が、他のシェーディング操作を実行しながら、いくつかのシェーダステージの入力／出力の制約を堅持することを可能にし得る。

【０１２４】
[0144]図６は、本開示の態様による、マージされた頂点シェーディング操作とジオメトリシェーディング操作とを実行するためのグラフィックスプロセシングユニット１７８の例示的なコンポーネントを示す図である。図６の例は、マージされたＶＳ／ＧＳユニット１８０と、頂点パラメータキャッシュ（ＶＰＣ）１８２と、プリミティブコントローラ（ＰＣ）１８４と、頂点フェッチデコーダ（ＶＦＤ）１８６と、グラフィックスラスタライザ（ＧＲＡＳ）１８８と、レンダーバックエンド（ＲＢ）１９０と、コマンドプロセッサ（ＣＰ）１９２と、ピクセルシェーダ（ＰＳ）１９４とを含む。加えて、図６は、ＰＭ４パケットバッファ１９８と、頂点オブジェクト２００と、インデックスバッファ２０２と

、ストリームアウトバッファ２０４と、フレームバッファ２０６とを有する、メモリ１９６を含む。

【０１２５】

[0145]図６の例では、ＶＳ／ＧＳユニット１８０は、上で説明された方式で頂点シェーディング操作を実行するように指定された１つまたは複数のシェーディングユニットによって実装される。ＶＰＣ　１８２は、ストリームアウトデータをストリームアウトバッファ２０４に記憶するために、ストリームアウト機能を実装することができる。ＰＣ　１８４は、変換される必要があり得る頂点を管理することができる。たとえば、ＰＣ　１８４は、複数の頂点を三角形のプリミティブへと組み立てることができる。ＶＦＤ　１８６は、頂点のフォーマット状態に基づいて、頂点データをフェッチすることができる。ＧＲＡＳ　１８８は、入力として三角形の頂点を受け取ることができ、三角形の境界内にあるピクセルを出力することができる。プリフェッチパーサ（ＰＦＰ）は、コマンドストリームを事前に復号し、メインＣＰエンジン１９２がデータを必要とし得るときまでにそのデータの準備ができているように、そのデータをポインタ（たとえば、リソースポインタ）を介してフェッチすることができる。

【０１２６】

[0146]説明のための例では、ＤｉｒｅｃｔＸ　１０のディスパッチ機構が、図６に示されるグラフィックスプロセシングユニット１７８を使用して実装され得る。たとえば、ＤｉｒｅｃｔＸドローコールは、ＶＳ操作とＧＳ操作がマージされる、たとえば同じシェーディングユニットによって実行されることを示す、モードビット（モード情報）を有するドローインジケータを伴う、単一パスのドローコールとして扱われ得る。このモードは、ＰＣ　１８４内のGSblockが、ＧＳ出力vertexIDおよびＧＳ instanceIDを伴うＶＦＤ　１８６のためのデータを生成することを可能にする。GSblockは、宣言されたmaxoutputvertexcountおよびGSinstancecountに基づいて、入力プリミティブに対する多数のＶＳファイバーを作成する。ウェーブ中のファイバーの数（たとえば、３２個のファイバーのような、１つのシェーディングユニットによって行われる作業の量）がmaxoutputvertexcount * GSinstancecountより大きい場合、ウェーブは、完全な入力ＧＳプリミティブを有し得る。それ以外の場合、ＧＳの入力プリミティブの頂点インデックスは、maxoutputvertexcount * GSinstancecount個のファイバーが作成されるまで、次のウェーブに対して繰り返され得る。入力プリミティブの頂点に対して、頂点の再使用は必要ではない。

【０１２７】

[0147]ＶＰＣ　１８２の出力において、ＰＣ　１８４は、ＧＳの出力プリミティブのタイプに基づいて、プリミティブの接続を生成する。たとえば、（ＶＳ／ＧＳ　１８０の）ＧＳからの第１の出力された頂点は通常、この頂点の前のプリミティブ（ストリップ）の完成を示し得る、位置キャッシュ中の「切断」ビットから構成され得る。ＰＣ　１８４はまた、ＶＰＣ　１８２に対する完成したプリミティブのこの接続情報を、ストリームアウトＧＳ出力に対するＶＰＣ　１８２のstreamidとともに、所与のストリームと結び付けられたバッファ２０４に送る。ＧＳ　１８０中の複数の完全なプリミティブの間に部分的なプリミティブがある場合、そのような部分的なプリミティブは、プリミティブを脱落させるために、ＧＲＡＳ　１８８についてPRIM_AMP_DEADとして標識される。ＰＣ　１８４はまた、無効なプリミティブのタイプをＶＰＣ　１８２に送り、そのようなプリミティブに対するパラメータのキャッシュの割り振りを解除する。

【０１２８】

[0148]maxoutputvertexcountに基づいて、ＧＰＵドライバ（図１に示されるＧＰＵドライバ５０のような）は、どれだけの入力プリミティブの頂点がローカルメモリに記憶されるかを計算することができる。この入力プリミティブ値は、次の式に従って、変数GS_LM_SIZEとして計算され得る。

【数１】

ウェーブ中のファイバー＊プリミティブごとの頂点の数＊頂点のサイズ
maxoutputv ertexcount

【０１２９】
このタイプのドローコールを受け取るハイレベルシーケンサ（ＨＬＳＱ）は、どのシェーダプロセッサのローカルメモリ（ＬＭ）がGS_LM_SIZEのために十分な記憶容量を有するかを（たとえば、場合によってはラウンドロビン手法を使用して）確認することができる。HLSQは、そのような割り振りの開始基本アドレス、さらには、割り振られたウェーブによるローカルメモリに対する任意の読取りまたは書込みのアドレスを保持することができる。HLSQはまた、ローカルメモリに書き込むときに、割り振られたメモリ内の計算されたオフセットを基本アドレスに追加することができる。
【０１３０】
[0149]したがって、本開示の態様によれば、入力と出力との関係は、ＶＳ／ＧＳ１８０に対して、（頂点シェーディング操作を実行するように指定されるシェーディングユニットでは通常そうであるように）１：１ではない。むしろ、ＧＳは、各々の入力プリミティブから１つまたは複数の頂点を出力することができる。加えて、ＧＳによって出力される頂点の数は動的であり、１から、ＡＰＩにより課される最大のＧＳの出力（たとえば、１０２４個の頂点という出力の最大値に等しいことがある、１０２４ダブルワード（dwords））まで変化し得る。
【０１３１】
[0150]すなわち、ＧＳは、最小で１つの頂点、かつ最大で１０２４個の頂点を生成することができ、ＧＳからの出力全体が１０２４dwordsであり得る。ＧＳは、変数dcl_maxoutputvertexcountを使用して、ＧＳからの出力された頂点の最大の数を、コンパイル時に宣言することができる。しかしながら、出力される頂点の実際の数は、ＧＰＵ　３６がＧＳを実行する時点では知られていないことがある。むしろ、dcl_maxoutputvertexcountの宣言は、ＧＳに対するパラメータとしてのみ必要とされ得る。
【０１３２】
[0151]ＧＳはまた、入力プリミティブごとに呼び出されるべきＧＳのインスタンス（操作）の数に対する変数instancecountを宣言することができる。この宣言は、ＧＳの呼出しのための外側のループとして機能し得る（ジオメトリシェーダインスタンスの最大の数を特定する）。最大のinstancecountは３２に設定され得るが、他の値も使用され得る。したがって、ＧＳは、ジオメトリシェーダ操作における変数GSInstanceIDへのアクセス権を有し、それは、所与のＧＳがどのインスタンスに対して行われているかを示す。ＧＳのインスタンスの各々は、最大で１０２４dwordsを出力することができ、各々は、最大の出力される頂点の数として、dcl_maxoutputvertexcountを有し得る。加えて、各々のＧＳのインスタンスは、他のＧＳのインスタンスとは独立であり得る。
【０１３３】
[0152]ＧＳの入力においてＧＰＵ　３６が宣言し得る入力プリミティブのタイプは、点、線、三角形、隣接を伴う線、隣接を伴う三角形、およびパッチ１～３２であり得る。隣接を伴う三角形は、ＤｉｒｅｃｔＸ　１０のようないくつかのＡＰＩの新たな機能であり得る。加えて、パッチ１～３２は、ＤｉｒｅｃｔＸ　１１　ＡＰＩに対して追加されたさらなる改善であり得る。ＧＳからの出力プリミティブのタイプは、点、ラインストリップ、またはトライアングルストリップであり得る。ＧＳの出力は、ＧＳにおいて宣言され得る４つのストリームのうちの１つに向かうことができ、ＧＳは、どれだけのストリームが使用されるかを宣言し得る。一般に、「ストリーム」は、（たとえば、メモリバッファに）記憶される、またはラスタライザのようなＧＰＵの別のユニットに送られる、シェーディングされたデータを指す。各頂点の「放出」命令は、頂点がどのストリームに向かっているかを示し得る「放出ストリーム」の指定を使用することができる。

【０１３４】
[0153]ＧＳは、「ストリーム切断」命令または「ストリーム放出後切断」命令を使用して、ストリップのプリミティブのタイプを完成させることができる。そのような例では、次の頂点は、所与のストリームに対する新たなプリミティブを開始する。いくつかの例では、プログラマは、（ＡＰＩを使用して）ストリームをセットアップするときにラスタライズされたストリームとして使用されるべき、多くとも１つのストリームを宣言することができる。加えて、４つの１Ｄバッファが１つのストリームと結び付けられ得るが、ＧＳストリームのすべてと結び付けられるバッファの総数は、４を超えなくてよい。オフチップバッファは通常、複数のストリームの間で共有されない。
【０１３５】
[0154]頂点が所与のストリームに対して放出される場合、ストリームと結び付けられる各バッファに対する頂点のサブセクションは、完成したプリミティブとしてオフチップバッファ（ストレージユニット４８のような）に書き込まれる。すなわち、部分的なプリミティブは一般に、オフチップバッファに書き込まれない。いくつかの例では、オフチップバッファに書き込まれるデータは、プリミティブのタイプの指示を含むように拡張されてよく、２つ以上のストリームが所与のＧＳに対して可能にされる場合、ＧＳに対する出力プリミティブのタイプは「点」のみであり得る。
【０１３６】
[0155]ＧＳステージは、PrimitiveIDパラメータを入力として受け取ることができ、それは、PrimitiveIDがシステムにより生成される値であるからである。ＧＳはまた、PrimitiveIDパラメータと、iewportIndexパラメータと、RenderTargetArrayIndexパラメータとを１つまたは複数のレジスタに出力することができる。ＧＳの入力に対する補間モードという属性は通常、定数として宣言される。いくつかの例では、ＧＳをＮＵＬＬとして宣言しながら、依然として出力を可能にすることが可能である。そのような例では、ストリーム０のみがアクティブであり得る。したがって、ＶＳの出力は、プリミティブのタイプを一覧にするように拡張されてよく、ストリーム０と結び付けられたバッファに値を書き込むことができる。入力プリミティブのタイプが隣接というプリミティブのタイプであると宣言される場合、隣接する頂点情報が脱落させられ得る。すなわち、たとえば、隣接したプリミティブの内部の頂点のみ（たとえば、偶数の頂点番号）が、非隣接というプリミティブのタイプを形成するために処理され得る。
【０１３７】
[0156]パッチの入力プリミティブのタイプがＮＵＬＬ　ＧＳを伴う場合、パッチは、ストリームと結び付けられたバッファに、点のリストとして書き出される。宣言されたストリームもラスタライズされる場合、ＧＰＵ　３６は、パッチ制御ポイントによって規定されるような、複数の点としてパッチをレンダリングすることができる。加えて、ＧＳがＮＵＬＬである場合、viewportindexパラメータおよびrendertargetarrayindexパラメータは０であると仮定され得る。
【０１３８】
[0157]クエリカウンタは、どれだけのＶＳ操作またはＧＳ操作がＧＰＵ　３６によって処理されているかを決定するために実装されてよく、これによって、ハードウェアコンポーネントがプログラムの実行を記録することが可能になる。クエリカウンタは、stat_startイベントおよびstat_endイベントに基づいて、カウントを開始し停止することができる。カウンタは、stat_sampleイベントを使用してサンプリングされ得る。stat_start ｔイベントおよび／またはstat_stopイベントを受け取る動作ブロックは、インクリメント信号が送信されそのようなイベントを受け取る様々な点において、カウントを開始または停止する。
【０１３９】
[0158]ＧＰＵ　３６のドライバがそのようなカウンタを読み取る必要がある場合、ドライバは、図５Ｂに関して示され説明されるように、コマンドプロセッサ（ＣＰ）を通じてstat_sampleイベントを送ることができる。レジスタバックボーン管理（ＲＢＢＭ）ユニ

ットが、カウンタをインクリメントすることを担う動作ブロックから確認応答（または「ａｃｋ」）を受け取るまで、ＣＰは、任意の追加のドローコールをＧＰＵ　３６に送るのを控えることができる。「ａｃｋ」が受け取られると、ＲＭＭＢユニットは、カウンタを読み取り、次のドローコールの送信を再開することができる。

【０１４０】

[0159]ＧＰＵ　３６は、種々のデータをローカルＧＰＵメモリ３８に記憶することができる。たとえば、次のクエリカウントは、ハードウェア中のＣＰによって保持され得る。いくつかの例では、次のクエリカウントは６４ビットカウンタとして形成されてよく、これは、以下に示されるように、様々な動作ブロックからの１～３ビットのパルスを使用してインクリメントされ得る。

・IAVerticesは、プリミティブを生成する際に使用される頂点の数を指し得る。したがって、入力プリミティブのタイプが、三角形を生成するストリップである場合、IAVerticesは６であり得る。この値は、Ｗｉｎｄｏｗｓ（登録商標）　Ｈａｒｄｗａｒｅ　Ｑｕａｌｉｔｙ　Ｌａｂｓ（ＷＨＱＬ）の数字と一致し得る。この値は、プリミティブコントローラ（ＰＣ）からの２ビットのパルスを使用して制御され得る。パッチのプリミティブのために、値は、制御ポイントごとに１だけインクリメントされ得る。

・IAPrimitivesは、生成された完成した入力プリミティブの数を指し得る。この値は、リセットをもたらし得る部分的なプリミティブを何ら含まなくてよい。この値は、ＷＱＨＬの数字と一致し得る。この値は、プリミティブが生成された後で、さらには、リセットインデックスと部分的なプリミティブの脱落とを確認した後で、ＰＣからの１ビットのパルスを使用して制御され得る。

・VSInvocationsは、ＶＳ操作が呼び出される回数を指し得る。この値は頂点の再使用の後に設定されてよく、ＶＳステージが呼び出される対象である固有の頂点の数を決定し得る。この値は、ＧＰＵ　３６の具体的なハードウェアに依存し得る。この値は、一度に最大で３つの頂点に対する頂点の再使用をＰＣが確認すると、ＰＣからの２ビットのパルスを使用して制御され得る。ＧＳおよびハルシェーダ（ＨＳ）（たとえば、図１２Ａ～図１３Ｂに関して以下で説明されるような）の場合に対しては通常、頂点の再使用はない。したがって、ＰＣは、VSInvocationsとして、ドローコール中のプリミティブにおける頂点の数を送ることができる。

・HSInvocationsは、ＨＳを経たパッチの数を指し得る。この値は、ＤｉｒｅｃｔＸ　１１のようないくつかのＡＰＩの新たな値であり得る。この値は、部分的なパッチを何ら含まなくてよい。この値は、パッチが頂点フェッチデコーダ（ＶＦＤ）に完全に送られると、ＰＣおよびＨＳブロックからの１つのビットパルスを使用して制御され得る。この値はまた、ＷＨＱＬの数字と一致すべきである。

・DSInvocationsは、ドメインシェーダ（ＤＳ）操作が呼び出される回数を指し得る。この値は、テッセレーションの出力プリミティブのタイプが点というタイプである場合、ＷＨＱＬと一致すべきである。この値は、生成されている各ドメイン点（ｕ，ｖ）に対して、ＰＣ中のテッセレーションエンジン（ＴＥ）からの１ビットのパルスを使用して制御される。

・GSInvocationsは、ＧＳ操作が呼び出される回数を指し得る。GSinstancecount値が使用される場合、各インスタンスは、１つのＧＳ呼出しとしてカウントされる。この値は、ＷＨＱＬの数字と一致すべきである。この値は、Gsinstanceごとに、入力プリミティブごとに一度送られる、ＧＳブロックからの１ビットのパルスを使用して制御され得る。いくつかの例では、ＧＳブロックは、ＧＳ増幅がウェーブサイズより大きい場合、入力ＧＳプリミティブを複数回送ることができる。この値は通常、ＧＳ入力プリミティブごとに一度カウントされる。

・GSPrimitivesは、生成されるＧＳ出力プリミティブの数を指し得る。この値は、「切断」操作に起因する部分的なプリミティブを何ら含まなくてよい。この値は、ＷＨＱＬの数字と一致し得る。この値は、プリミティブが構成される位置キャッシュへのアクセスの後で、かつ、「切断」操作または頂点消滅イベントが原因で部分的なプリミティブが脱

落した後で、ＰＣからの出力プリミティブごとに１ビットのパルスを使用して制御され得る。

・CInvocationsは、いわゆる「クリッパー」が実行される回数を指し得る。この値は、ＧＰＵ　３６の具体的なハードウェアに依存し得る。

・CPrimitivesは、クリッパーが生成したプリミティブの数を指し得る。この値は、ＧＰＵ　３６の具体的なハードウェアに依存し得る。

・PSInvocationsは、ピクセルシェーダ（ＰＳ）のスレッド（「ファイバー」とも呼ばれ得る）が呼び出される回数を指し得る。

・CSInvocationsは、計算ファイバーが呼び出される回数を指し得る。

【０１４１】

[0160]上で説明された値に加えて、ストリームごとに保持される、２つのストリームアウト関連のクエリカウントがあり得る。これらのストリームアウト関連の値は、次の値を含み得る。

・NumPrimitiveWrittenは、ドローコールが終了する前に、所与のストリームに対して書かれるプリミティブの総数を指し得る。この値はまた、完成したプリミティブのためのバッファの記憶容量がなくなったときに、ストリームと結び付けられるバッファのデータを含み得る。この値は、完成したプリミティブを記憶するための空間が所与のストリームのバッファのいずれかに存在するたびに、頂点パラメータキャッシュ（ＶＰＣ）からＣＰへのストリームごとの１ビットのパルスを使用して制御され得る。

・PrimitiveStorageNeededは、ストリームと結び付けられるいずれのバッファの記憶容量もなくなっていなければ書き込まれることが可能であったであろう、プリミティブの総数を指し得る。この値は、ストリームに対するプリミティブがＧＳによって生成されるたびに、ＶＰＣからＣＰへのストリームごとに１ビットのパルスを使用して制御され得る。

【０１４２】

[0161]通常、ＧＰＵ　３６は、ＶＰＣから直接、ストリームアウトをサポートすることができる。上で述べられたように、ＧＳによってサポートされる最大で４つのストリームがあり得る。これらのストリームの各々は、最大で４つのバッファに束縛されることがあり、バッファは通常、異なるストリームの間で共有可能ではない。各バッファへの出力のサイズは、最大で１２８dwordsであってよく、これは頂点の最大サイズと同じである。しかしながら、ストライドは最大で５１２dwordsであり得る。ストリームからの出力データは複数のバッファに記憶され得るが、データは一般に、バッファ間で複製され得ない。説明のための例では、「color.x」がストリームと結び付けられたバッファの１つに書き込まれる場合、この「color.x」は、同じストリームと結び付けられた別のバッファに送られなくてよい。

【０１４３】

[0162]バッファへのストリームアウトは、完成したプリミティブとして実行され得る。すなわち、たとえば、２つのみの頂点に対する所与のストリームのための空間がいずれかのバッファにあり、プリミティブのタイプが三角形である（たとえば、３つの頂点を有する）場合、プリミティブの頂点は、そのストリームと結び付けられるいずれのバッファにも書き込まれなくてよい。

【０１４４】

[0163]ＧＳがヌルであり、ストリームアウトが有効にされる場合、ストリームアウトは、デフォルトのストリーム０として識別され得る。ストリームアウトが実行されているとき、位置情報は、ＶＰＣ、さらにはＰＣにも書き込まれることがあり、これはさらなるスロットを消費し得る。加えて、ビニングが実行されるとき（たとえば、タイルベースのレンダリングのために頂点をビンに割り当てるプロセス）、ストリームアウトはビニングパスの間に実行され得る。

【０１４５】

[0164]ＤｉｒｅｃｔＸ　１０のようないくつかのＡＰＩでは、ストリームアウトデータ

を消費する、DrawAuto機能（以前に作成されたストリームをパッチしてレンダリングし得る）が規定され得る。たとえば、ＧＰＵドライバは、メモリアドレスとともに、所与のストリームに対するストリームアウトフラッシュのイベントを送ることができる。ＶＰＣは、そのようなイベントを受け取ると、ＲＢＢＭに確認応答（ａｃｋ）ビットを送ることができる。ＲＢＢＭは、ａｃｋビットを受け取ると、バッファにおいて利用可能なバッファ空間の量（バッファリングされた満杯のサイズ）を、ドライバにより規定されるメモリまたはメモリ位置に書き込む。

【０１４６】

[0165]その間、コマンドプロセッサ（ＣＰ）内に含まれ得るプリフェッチパーサ（ＰＦＰ）は、任意のドローコールの送信を待機する。メモリアドレスが書き込まれると、ＰＦＰは次いで、次のドローコールを送ることができる。次のドローコールが自動ドローコールである場合、ＧＰＵドライバは、ドローコールと状態の変化とを示すパケット（たとえば、いわゆる「ＰＭ４」パケット）の一部として、満杯のバッファサイズを含むメモリアドレスを送ることができる。ＰＦＰは、そのメモリ位置からbuffer_filled_sizeを読み取り、ドローコールをＰＣに送る。

【０１４７】

[0166]図７は、本開示の態様による、頂点シェーディング操作とジオメトリシェーディング操作とを実行するための例示的なプロセスを示すフローチャートである。ＧＰＵ　３６（図１）によって実行されるものとして説明されるが、図７に関して説明される技法は、種々のＧＰＵまたは他のプロセシングユニットによって実行され得ることを理解されたい。

【０１４８】

[0167]ＧＰＵ　３６は最初に、たとえば、頂点シェーダ命令を受け取ると、頂点シェーディング操作を呼び出すことができる（２１０）。頂点シェーディング操作を呼び出すことで、ＧＰＵ　３６は、頂点シェーディング操作のために１つまたは複数のシェーディングユニット４０を指定し得る。加えて、ＧＰＵ　３６の他のコンポーネント（頂点パラメータキャッシュ、ラスタライザなどのような）は、指定されたシェーディングユニット４０の各々からの入力ごとに、単一の出力を受け取るように構成され得る。

【０１４９】

[0168]ＧＰＵ　３６は、頂点シェーディング操作のために指定されたハードウェアシェーディングユニットによって、頂点シェーディング操作を実行して、入力される頂点をシェーディングすることができる（２１２）。すなわち、ハードウェアシェーディングユニットは、頂点シェーディング操作を実行して、入力された頂点をシェーディングし、頂点シェーディングされたインデックスを出力することができる。ハードウェアシェーディングユニットは、１つの頂点を受け取り、１つのシェーディングされた頂点を出力することができる（たとえば、入力と出力との間の１：１の関係）。

【０１５０】

[0169]ＧＰＵ　３６は、ジオメトリシェーディング操作を実行するかどうかを判定することができる（２１４）。ＧＰＵ　３６は、たとえば、モード情報に基づいて、そのような判定を行うことができる。すなわち、ＧＰＵ　３６は、パッチコードを実行して、任意の有効なジオメトリシェーダ命令が実行された頂点シェーダ命令に付加されるかどうかを判定することができる。

【０１５１】

[0170]ＧＰＵ　３６がジオメトリシェーディング操作を実行しない場合（ステップ２１４のいいえの分岐）、ＧＰＵのハードウェアシェーディングユニットは、各々の入力された頂点に対する１つのシェーディングされた頂点を出力することができる（２２２）。ＧＰＵ　３６がジオメトリシェーディング操作を実行する場合（ステップ２１４のはいの分岐）、ハードウェアシェーディングユニットは、ジオメトリシェーディング操作の複数のインスタンスを実行して、受け取られた頂点に基づいて１つまたは複数の新たな頂点を生成することができる（２１６）。たとえば、ハードウェアシェーディングユニットは、所

定の数のジオメトリシェーディングのインスタンスを実行することができ、各インスタンスは出力識別子と関連付けられる。ハードウェアシェーディングユニットは、ジオメトリシェーディング操作の各インスタンスに対する出力カウントを保持することができる。加えて、出力識別子は、各々の出力された頂点に割り当てられ得る。

【０１５２】

[0171]したがって、ジオメトリシェーディングされた頂点をいつ出力するかを決定するために、ハードウェアシェーディングユニットは、出力カウントが出力識別子と一致するときを決定することができる（２１８）。たとえば、ジオメトリシェーディング操作に対する出力カウントが出力識別子と一致しない場合（ステップ２１８のいいえの分岐）、そのジオメトリシェーディング操作と関連付けられる頂点は廃棄される。ジオメトリシェーディング操作に対する出力カウントが出力識別子と一致する場合（ステップ２１８のはいの分岐）、ハードウェアシェーディングユニットは、ジオメトリシェーディング操作と関連付けられる頂点を出力することができる。このようにして、頂点シェーディングのために指定されるハードウェアシェーディングユニットは、単一のシェーディングされた頂点を出力し、ジオメトリシェーディングプログラムの各インスタンスに対する任意の使用されない頂点を廃棄し、これによって、１：１という入力対出力の比を維持する。

【０１５３】

[0172]図８は、テッセレーションステージを含む例示的なグラフィックスプロセシングパイプライン２３８を示すブロック図である。たとえば、パイプライン２３８は、入力アセンブラステージ２４０と、頂点シェーダステージ２４２と、ハルシェーダステージ２４４と、テッセレータステージ２４６と、ドメインシェーダステージ２４８と、ジオメトリシェーダステージ２５０と、ラスタライザステージ２５２と、ピクセルシェーダステージ２５４と、出力マージャステージ２５６とを含む。いくつかの例では、ＤｉｒｅｃｔＸ１１　ＡＰＩのようなＡＰＩは、図８に示されるステージの各々を使用するように構成され得る。グラフィックスプロセシングパイプライン２３８は、ＧＰＵ　３６によって実行されるものとして以下で説明されるが、種々の他のグラフィックスプロセッサによって実行され得る。

【０１５４】

[0173]図８に示されるいくつかのステージは、図２に関して示され説明されたステージ（たとえば、アセンブラステージ２４０、頂点シェーダステージ２４２、ジオメトリシェーダステージ２５０、ラスタライザステージ２５２、ピクセルシェーダステージ２５４、および出力マージャステージ２５６）と同様に、またはそれと同じように構成され得る。加えて、パイプライン２３８は、ハードウェアテッセレーションのための追加のステージを含む。たとえば、グラフィックスプロセシングパイプライン２３８は、図２に関して上で説明されたステージに加えて、ハルシェーダステージ２４４と、テッセレータステージ２４６と、ドメインシェーダステージ２４８とを含む。すなわち、ハルシェーダステージ２４４、テッセレータステージ２４６、およびドメインシェーダステージ２４８が、たとえば、ＣＰＵ　３２によって実行されているソフトウェアアプリケーションによる実行ではなく、ＧＰＵ　３６によるテッセレーションに対応するために含まれる。

【０１５５】

[0174]ハルシェーダステージ２４４は、頂点シェーダステージ２４２からプリミティブを受け取り、少なくとも２つの動作を実行することを担う。まず、ハルシェーダステージ２４４は通常、テッセレーション係数のセットを決定することを担う。ハルシェーダステージ２４４は、プリミティブごとに一度、テッセレーション係数を生成することができる。テッセレーション係数は、所与のプリミティブのテッセレーションをどの程度精密に行うか（たとえば、プリミティブをどのようにより小さな部分に分割するか）を判定するために、テッセレータステージ２４６によって使用され得る。ハルシェーダステージ２４４はまた、ドメインシェーダステージ２４８によって後で使用される制御ポイントを生成することを担う。すなわち、たとえば、ハルシェーダステージ２４４は、ドメインシェーダステージ２４８によって使用される制御ポイントを生成し、レンダリングにおいて最終的

に使用される、実際のテッセレーションが行われた頂点を作成することを担う。

【０１５６】

[0175]テッセレータステージ２４６がハルシェーダステージ２４４からのデータを受け取ると、テッセレータステージ２４６は、いくつかのアルゴリズムの１つを使用して、現在のプリミティブのタイプに対する適切なサンプリングパターンを決定する。たとえば、一般に、テッセレータステージ２４６は、現在の「ドメイン」内の座標点のグループへと、要求された量のテッセレーション（ハルシェーダステージ２４４によって決定されるような）を変換する。すなわち、ハルシェーダステージ２４４からのテッセレーション係数、さらにはテッセレータステージ２４６の具体的な構成に応じて、テッセレータステージ２４６は、より小さな部分への入力プリミティブのテッセレーションを行うために、現在のプリミティブのどの点がサンプリングされる必要があるかを決定する。テッセレータステージの出力は、重心座標を含み得るドメイン点のセットであり得る。

【０１５７】

[0176]ドメインシェーダステージ２４８は、ハルシェーダステージ２４４によって生成される制御ポイントに加えてドメイン点を取り込み、ドメイン点を使用して新たな頂点を作成する。ドメインシェーダステージ２４８は、現在のプリミティブ、テクスチャ、手順的アルゴリズム、または他の何かに対して生成される制御ポイントの完全なリストを使用して、各々のテッセレーションが行われた点に対する重心「位置」を、パイプライン中の次のステージに渡される出力ジオメトリへと変換することができる。上で述べられたように、一部のＧＰＵは、図８に示されたシェーダステージのすべてをサポートすることが不可能であり得る。たとえば、一部のＧＰＵは、ハードウェアおよび／またはソフトウェアの制約（たとえば、限られた数のシェーディングユニット４０および関連するコンポーネント）により、３つ以上のシェーディング操作を実行するようにシェーディングユニットを指定することが不可能であり得る。ある例では、いくつかのＧＰＵは、ジオメトリシェーダステージ２５０、ハルシェーダステージ２４４、およびドメインシェーダステージ２４８と関連付けられる動作をサポートしないことがある。むしろ、ＧＰＵは、頂点シェーダステージ２４２とピクセルシェーダステージ２５２とを実行するようにシェーディングユニットに指定することに対するサポートのみを含み得る。したがって、シェーディングユニットによって実行される操作は、頂点シェーダステージ８４およびピクセルシェーダステージ９０と関連付けられる入力／出力インターフェースを堅持しなければならない。

【０１５８】

[0177]加えて、比較的長いグラフィックスプロセシングパイプラインをサポートすることは、比較的複雑なハードウェアの構成を必要とし得る。たとえば、ハルシェーダステージ２４４、テッセレータステージ２４６、およびドメインシェーダステージ２４８からの、制御ポイント、ドメイン点、およびテッセレーション係数は、オフチップメモリに対する読取りと書込みとを必要とすることがあり、このことは、メモリのバス帯域幅を消費し、消費される電力の量を増やし得る。この意味で、各シェーダステージに対して専用のシェーディングユニット４０を使用する、多くのステージを伴うグラフィックスパイプラインを実装することは、より電力効率の低いＧＰＵをもたらし得る。加えて、そのようなＧＰＵはまた、メモリのバス帯域幅が限られている結果としての、オフチップメモリからのデータの取り出しの遅延により、レンダリングされる画像の出力がより遅くなり得る。

【０１５９】

[0178]本開示の態様によれば、以下でより詳細に説明されるように、特定のシェーディング操作を実行するようにＧＰＵ　３６によって指定されるシェーディングユニット４０は、２つ以上の操作を実行することができる。たとえば、頂点シェーディング（ＶＳ）操作を実行するように指定されるシェーディングユニット４０はまた、ハルシェーダステージ２４４と関連付けられるハルシェーディング操作を実行することができる。別の例では、同じシェーディングユニット４０はまた、ドメインシェーダステージ２４８と関連付けられるドメインシェーディング操作を実行し、続いて、ジオメトリシェーダステージ２５０と関連付けられるジオメトリシェーダ操作を実行することができる。

【０１６０】
[0179]以下でより詳細に説明されるように、ＧＰＵ　３６は、ドローコールを２つのサブドローコール（たとえば、パスＩおよびパスＩＩ）へと分割することによって上のシェーディング操作を実行することができ、各サブドローコールは関連するマージされたシェーダステージを有する。すなわち、ＧＰＵ　３６は、頂点シェーディング操作を実行するためにシェーディングユニット４０を呼び出すことができるが、第１のパスの間にハルシェーディング操作を実行するように、シェーディングユニット４０を実装することもできる。ＧＰＵ　３６は次いで、同じシェーディングユニット４０（頂点シェーディング操作を実行するように指定される）を使用して、ハルシェーディング、ドメインシェーディング、またはジオメトリシェーディングのタスクを実行するようにシェーディングユニット４０を決して再指定することなく、ドメインシェーディング操作とジオメトリシェーディング操作とを実行することができる。
【０１６１】
[0180]図９は、より詳細にテッセレーションを示す概念図である。ハルシェーダ（ＨＳ）ステージ２４４およびドメインシェーダ（ＤＳ）２４８は、完全なシェーダステージであってよく、このステージの各々は、定数バッファ、テクスチャ、および他のリソースの固有のセットを伴う。一般に、テッセレーションは、パッチと呼ばれるプリミティブのタイプを使用して実行され得る。したがって、図９に示される例では、ハルシェーダステージ２４４は最初に、１つまたは複数の入力される制御ポイントを受け取り、これはパッチ制御ポイントと呼ばれ得る。パッチ制御ポイントは、（たとえば、ＡＰＩを使用して）開発者により制御されるものであり得る。ハルシェーダステージ２４４は、以下で説明されるように、計算を実行して、ドメインシェーダステージ２４８によって使用される制御ポイントを含むいわゆるベジェパッチを生成することができる。
【０１６２】
[0181]ハルシェーダステージ２４４はまた、パッチのテッセレーションの量を制御するために使用され得る、テッセレーション係数を生成する。たとえば、ハルシェーダステージ２４４は、パッチの視点および／または視距離に基づいて、どの程度テッセレーションを行うかを決定し得る。オブジェクトがあるシーンにおいて見る者に比較的近い場合、比較的多量のテッセレーションが、一般に滑らかに見えるパッチを生成するために必要とされ得る。オブジェクトが比較的遠い場合、より少量のテッセレーションが必要とされ得る。
【０１６３】
[0182]テッセレータステージ２４６は、テッセレーション係数を受け取り、テッセレーションを実行する。たとえば、テッセレータステージ２４６は、多数の｛Ｕ，Ｖ｝座標を生成するための均一の等級を有する所与のパッチ（たとえば、ベジェパッチ）に対して行われる。｛Ｕ，Ｖ｝座標は、パッチに対するテクスチャを提供することができる。したがって、ドメインシェーダステージ２４８は、制御ポイント（変位情報を有する）と、｛Ｕ，Ｖ｝座標（テクスチャ情報を有する）と、出力されるテッセレーションが行われた頂点とを受け取ることができる。上で説明されたように、これらのテッセレーションが行われた頂点は次いで、ジオメトリシェーディングされ得る。
【０１６４】
[0183]本開示の態様によれば、かつ以下でより詳細に説明されるように、ハルシェーダステージ２４４およびドメインシェーダステージ２４８と関連付けられるシェーディング操作は、ＧＰＵの同じシェーディングユニット（シェーディングユニット４０のような）によって実行され得る。すなわち、たとえば、１つまたは複数のシェーディングユニット４０は、頂点シェーディング操作を実行するように指定され得る。頂点シェーディング操作に加えて、ＧＰＵは、シェーダが、順番に、かつテッセレーション操作を実行するように再構成されることなく、同じシェーディングユニットによって実行されるように、ハルシェーダステージ２４４およびドメインシェーダステージ２４８と関連付けられるシェーダ命令を付加することができる。

【０１６５】
[0184]図１０Ａおよび図１０Ｂは、本開示の態様による、グラフィックスレンダリングパイプラインにおけるデータフローの概念図である。たとえば、図１０Ａは、頂点シェーダステージ２６０と、ハルシェーダステージ２６２と、テッセレータステージ２６４と、ドメインシェーダステージ２６６と、ジオメトリシェーダステージ２６８と、ストリームアウト２７０と、ピクセルシェーダステージ２７２とを示す。一般に、図１０Ａに示されるシェーダステージの各々は、シェーディング操作を実行するための関連するハードウェアを表す。すなわち、たとえば、頂点シェーダステージ２６０、ハルシェーダステージ２６２、ドメインシェーダステージ２６６、ジオメトリシェーダステージ２６８、およびピクセルシェーダステージ２７２の各々は、シェーディングユニット４０のような、別々に指定されたプロセシングユニットと関連付けられ得る。
【０１６６】
[0185]図１０Ａに示される例では、頂点シェーダステージ２６０は、いわゆる「パッチ制御ポイント」（または、図８および図９に関して上で説明されるような「制御ポイント」）で呼び出され得る。所与のパッチの中の点は、ハルシェーダステージ２６２に可視であってよく、ハルシェーダステージ２６２は、それらの点を使用して、テッセレーションステージ２６４による使用のためのテッセレーション係数を計算する。ハルシェーダステージ２６２はまた、ドメインシェーダステージ２６６による使用のための、パッチ制御ポイントと定数データとを出力することができる。
【０１６７】
[0186]いくつかの例では、テッセレータステージ２６４は、テッセレーションを実行するための固定された機能のハードウェアユニットを含み得る。テッセレータステージ２６４は、ハルシェーダステージ２６２からテッセレーション係数と制御ポイントとを受け取り、いわゆるドメイン点（たとえば、どこでテッセレーションを行うかを規定する｛Ｕ，Ｖ｝点）を出力することができる。ドメインシェーダステージ２６６は、これらのドメイン点を使用して、ハルシェーダステージ２６２からの出力されるパッチデータを使用して頂点を計算する。ドメインシェーダステージ２６６からのあり得る出力プリミティブは、ラスタライズのために、ストリームアウト２７０のために、またはジオメトリシェーダステージ２６８へと送信され得る、たとえば、点、線、または三角形を含む。テッセレーション係数のいずれかが０以下である場合、または数字ではない（ＮａＮ）場合、パッチは間引かれ得る（さらに計算されることなく廃棄され得る）。
【０１６８】
[0187]図１０Ａに示されるシェーダステージは、１つまたは複数のグラフィックスＡＰＩをサポートすることができる。説明のための例では、頂点シェーダステージ２６０、ハルシェーダステージ２６２、ドメインシェーダステージ２６６、ジオメトリシェーダステージ２６８、およびピクセルシェーダステージ２７２は、ＤｉｒｅｃｔＸ　１１　ＡＰＩをサポートすることができる。すなわち、ＤｉｒｅｃｔＸ　１１　ＡＰＩを使用して作成されたコードが、グラフィックスデータをレンダリングするために、頂点シェーダステージ２６０、ハルシェーダステージ２６２、ドメインシェーダステージ２６６、ジオメトリシェーダステージ２６８、およびピクセルシェーダステージ２７２によって実行され得る。しかしながら、ハルシェーダステージ２６２、ドメインシェーダステージ２６６、および／またはジオメトリシェーダステージ２６８のようないくつかのステージは、すべてのグラフィックスレンダリングパイプラインに含まれなくてよく、すべてのＧＰＵによって実行されなくてよい。たとえば、ＤｉｒｅｃｔＸ　１１　ＡＰＩはそのようなステージに対するサポートを含むが、いくつかのより以前の改訂（たとえば、ＤｉｒｅｃｔＸ　９および１０）はそのようなサポートを含まない。したがって、ＤｉｒｅｃｔＸ　ＡＰＩのより以前の改訂によって作成されたコードを実行するように設計されるＧＰＵ（または他のＡＰＩのために設計されたＧＰＵ）は、ハルシェーダステージ２６２、ドメインシェーダステージ２６６、および／またはジオメトリシェーダステージ２６８と関連付けられる操作を実行するようにシェーディングユニット４０を指定することが不可能であり得る。

【０１６９】
[0188]本開示の態様によれば、図１０Ａのシェーダステージの２つ以上は、シェーダステージが単一のハードウェアシェーディングユニット（たとえば、シェーディングユニット４０のような）によって実行されるという点でマージされ得る。たとえば、本開示の態様によれば、ＧＰＵ（ＧＰＵ　３６のような）は、図１０Ｂに関して以下で説明されるように、ドローコールを実行して図１０Ａに示されるシェーダステージを行うとき、複数のパスを実行することができる。
【０１７０】
[0189]図１０Ｂは、マージされた頂点シェーダおよびハルシェーダ（ＶＳ／ＨＳ）ステージ２８０を有する第１のパス（パスＩ）を含む、グラフィックスレンダリングパイプラインにおけるデータフローを示す。加えて、データフローは、テッセレーションステージ２８２と、マージされたドメインシェーダおよびジオメトリシェーダ（ＤＳ／ＧＳ）ステージ２８４と、ストリームアウト２８６と、ピクセルシェーダステージ２８８とを有する、第２のパス（パスＩＩ）を含む。図１０Ｂに示されるパスは、テッセレーション操作を有するドローコールを実行するように実施され得る。
【０１７１】
[0190]たとえば、図１０Ａに関して上で説明されたように、ＧＰＵ　３６は、テッセレーション操作を含む入力ドローコールを実行することができる。ＧＰＵ　３６は最初に、複数のサブドローコールへとドローコールを分割することができ、各サブドローコールは、パスＩ操作とパスＩＩ操作の両方を含む。ＧＰＵ　３６がドローコールを分割する方式は、利用可能なメモリ（たとえば、オンチップＧＰＵメモリ、Ｌ２、グローバルメモリ（ＧＭＥＭ）、またはオフチップメモリ）の量に少なくとも部分的に依存し得る。たとえば、ＧＰＵ　３６は、ＧＰＵ　３６がパスＩ操作によって生成されたデータのすべてをパスＩＩ操作において使用するためにローカルメモリに記憶することが可能であるように、サブドローコールを構成することができる。ドローコールの分割は、コマンドプロセッサ（ＣＰ）コードの制御のもとで、ＣＰにおいて行われてよく、ＣＰコードは、入力ドローコールのタイプに基づき得る。
【０１７２】
[0191]説明のための例では、ドローコールがレンダリングのための１０００個の関連するパッチを含むことを仮定する。加えて、ローカルメモリは１００個のパッチと関連付けられるデータを記憶するための容量を有すると仮定する。この例では、ＧＰＵ　３６（または、ＧＰＵドライバ５０のようなＧＰＵのためのドライバ）は、ドローコールを１０個のサブドローコールへと分割することができる。ＧＰＵ　３６は次いで、１０個のサブドローコールの各々に対するパスＩ操作とパスＩＩ操作とを順番に実行する。
【０１７３】
[0192]パスＩ操作に関して、頂点シェーディング操作がＧＰＵ　３６によって呼び出されると、ＶＳ／ＨＳステージ２８０は、頂点シェーディング操作とハルシェーディング操作の両方を実行することができる。すなわち、マージされたＶＳ／ＨＳステージ２８０は、１つまたは複数のシェーディングユニットの単一のセットを含んでよく、頂点シェーダステージ２６０およびハルシェーダステージ２６２に関して上で説明された操作を順番に実行してよい。以下でより詳細に説明されるように、本開示の態様は、ＧＰＵ　３６が、頂点シェーディング操作と同じシェーディングユニットによってハルシェーディング操作を実行しつつ、適切なインターフェースを依然として堅持することを可能にする。いくつかの例では、ハルシェーダ命令は、パッチコードを使用して頂点シェーダ命令に付加されてよく、これによって、同じシェーディングユニットが命令の両方のセットを実行することを可能にする。
【０１７４】
[0193]ＧＰＵ　３６は次いで、パスＩＩ操作を実行する。たとえば、テッセレーションステージ２８２は、上のテッセレーションステージ２６４に関して説明されたように、テッセレーションを実行することができる。マージされたＤＳ／ＧＳステージ２８４は、上

で説明されたマージされたＶＳ／ＨＳステージ２８０と同じ、１つまたは複数のシェーディングユニット４０のセットを含み得る。マージされたＤＳ／ＧＳステージ２８４は、ドメインシェーダステージ２６６およびジオメトリシェーダステージ３６８に関して上で説明されたドメインシェーディング操作とジオメトリシェーディング操作とを順番に実行することができる。いくつかの例では、ジオメトリシェーダ命令は、パッチコードを使用してドメインシェーダ命令に付加されてよく、これによって、同じシェーディングユニットが命令の両方のセットを実行することを可能にする。その上、これらのドメインシェーダ命令およびジオメトリシェーダ命令は、（パスＩの）ハルシェーダ命令に付加され得るので、同じシェーディングユニットが、頂点シェーディングと、ハルシェーディングと、ドメインシェーディングと、ジオメトリシェーディングとを、再構成されることなく実行することができる。

【０１７５】

[0194]パスＩＩのジオメトリシェーディング操作は、上で説明されたものと同じジオメトリシェーディング操作を基本的に含み得る。しかしながら、パスＩＩ操作を開始するとき、ＧＰＲにより初期化された入力（前はＶＳステージのための、今はＤＳステージのための）は、頂点フェッチデコーダ（ＶＦＤ）からフェッチされたデータではなく、テッセレーションステージ２８２によって生成される（u，v，patch_id）を含み得る。ＰＣは、パスＩＩに対するrel_patch_idも計算することができ、テッセレーションステージ２８２によって計算される（u，v）とともに、パッチＩＤ情報をＤＳに渡すことができる。テッセレーションステージ２８２は、テッセレーション係数を使用して、テッセレーションが行われた頂点の（u，v）座標を生成することができる。テッセレーションステージ２８２の出力は、さらなる増幅（ジオメトリシェーディング）またはストリームアウト２８６のためにテッセレーションが行われることを準備するために、マージされたＤＳ／ＧＳステージ２８４に与えられ得る。ＤＳは、オフチップスクラッチメモリからの、ハルシェーダ（ＨＳ）の出力制御ポイントデータとＨＳパッチ定数データとを使用する。

【０１７６】

[0195]いくつかの例では、図１０Ｂに示される２つのパスは、連続的に実行され得るが、２つのパスの間のアイドル状態の待機によって分離され得る。たとえば、ＧＰＵのＣＰは、パスＩ操作のためのドローコールを送ることができる。データに対してパスＩＩを開始する前に、ＧＰＵは、制御ポイントの値がローカルメモリに完全に書き込まれるのを待機することができる。現在値がローカルメモリにおいて利用可能であることを確実にするために、ＧＰＵは、ＧＰＵのコンポーネントがパスＩＩ操作を開始する前はアイドル状態であることを確認することができる。

【０１７７】

[0196]コマンドプロセッサ（ＣＰ）は次いで、パスＩＩのためのドローコールを送ることができる。ある例では、パスＩＩにおいて行われた作業の量に対する、第１の有用な頂点を開始するための遅延の量の比は、およそ２％未満であり得る。したがって、いくつかの例では、パスＩとパスＩＩとの間には重複がないことがある。他の例では、以下で説明されるように、ＧＰＵは、パスＩ操作とパスＩＩ操作の間に重複を含み得る。すなわち、ＧＰＵは、以前のドローコールのパスＩＩのピクセルシェーダステージ２８８のピクセルシェーディング操作を、現在のドローコールのパスＩのＶＳ／ＨＳステージ２８０の頂点シェーディング操作と重複させることができ、それは、ピクセルシェーダの処理が頂点シェーダの処理よりも長くかかり得るからである。

【０１７８】

[0197]本開示の態様によれば、プリミティブコントローラ（ＰＣ）は、パスＩの後にPASS_doneイベントを送ることができ、これは、ハードウェアユニットがパスＩＩに切り替えることを助け得る。パスＩとパスＩＩとの間に重複があり得る例では、パスＩ操作とパスＩＩ操作の存在は、命令を実行するシェーダプロセッサにおいて相互に排他的であり得る。しかしながら、パスＩＩに対するテッセレーション係数は、パスＩがまだ実行されている間にフェッチされ得る。

【０１７９】
[0198]図１１に関して以下で説明されるように、ＰＣは、どれだけのパスＩのウェーブが完了したかを記録するために、シェーディングされたパッチごとにカウンタを保持することができる。これらのカウンタは、どれだけのパッチがパスＩの処理を完了したかを示すことができる。すべてのカウンタ値が０より大きくなるとすぐに、テッセレーション係数が、パスＩＩのためにフェッチされ得る。したがって、パスＩＩは、パスＩが完了する前に開始し得る。しかしながら、パスＩのドローコールに対するインデックスのすべてが処理されるまで、パスＩＩに対するドローコールは開始しなくてよい。このようにして、複数のパスの間でのパイプラインのフラッシュ（ローカルＧＰＵメモリから外部メモリへの移行）が回避され得る。

【０１８０】
[0199]図１１は、本開示で説明される技法を実施して頂点シェーディング操作とハルシェーディング操作とを実行する、ハードウェアシェーディングユニットの例示的な動作を示す図である。たとえば、図１１は一般に、本開示の技法に従って、図１０Ｂに関して上で説明されたように、ドローコールの第１のパス（パスＩ）の間に頂点シェーディング操作とハルシェーディング操作とを実行することを示す。ＧＰＵ　３６（図１）に関して説明されるが、本開示の態様は、種々の他のコンポーネントを有する多種多様な他のＧＰＵによって実行され得る。

【０１８１】
[0200]図１１の例では、ＧＰＵ　３６は、頂点シェーディング操作を実行するようにシェーディングユニット４０を指定することができ、このシェーディングユニット４０はまた、最終的に、以下でより詳細に説明されるように、ハルシェーディングと、ドメインシェーディングと、ジオメトリシェーディングとを、そのようなシェーディング操作を実行するように再構成されることなく、実行することができる。たとえば、シェーディングユニット４０は最初に、頂点シェーディング操作を実行して、点ｐ０～ｐ２として図示される３つの頂点を有する入力プリミティブ（トライアングルストリップ）を生成することができる。

【０１８２】
[0201]頂点シェーディング操作を実行した後で、ＧＰＵ　３６は、シェーディングされた頂点をローカルのメモリリソースに記憶することができる。たとえば、ＧＰＵ　３６は、（たとえば、ＧＰＵメモリ３８の）位置キャッシュに頂点シェーダ出力をエクスポートすることができる。頂点シェーディング操作およびハルシェーディング操作は、ＶＳ　ＥＮＤ命令によって分離され得る。したがって、ＶＳ　ＥＮＤ命令を実行し頂点シェーディング操作を完了した後、頂点シェーディング操作を実行するように指定された１つまたは複数のシェーディングユニット４０は、各々、ハルシェーディング操作の実行を開始する。

【０１８３】
[0202]同じシェーディングユニット４０は次いで、ハルシェーディング操作を実行して、制御ポイントＶ０～Ｖ３を有する出力パッチを生成することができる。この例では、シェーディングユニット４０は、ハルシェーダ操作の複数のインスタンスを実行する（これは、図４に関して上で説明されたジオメトリシェーダ操作と同様の方式で、出力識別子（Ｏｕｔｖｅｒｔ）によって図示されている）。ハルシェーダ操作の各インスタンスは、同じアルゴリズムを実行して、同じハルシェーディング操作を実行し、１つまたは複数の新たな制御ポイントＶ０～Ｖ３のそれぞれのインスタンスを生成する。

【０１８４】
[0203]すなわち、図１１に示される表の４個の列は、ハルシェーダ操作（またはプログラム）の４個の別個のインスタンスに対応し、各列は左から右へ、０～３のハルシェーダ操作Ｏｕｔｖｅｒｔによって識別され得る。ハルシェーダ操作のこれらの４個のインスタンスの各々は、シェーディングユニット４０によって、しばしば同時に実行されて、１つまたは複数の新たな制御ポイントの別個のインスタンスを生成する。したがって、ハルシ

ェーダ操作のインスタンスの各々は、４個すべての頂点（Ｖ０～Ｖ３）を生成するが、４個の新たな制御ポイントの対応する１つのみを出力する。ハルシェーダ操作の各インスタンスは、頂点シェーディング操作のために呼び出された、シェーディングユニット４０の１：１のインターフェースを堅持するために、４個の新たな制御ポイントの対応する１つのみを出力する。

【０１８５】

[0204]図１１の例では、ハルシェーダ操作の各々は、そのOutvertと一致する４個の新たな頂点の１つを出力する。したがって、Outvert = 0を有するハルシェーダ操作の第１のインスタンスは、４個の新たな頂点のうちの第１の頂点、Ｖ０を出力する。したがって、Outvert = 1を有するハルシェーダ操作の第２のインスタンスは、４個の新たな頂点のうちの第２の頂点、Ｖ１を出力する。したがって、Outvert = 2を有するハルシェーダ操作の第３のインスタンスは、４個の新たな頂点のうちの第３の頂点、Ｖ２を出力する。したがって、Outvert = 3を有するハルシェーダ操作の第４のインスタンスは、４個の新たな頂点のうちの第４の頂点、Ｖ３を出力する。ハルシェーダ値がローカルメモリに書き込まれた後、上で説明されたように、ドメインシェーディング操作およびジオメトリシェーディング操作が、第２のパス（パスＩＩ）の間に実行され得る。

【０１８６】

[0205]本開示の態様によれば、頂点シェーディング操作を実行するように指定された同じシェーディングユニット４０はまた、上で説明されたハルシェーディング操作を実行する。その上、同じシェーディングユニット４０がまた、ドローコールの第２のパス（パスＩＩ）の間に、ドメインシェーディング操作とジオメトリシェーディング操作とを実行することができる。たとえば、ＧＰＵ　３６は、状態をシェーダ固有のリソース（たとえば、ハルシェーダ、ドメインシェーダ、および／またはジオメトリシェーダの定数、テクスチャオフセットなど）へと変更することができる。ＧＰＵ　３６は、シェーディング操作に割り当てられたモード（ドローモード）に従って、この状態変更を実行することができる。

【０１８７】

[0206]以下に示される表４は、頂点シェーディングと、ハルシェーディングと、ドメインシェーディングと、ジオメトリシェーディングとを同じシェーディングユニット４０によって実行するための、ＧＰＵ　３６によって保持され得る動作モードとパラメータとを示す。

【表４】

表４：シェーディング操作を実行するためのモード

モード	モード０ GS:オフ, HS:オフ	モード１ GS:オン HS:オフ	モード４ GS:オン, HS:オン (パスII)	モード３ GS:オフ, HS:オン (パスII)	モード２ GS:オフ, HS:オン (パスI)
フロー	VS->PS	VS| GS->PS	DS| GS ->PS	DS->PS	VS|HS
インデックス (32ビット)	*頂点インデックス(VS)*	*頂点インデックス(VS)*	u(15:0) v (31:16)	u(15:0) v (31:16)	頂点 インデックス
uv_msb (2ビット)	使用 されない	使用 されない	u、vの上位 ビット	u、vの上位 ビット	*使用 されない*
PrimitiveID (32ビット)	使用 されない	*PrimitiveID (GS)*	*PrimtiiveID (DS, GS)*	*PrimtiiveID (DS)*	*PrimtiiveID (HS)*
Rel_patchid (32ビット)	使用 されない	使用 されない	Rel_patchid (DS)	Rel_patchid (DS)	Rel_patchid (HS)
Misc (25ビット)	使用 されない	misc->rel_primID (4:0)	misc->rel_primID (4:0)	使用 されない	misc->rel_primID (4:0)
		misc->rel_vertex (9:5)	misc->rel_vertex (9:5)		misc->rel_vertex (9:5)
		misc->GsInstance (14:10)	misc->GsInstance (14:10)		misc->outvertID (14:10)
		misc->Gsoutvertex (24:15)	misc->Gsoutvertex (24:15)		
Vs_valid (1ビット)					
Gshs_valid (1ビット)					
モード (2:0)	モード= モード_0	モード= モード_1	モード = モード_4	モード= モード_3	モード= モード_2
Instance_cmd (2ビット)					

【０１８８】

[0207]いくつかの例では、上の表４に示されるように、いくつかのシェーディング操作は、特定のドローコールに対しては実行されなくてよい。たとえば、ドローコールは、頂点シェーディング操作と、ハルシェーディング操作と、ドメインシェーディング操作と、ピクセルシェーディング操作とを含み得るが、（モード３について示されるように）ジオメトリシェーディング操作を含まないことがある。ＧＰＵ　３６は、モード情報を使用し

て、ドローコールを実行するときにどのシェーディング操作を行うかを決定することができる。

【０１８９】

[0208]以下で示される表５は、ジオメトリシェーディング操作を実行することなくパスＩＩ操作を実行するときのパラメータ値を示す。

【表５】

表5:ジオメトリシェーディングを伴わないパラメータ値

モード3 GS:オフ、 HS:オン	ファイバー0	ファイバー1	ファイバー2	ファイバー3	ファイバー4	ファイバー5	ファイバー6	ファイバー7
Valid_as_input	1	1	1	1	1	1	1	1
頂点インデックス(VS)	U|V	U|V	U|V	U|V	U|V	U|V	U|V	U|V
Uv_msb	u|v	u|v	u|v	u|v	u|v	u|v	u|v	u|v
primitiveID (HS)	105	105	105	105	105	105	105	105
Rel_patchID	5	5	5	5	5	5	5	5

【０１９０】

[0209]以下で示される表６は、ジオメトリシェーディング操作を実行することを含むパスＩＩ操作を実行するときのパラメータ値を示す。

【表６】

表6:ジオメトリシェーディングを伴うパラメータ値

モード4 GS:オフ、 HS:オン	ファイバー0	ファイバー1	ファイバー2	ファイバー3	ファイバー4	ファイバー5	ファイバー6	ファイバー7
Valid_as_input	1	1	1	0	0	0	0	0
頂点インデックス(VS)	U|V	U|V	U|V	U|V	0	0	0	0
Uv_msb	u|v	u|v	u|v	u|v	0	0	0	0
primitiveID (HS およびGS)	105	105	105	105	105	105	105	105
Rel_patchID	5	5	5	5	5	5	5	5
Valid_as_output	1	1	1	1	1	1	1	1
misc-> rel_primID (4:0)	0	0	0	0	0	0	0	0
misc-> rel_vertex (9:5)	0	1	2	0	0	0	0	0
misc-> GSInstance1 4:10)	0	0	2	0	0	0	0	0
misc-> GsOutvertex (24:15)	0	1	2	3	4	5	6	7

【０１９１】

[0210]図１１に示されるような第１のパス（パスＩ）と関連付けられる操作を完了した後、ＧＰＵ　３６はアイドル状態を待機することができる。ＧＰＵ　３６は次いで、ドローコールの第２のパス（パスＩＩ）を実行して、ドローコールを完了することができる。

【０１９２】

[0211]図１２Ａおよび図１２Ｂは、本開示の技法を実施するハードウェアシェーディングユニットによって実行され得る例示的な動作を示す。図１２Ａおよび図１２Ｂは一般に、パスＩに関して上で説明されたシェーディング操作に対応し得る。

【０１９３】

[0212]たとえば、図１２Ａは一般に、頂点シェーディング操作とハルシェーディング操作とを実行するときにマージされたＶＳ／ＨＳハードウェアシェーディングユニットによって実行される動作のフローを示す。マージされたＶＳ／ＨＳハードウェアシェーディングユニットは、いくつかの例では、頂点シェーディング操作を実行するようにＧＰＵ　３６によって指定されるが本開示の技法に従って頂点シェーディング操作とハルシェーディ

ング操作の両方を実行する、シェーディングユニット４０を含み得る。図１２Ｂは、マージされたＶＳ／ＨＳハードウェアシェーディングユニットによって実行され得る、図１２Ａに示される動作のフローに対応する擬似コードを一般に示す。

【０１９４】

[0213]図１２Ａに示されるように、ハードウェアシェーディングユニットは、ＶＳ操作を実行し、続いてＨＳ操作を実行することができる。たとえば、ＧＰＵ（ＧＰＵ　３６のような）は、頂点の属性、vertex_id、instance_id、primitive_id、およびmisc（上で説明されたような）を含む、システムにより生成された値をレジスタに書き込むことができる。上で述べられたように、システムにより生成された値を所定の位置にある一連のレジスタに記憶することによって、ＧＰＵ　３６は、ＶＳステージおよびＨＳステージの各々に対する、システムにより生成された値にアクセスすることができる。したがって、ＨＳステージは、システムにより生成された値がどこに記憶されたかを判定するために、ＶＳステージに基づいてコンパイルされる必要がない。むしろ、ＧＰＵ　３６は、ステージの各々を実行するときに所定のメモリ位置にアクセスして、システムにより生成された必要とされる値にアクセスすることができる。

【０１９５】

[0214]ハードウェアシェーディングユニットは次いで、頂点シェーディング操作を実行して、１つまたは複数のシェーディングされた頂点を生成することができる。ハードウェアシェーディングユニットは、シェーディングされた頂点がハルシェーディング操作のために利用可能であるように、シェーディングされた頂点をローカルメモリに書き込むことができる。

【０１９６】

[0215]ＧＰＵは次いで、ハルシェーディング操作を実行する前に、メモリオフセットとプログラムカウンタとを切り替えることができる。ＧＰＵは、たとえば、上で説明されたパッチコードを実行するときに、そのようなタスクを実行することができる。ハードウェアシェーディングユニットは次いで、ローカルメモリからシェーディングされた頂点を読み取り、ハルシェーディング操作を実行して、１つまたは複数の制御ポイントとテッセレーション係数とを生成することができる。

【０１９７】

[0216]第１のパスの間に生成される制御ポイントおよびテッセレーション係数は、たとえば、ローカルＧＰＵメモリに記憶され得る。いくつかの例では、制御ポイントおよびテッセレーション係数は、ローカルＧＰＵメモリ内の別個のバッファに記憶され得る。

【０１９８】

[0217]図１２Ｂは、上で説明されたパスＩ操作を実行するハードウェアシェーディングユニットによって実行され得るコードの例示的な一部分である。図１２Ｂに示される例では、大文字の用語は、状態または定数レジスタである。斜字の用語は、シェーダ入力を示す。ＶＳ／ＨＳ操作に割り振られるＧＰＲの数は、（gprs_needed_for_vsとgprs_needed_for_hs）の大きい方である。したがって、ＶＳ操作で使用された後、ＧＰＲは解放され、ＨＳ操作のために使用される。

【０１９９】

[0218]いくつかの例では、シェーディング操作のＶＳ部分では、（図５Ｂに関して上で述べられたように）有効なＶＳファイバーのみが実行される。「SWITCH_ACTIVE」命令に遭遇すると、カバレッジマスクビットが、ＨＳシェーダと関連付けられるように変更され、アクティブなＨＳファイバーのみが実行される。このようにして、確保されたレジスタはＶＳとＨＳの両方のために使用されてよく、ＶＳおよびＨＳは、ＨＳ操作を実行するようにシェーディングユニットを再指定することなく、単一のハードウェアシェーディングユニットによって実施され得る。

【０２００】

[0219]図１３Ａおよび図１３Ｂは、本開示の技法を実施するハードウェアシェーディングユニットによって実行され得る例示的な動作を示す。図１３Ａおよび図１３Ｂは一般に

、上で説明されたパスＩＩシェーディング操作に対応し得る。

【０２０１】

[0220]たとえば、図１３Ａは一般に、ドメインシェーディング操作とジオメトリシェーディング操作とを実行するときにマージされたＤＳ／ＧＳハードウェアシェーディングユニットによって実行される動作のフローを示す。マージされたＤＳ／ＧＳハードウェアシェーディングユニットは、いくつかの例では、図１２Ａおよび図１２Ｂに関して上で説明されたものと同じ、かつ最初は頂点シェーディング操作を実行するようにＧＰＵ　３６によって指定された、シェーディングユニット４０を含み得る。図１３Ｂは、マージされたＤＳ／ＧＳハードウェアシェーディングユニットによって実行され得る、図１３Ａに示される動作のフローに対応する擬似コードを一般に示す。

【０２０２】

[0221]本開示の態様によれば、第１のパス（図１２Ａおよび図１２Ｂに関して説明された）の後に、「アイドル状態を待機する」が続き得る。すなわち、データが第１のパスの間にメモリへ完全に書き込まれる前に、第２のパスの間にデータがローカルメモリから読み取られるのを防ぐために、ＧＰＵは、図１３Ａおよび図１３Ｂに示される第２のパスの操作を開始する前はアイドル状態（たとえば、データを計算または転送していない）であるものとして登録するために、ＧＰＵの１つまたは複数のコンポーネントを待機することができる。

【０２０３】

[0222]いずれの場合でも、図１３Ａに示されるように、ハードウェアシェーディングユニットは、ドメインシェーディングとジオメトリシェーディングとを含むパスＩＩ操作を実行することができる（テッセレーションも、固定された機能のテッセレーションユニットによって実行され得る）。たとえば、ＧＰＵは、｛U, V｝座標と、primitive_idと、misc（上で説明されたような）とを含む、システムにより生成された値をレジスタに書き込むことができる。上で述べられたように、システムにより生成された値を所定の位置にある一連のレジスタに記憶することによって、ＧＰＵ　３６は、ＤＳステージおよびＧＳステージの各々に対する、システムにより生成された値にアクセスすることができる。したがって、ＧＳステージは、システムにより生成された値がどこに記憶されたかを判定するために、ＤＳステージに基づいてコンパイルされる必要がない。むしろ、ＧＰＵ　３６は、ステージの各々を実行するときに所定のメモリ位置にアクセスして、システムにより生成された必要とされる値にアクセスすることができる。

【０２０４】

[0223]ハードウェアシェーディングユニットは次いで、ドメインシェーディング操作を実行して、１つまたは複数のテッセレーションが行われた頂点を生成することができる。ハードウェアシェーディングユニットは、テッセレーションが行われた頂点がジオメトリシェーディング操作のために利用可能であるように、テッセレーションが行われた頂点をローカルメモリに書き込むことができる。

【０２０５】

[0224]ＧＰＵは次いで、ジオメトリシェーディング操作を実行する前に、メモリオフセットとプログラムカウンタとを切り替えることができる。ＧＰＵは、たとえば、上で説明されたパッチコードを実行するときに、そのようなタスクを実行することができる。ハードウェアシェーディングユニットは次いで、ローカルメモリからテッセレーションが行われた頂点を読み取り、ジオメトリシェーディング操作を実行して、頂点パラメータキャッシュに記憶され得る１つまたは複数のジオメトリシェーディングされた頂点を生成することができる。

【０２０６】

[0225]図１３Ｂに示される例では、大文字の用語は、状態または定数レジスタである。斜字の用語は、シェーダ入力を示す。このシェーダに割り振られるＧＰＲの数は、（gprs_needed_for_vsとgprs_needed_for_gs）の大きい方である。したがって、ＤＳ操作で使用されたＧＰＲは、解放されＧＳ操作のために使用される。「SWITCH_ACTIVE」命令に遭遇

すると、カバレッジマスクビットが、ＧＳ操作と関連付けられるように変更され、アクティブなＧＳファイバーのみが実行される。「END_1st」命令に遭遇すると、ハードウェアシェーダユニットは、定数ファイルおよびテクスチャポインタ（たとえば、リソースポインタ）に対するリソースオフセットを、ＧＳによりプログラムされたオフセットへと切り替え、ＧＳの第１の命令にジャンプすることができる。このようにして、確保されたレジスタは、ＤＳシェーダステージとＧＳシェーダステージの両方によって使用されてよく、ＤＳシェーダステージおよびＧＳシェーダステージは、パスＩ操作を実行した同じハードウェアシェーディングユニットによって実行され得る。

【０２０７】

[0226]図１２Ａ～図１３Ｂの例に示されるように、単一のハードウェアシェーディングユニットが、４個の異なるシェーダステージを実行することができる。いくつかの例によれば、シェーダステージをマージするためのパッチコードは、どのシェーダステージがマージされているかに関係なく、同じであり得る。たとえば、ＤＳ操作は、（図１２Ｂの一番上から２番目の破線のボックスにおいて示される）ＶＳ操作とＨＳ操作とをマージするために使用されたものと同じパッチコードを使用して、ＧＳ操作とマージされ得る（図１３Ｂの一番上から２番目の破線のボックスにおいて示される）。ハードウェアシェーディングユニットは、動作モード（上の表に関して示された説明されたような）に基づいて、適切なシェーディング操作へと切り替えることができ、これは、ドローのときにＧＰＵによって決定され得る。

【０２０８】

[0227]本開示の態様によれば、各シェーダステージ（ＶＳ／ＧＳ／ＨＳ／ＤＳ）は、別個に、かつ、実行の間にどのようにステージがリンクされるかを知ることなく、まとめられ得る。したがって、３個のＧＰＲが、primitveID、rel_patch_ID、およびmiscのようなパラメータを記憶するために確保され得る。コンパイラは、ＤＸ１０／ＤＸ１１の適用形態において、入力の属性または内部変数を、２つ超のGPRs IDへ記憶させ得る。

【０２０９】

[0228]図１４は、本開示の態様による、マージされた頂点シェーディング操作と、ハルシェーディング操作と、ドメインシェーディング操作と、ジオメトリシェーディング操作とを実行するためのグラフィックスプロセシングユニット３３０の例示的なコンポーネントを示す図である。図１４の例は、マージされたＶＳ／ＨＳユニット（パスＩ）およびマージされたＤＳ／ＧＳユニット（パスＩＩ）３３２と、頂点パラメータキャッシュ（ＶＰＣ）３３４と、テッセレータ３３７を有するプリミティブコントローラ（ＰＣ）３３６と、頂点フェッチデコーダ（ＶＦＤ）３３８と、グラフィックスラスタライザ（ＧＲＡＳ）３４０と、レンダーバックエンド（ＲＢ）３４２と、コマンドプロセッサ（ＣＰ）３４４と、ピクセルシェーダ（ＰＳ）３４６とを含む。加えて、図１４は、ＰＭ４パケットバッファ３５０と、頂点オブジェクト３５２と、インデックスバッファ３５４と、システムスクラッチ３５６と、フレームバッファ３５８とを有する、メモリ３４８を含む。

【０２１０】

[0229]図１４の例では、ＶＳ／ＧＳユニット３３２は、上で説明された方式で１つまたは複数のシェーディングユニットによって実装される。ＶＰＣ　３３４は、ストリームアウトデータをメモリ３４８に記憶するために、ストリームアウト機能を実装することができる。ＰＣ　３３６は、変換される必要があり得る頂点を管理することができ、頂点を三角形のプリミティブへと組み立てる。ＶＦＤ　３３８は、頂点のフォーマット状態に基づいて、頂点データをフェッチすることができる。ＧＲＡＳ　３４０は、入力として三角形の頂点を受け取ることができ、三角形の境界内にあるピクセルを出力することができる。プリフェッチパーサ（ＰＦＰ）は、コマンドストリームを事前に復号し、メインＣＰエンジン３４４がデータを必要とするときまでにそのデータの準備ができているように、そのデータをポインタ（たとえば、リソースポインタ）を介してフェッチすることができる。

【０２１１】

[0230]ＤｉｒｅｃｔＸ　１１のためのディスパッチ機構に関して、ドローコールが、Ｃ

Ｐ　３４４によって２つのパスのドローへと分割され得る。パスＩの出力を記憶するために利用可能な記憶容量に基づいて、ドローコールは、複数のサブドローコールへと分割されてよく、各サブドローコールはパスＩとパスＩＩとを有する。パスＩがあるサブドローコールのために実行され、続いて、パスＩＩがそのサブドローコールのために実行されるように、各サブドローコールは、パスの順序を堅持することができる。

【０２１２】

[0231]パスＩでサブドローコールを受け取ると、ＰＣ　３３６は、インデックスをフェッチし、ＶＳ／ＨＳ　３３２を使用してパッチプリミティブのタイプを処理することができる。ＶＳ／ＨＳ　３３２は、パッチごとに、

【数２】

$$HS\_FIBERS\_PER\_PATCH = 2^{\mathrm{ceil}(\log_2(\max(input_{patch}, output_{patch})))}$$

【０２１３】

個のＶＳファイバーを作成し、ウェーブごとに整数個のパッチを収める（ウェーブは所与の量の作業である）。入力における頂点の再使用はない。ＶＳ／ＨＳ　３３２の出力はオフチップからシステムスクラッチ３５６へと転送されるので、位置キャッシュおよびパラメータキャッシュの割り振りはないことがある。

【０２１４】

[0232]ＨＳ＿ＦＩＢＥＲＳ＿ＰＥＲ＿ＰＡＴＣＨに基づいて、ＧＰＵドライバ（図１に示されるＧＰＵドライバ５０のような）は、どれだけの入力プリミティブの頂点がローカルメモリ（ＶＳ／ＨＳ　３３２に対してローカル）に記憶されるかを計算することができる。これは次のように行われ得る。

【数３】

$$HS\_LM\_SIZE\left\lceil \frac{\text{ウェーブ中のファイバー}}{HS\_FIBERS\_PER\_PATCH} \right\rceil * \text{入力パッチ中の制御点} * \text{頂点のサイズ}$$

【０２１５】

ドライバが最終的なデータをメモリ３４８に書き込む前に中間のデータをローカルメモリに書き込むべきである場合、ドライバはまた、追加のサイズをHS_LM_SIZEに加算することができる。そのような追加の空間は、ＨＳがＨＳの複数のステージにおいて（たとえば、ＨＳの不変のステージにおいて）計算された制御ポイントを使用している場合、有用であり得る。このタイプのドローコールを受け取るハイレベルシーケンサ（ＨＬＳＱ）は、どのシェーディングユニットのローカルメモリ（ＬＭ）がGS_LM_SIZEのために十分な記憶容量を有するかを確認することができる。ＨＬＳＱは、そのような割り振りの開始基本アドレス、さらには、割り振られたウェーブによるローカルメモリに対する任意の読取りまたは書込みのアドレスを保持することができる。ＨＬＳＱはまた、ローカルメモリに書き込むときに、割り振られたメモリ内の計算されたオフセットを基本アドレスに追加することができる。

【０２１６】

[0233]システムにより解釈される値（ＳＩＶ）（たとえば、クリップ／間引きの距離、レンダリング対象、ビューポート）も、ＰＳ　３４６へとロードするためにＶＰＣ　３３４に提供され得る。シェーダステージ（たとえば、ＶＳまたはＧＳ）は、条件的にそれらの値を出力することができる。したがって、ＰＳ　３４６がそれらの値を必要とする場合、ＰＳ　３４６は、そのような条件を状態の一部として設定することができる。ＰＳ　３４６がそれらの値を必要とせず、そのような決定がピクセルシェーディング操作の集約の後に行われる場合、これらのＳＩＶを出力する状態は、ＶＳまたはＧＳがドローのときにそれらの値をＶＰＣ　３３４に書き込まないように、リセットされ得る。

【０２１７】

[0234]ヌルのＧＳに対して（ジオメトリシェーダステージが実行されていない場合）、ヌルのＧＳまたはヌルではないＧＳに対する別個のパスが存在しないように、コンパイラは、テンプレートＧＳも作成することができる。このテンプレートＧＳは、ＶＳまたはドメインシェーダ（ＤＳ）の出力をローカルメモリに複製して、ローカルメモリからさらに複製してＶＰＣ　３３４へ出力することができる。これは、ストリームアウトが実行される場合にのみ行われ得る。

【０２１８】

[0235]どのシェーダが実施されているかに応じて、可視性ストリームをビニングして消費するプロセスは異なり得る。たとえば、いくつかのＧＰＵは、タイルまたは「ビン」へとレンダリングされるように画像データを分割することができ、画像全体がレンダリングされるまで、各ビンを連続的に（または場合によっては、同時にもしくは並列に）レンダリングする。画像をビンへと分割することによって、ＧＰＵは、（オンチップメモリが、タイルをレンダリングするために十分な画像データを記憶するのに十分大きい可能性があることを考慮すると）オンチップメモリの要件を低減しつつ、オフチップメモリからのデータの取り出しをより少なくすることを助けることもできる。

【０２１９】

[0236]可視性ストリームに関して、Ｚバッファアルゴリズムが、他のプリミティブによって塞がれるプリミティブを決定するために使用され得る（したがって、レンダリングされる必要はない）。たとえば、ＧＰＵは、（深度の順で）最も後ろにあるプリミティブから、（やはり深度の順で）最も前にあるプリミティブへと作業して、各プリミティブを描くことができる。この例では、いくつかのプリミティブはレンダリングされるが、結局他のプリミティブがその上に描かれることがある。

【０２２０】

[0237]このいわゆる「オーバードロー」の結果として、ＧＰＵは、早期のＺバッファアルゴリズムのテストを実行するように適合されてよく、これは、ＧＰＵがレンダリングを実行するときに無視または迂回されるべき、完全に塞がれる、または視界の中にないプリミティブをＧＰＵが識別することを可能にする。この観点で、ＧＰＵは、各プリミティブおよび／またはオブジェクトに関して可視性情報と呼ばれ得るものを決定するように適合され得る。

【０２２１】

[0238]ＤＸ１０に関して、ビニングパスの間、ＰＣ　３３６は、ＧＳからのすべての出力プリミティブの終了の時点で、「プリミティブの終了」をＧＲＡＳ　３４０に送る。したがって、可視性情報は、入力プリミティブごとに記録される。ストリームアウトは、ビニングパスの間に実行され得る。ＣＰ　３４４は、ビニングパスの終了の時点で、すべてのストリームアウトバッファ関連の情報を読み取ることができる。ジオメトリ関連のクエリカウンタは、ビニングパスの間に更新され得る。

【０２２２】

[0239]可視性パスは、可視性ストリームを読み取り、プリミティブごとの可視性情報が読み取られるとストリームを進めることができる。ストリームがラスタライズされていない場合、可視性パスは飛ばされ得る。それ以外の場合、ＰＣ　３３６は、可視性入力ＧＳプリミティブを確認して、ストリームアウトを何ら伴わずにレンダリングするように処理する。

【０２２３】

[0240]ＤＸ１１に関して、ビニングパスの間、ＰＣ　３３６は、パスＩＩにおけるＧＳからのすべての出力プリミティブの終了の時点で、「プリミティブの終了」をＧＲＡＳ　３４０に送る（たとえば、入力パッチごとに１ビット）。ストリームアウトは、上で説明されたように実行され得る。可視性パスの間、可視性ストリームは、パッチとともにパスＩにおいて処理される（可視性のあるパッチのみが処理され得る）。パスＩＩは、可視のパッチのみを処理し、可視のパッチのみに対してテッセレーション係数をフェッチする。

【０２２４】

[0241]以下に示される表７は、５個の異なる動作のモードの各々に対する、ビニングパスとレンダリングパスとに関する情報を提供する。各モードは、上で説明されたように、単一のハードウェアシェーディングユニットによって実行されるある動作に対応する。
【表７】

表７：異なるモードに対するビニング

モード	VS段階	PS段階	ビニングパス	レンダリングパス
モード_0	VS	PS	プリミティブごとの可視性情報	可視性ストリームを消費する
モード_1	VS+GS	PS	入力プリミティブごとの可視性情報：増幅されたプリミティブに対して、ビンのカバレッジが、入力プリミティブに対する可視性情報を生成するために論理和をとられる	可視性ストリームを消費する
モード_2	VS+HS		可視性情報なし	可視性ストリームを消費する
モード_3	DS	PS	可視性情報が入力パッチごとに生成され、すべてのテッセレーションが行われたプリミティブのビンカバレッジが、入力プリミティブに対する可視性情報を生成するために論理和をとられる	可視性ストリームを消費しない
モード_4	(DS+GS)	PS	可視性情報が入力パッチごとに生成され、すべてのテッセレーションが行われたプリミティブおよびGSプリミティブのビンカバレッジが、入力プリミティブに対する可視性情報を生成するために論理和をとられる	可視性ストリームを消費しない

【０２２５】
[0242]図１５は、本開示の態様による、同じハードウェアシェーディングユニットを使用して２つのレンダリングパスでグラフィックスレンダリングを実行することを示すフロー図である。ＧＰＵ　３６（図１）に関して説明されるが、本開示の態様は、種々の他のコンポーネントを有する多種多様な他のＧＰＵによって実行され得る。
【０２２６】
[0243]図１５の例では、ＧＰＵ　３６は、グラフィックスをレンダリングするために現在実行されているドローコールがテッセレーション操作を含むかどうかを判定する（３８０）。上で説明されたように、テッセレーション操作は、たとえば、ハルシェーダステー

ジ、テッセレーションステージ、およびドメインシェーダステージと関連付けられる操作を含み得る。ドローコールがテッセレーション操作を含まない場合、ＧＰＵ　３６は、単一のパスを伴うレンダリングを実行することができる（３８２）。たとえば、ＧＰＵ　３６は、上で説明された方式で、頂点シェーディングと、ジオメトリシェーディングと、ピクセルシェーディングとを実行することができる。

【０２２７】

[0244]ドローコールがテッセレーション操作を含まない場合、ＧＰＵ　３６は、ＧＰＵメモリ３８のようなローカルのＧＰＵメモリリソースのサイズを決定することができる（３８４）。ＧＰＵ　３６は次いで、ドローコールを複数のサブドローコールへと分割することができる（３８６）。いくつかの例では、各サブドローコールは、上で説明されたパスＩ操作とパスＩＩ操作とを含み得る。たとえば、パスＩ操作は、頂点シェーディング操作とハルシェーディング操作とを含み得るが、パスＩＩ操作は、ドメインシェーディング操作とジオメトリシェーディング操作とを含み得る。

【０２２８】

[0245]各サブドローコールによってレンダリングされるデータの量は、ＧＰＵメモリ３８のサイズに基づいて決定され得る。たとえば、ＧＰＵ　３６は、ＧＰＵ　３６がパスＩ操作によって生成されたデータのすべてをパスＩＩ操作において使用するためにローカルメモリに記憶することが可能であるように、サブドローコールを構成することができる。このようにして、上で説明されたように、ＧＰＵ　３６は、ローカルＧＰＵメモリと、ＧＰＵの外部のメモリとの間で転送されるデータの量を減らすことができ、これにより、レンダリングと関連する遅延が減り得る。

【０２２９】

[0246]サブドローコールを決定した後で、ＧＰＵ　３６は、第１のサブドローコールに対するパスＩ操作を実行することができる（３８８）。上で述べられたように、パスＩ操作は、同じハードウェアシェーディングユニット、たとえば、１つまたは複数のシェーディングユニット４０の各々を使用して、頂点シェーディング操作とハルシェーディング操作とを実行することを含み得る。すなわち、ＧＰＵ　３６は、頂点シェーディングを実行するようにいくつかのシェーディングユニット４０を指定することができ、シェーディングユニット４０の各々は、頂点シェーディング操作とハルシェーディング操作の両方を実行することができる。

【０２３０】

[0247]ＧＰＵ　３６はまた、第１のサブドローコールに対するパスＩＩ操作を実行することができる（３９０）。上で述べられたように、パスＩＩ操作は、同じ１つまたは複数のシェーディングユニット４０を使用して、ドメインシェーディング操作とジオメトリシェーディング操作とを実行することを含み得る。やはり、ＧＰＵ　３６は、頂点シェーディング操作を実行するようにいくつかのシェーディングユニット４０を指定し得るが、シェーディングユニット４０の各々は、シェーディングユニット４０の各々が頂点シェーディング操作と、ハルシェーディング操作と、ドメインシェーディング操作と、ジオメトリシェーディング操作とを実行するように、パスＩＩ操作を実行することができる。

【０２３１】

[0248]ＧＰＵ　３６はまた、サブドローコールに対するピクセルシェーディング操作を実行することができる（３９２）。ＧＰＵ　３６は、１つまたは複数の他のシェーディングユニット４０を使用して、ピクセルシェーディング操作を実行することができる。他の例では、ＧＰＵ　３６は、サブドローコールのすべてが完了した後で、ドローコール全体に対するピクセルシェーディングを実行することができる。

【０２３２】

[0249]ＧＰＵ　３６は次いで、完了したサブドローコールがドローコールの最後のサブドローコールかどうかを判定することができる（３９２）。サブドローコールがドローコールの最後のサブドローコールである場合、ＧＰＵ　３６は、ドローコールと関連付けられるレンダリングされたグラフィックスデータを出力することができる。サブドローコー

ルがドローコールの最後のサブドローコールではない場合、ＧＰＵ　３６は、ステップ３８８に戻り、次のサブドローコールに対するパスＩ操作を実行することができる。

【０２３３】

[0250]図１５に示されるステップは一例として与えられたものにすぎないことを理解されたい。すなわち、図１５に示されるステップは必ずしも示される順序で実行される必要があるとは限らず、より少数の、追加の、または代替のステップが実行され得る。

【０２３４】

[0251]図１６は、本開示の態様による、２つのパスのグラフィックスレンダリングプロセスの第１のパスと関連付けられる、グラフィックスレンダリング操作を実行することを示すフロー図である。図１６に示されるプロセスは、図１５のステップ３８８に関して上で説明されたパスＩ操作に対応し得る。ＧＰＵ　３６（図１）に関して説明されるが、本開示の態様は、種々の他のコンポーネントを有する多種多様な他のＧＰＵによって実行され得る。

【０２３５】

[0252]図１６の例では、上で説明されたように、ＧＰＵ　３６は最初に、グラフィックスレンダリングパイプラインの頂点シェーダステージと関連付けられる頂点シェーディング操作を実行するように、１つまたは複数のシェーディングユニット４０を指定することができる（４００）。頂点シェーディング操作を実行した後、指定されたシェーディングユニット４０の各々は、ハルシェーディング操作のために、シェーディングされた頂点をローカルメモリに記憶することができる（４０２）。ＧＰＵ　３６はまた、ハルシェーディング操作を記録するためのプログラムカウンタを変更し、さらに、１つまたは複数のリソースポインタをハルシェーダリソースオフセットへと変更することができる。たとえば、リソースポインタは、ハルシェーディング操作に割り振られたデータ位置を指し得る。

【０２３６】

[0253]この意味で、シェーディングユニット４０の各々は、動作モードを変更してハルシェーディング操作を実行する。しかしながら、モード変更は、ハルシェーディング操作を実行するようにシェーディングユニット４０を再指定することを含まない。すなわち、ＧＰＵ　３６のコンポーネントは依然として、頂点シェーディング操作のために指定されたシェーディングユニットの１：１のインターフェースのフォーマットとの間でデータを送信し受信するように構成され得る。

【０２３７】

[0254]ＧＰＵ　３６は次いで、上で説明されたように、頂点シェーディング操作を実行した同じシェーディングユニット４０を使用して、グラフィックスレンダリングパイプラインのハルシェーダステージと関連付けられるハルシェーディング操作を実行することができる（４０４）。たとえば、各シェーディングユニット４０は、シェーディングされた頂点に対して動作して、テッセレーションのために使用され得る１つまたは複数の制御ポイントを生成することができる。

【０２３８】

[0255]図１６に示されるステップは一例として与えられたものにすぎないことを理解されたい。すなわち、図１６に示されるステップは必ずしも示される順序で実行される必要があるとは限らず、より少数の、追加の、または代替のステップが実行され得る。

【０２３９】

[0256]図１７は、本開示の態様による、２つのパスのグラフィックスレンダリングプロセスの第２のパスと関連付けられる、グラフィックスレンダリング操作を実行することを示すフロー図である。図１７に示されるプロセスは、図１５のステップ３９０に関して上で説明されたパスＩＩ操作に対応し得る。ＧＰＵ　３６（図１）に関して説明されるが、本開示の態様は、種々の他のコンポーネントを有する多種多様な他のＧＰＵによって実行され得る。

【０２４０】

[0257]図１７の例では、ＧＰＵ　３６は、図１７の操作を実行するために、図１６に関

して上で説明された同じシェーディングユニット４０を使用することができる。たとえば、パスＩＩ操作を実行するために、同じシェーディングユニット４０はまず、上で説明されたように、グラフィックスレンダリングパイプラインのドメインシェーダステージと関連付けられるドメインシェーディング操作を実行することができる（４２０）。すなわち、シェーディングユニット４０は、（ハルシェーダステージからの）制御ポイントに対して動作して、ドメインシェーディングされた頂点を生成することができる。

【０２４１】

[0258]ドメインシェーディング操作を実行した後、指定されたシェーディングユニット４０の各々は、ジオメトリシェーディング操作のために、ドメインシェーディングされた頂点をローカルメモリに記憶することができる（４０２）。ＧＰＵ　３６はまた、ハルシェーディング操作を記録するためのプログラムカウンタを変更し、さらに、１つまたは複数のリソースポインタをハルシェーダリソースオフセットへと変更することができる。図１７の操作が図１６に関して説明されたものに従う例では、これらの機能（たとえば、値をローカルメモリに記憶すること、プログラムカウンタを変更すること、リソースオフセットを変更すること）も、ステップ４２０の前に実行され得る。

【０２４２】

[0259]この意味で、シェーディングユニット４０の各々は、動作モードを変更して、ドメインシェーディング操作とジオメトリシェーディング操作とを実行する。しかしながら、モード変更は、ドメインシェーディング操作とジオメトリシェーディング操作とを実行するようにシェーディングユニット４０を再指定することを含まない。すなわち、ＧＰＵ　３６のコンポーネントは依然として、頂点シェーディング操作のために指定されたハードウェアシェーディングユニットの１：１のインターフェースのフォーマットとの間でデータを送信し受信するように構成され得る。

【０２４３】

[0260]ＧＰＵ　３６は次いで、上で説明されたように、ドメインシェーディング操作を実行した同じシェーディングユニット４０を使用して、グラフィックスレンダリングパイプラインのジオメトリシェーダステージと関連付けられるジオメトリシェーディング操作を実行することができる（４２４）。たとえば、各シェーディングユニット４０は、ドメインシェーディングされた頂点に対して動作して、１つまたは複数のジオメトリシェーディングされた頂点を生成することができる。

【０２４４】

[0261]図１７に示されるステップは一例として与えられたものにすぎないことを理解されたい。すなわち、図１７に示されるステップは必ずしも示される順序で実行される必要があるとは限らず、より少数の、追加の、または代替のステップが実行され得る。

【０２４５】

[0262]図１８は、本開示の態様による、同じハードウェアシェーディングユニットによる実行のために２つ以上のシェーダステージを一緒にパッチされることを示すフロー図である。ＧＰＵ　３６（図１）に関して説明されるが、本開示の態様は、種々の他のコンポーネントを有する多種多様な他のＧＰＵによって実行され得る。

【０２４６】

[0263]図１８の例では、ＧＰＵ　３６は、第１のシェーダステージと関連付けられるシェーディング操作を実行するように、１つまたは複数のハードウェアシェーディングユニット、たとえば、１つまたは複数のシェーディングユニット４０を指定することができる（４４０）。いくつかの例では、第１のシェーダステージは、ＧＰＵ　３６が頂点シェーディング操作を実行するように１つまたは複数のシェーディングユニットを指定するように、頂点を生成するための頂点シェーダステージであり得る。

【０２４７】

[0264]第１のシェーダステージと関連付けられる操作を完了すると、ＧＰＵ　３６は動作モードを切り替えて、同じシェーディングユニット４０が種々の他のシェーディング操作を実行することを可能にし得る（４４２）。たとえば、上で説明されたように、ＧＰＵ

３６は、第２のシェーディング操作を実行するために、プログラムカウンタと１つまたは複数のリソースポインタとを変更することができる。

【０２４８】

[0265]いくつかの例では、ＧＰＵ　３６は、実行されているドローコールと関連付けられるモード情報に基づいて、シェーディングユニット４０の動作モードを切り替えることができる。たとえば、ＧＰＵ　３６のドライバ（ＧＰＵドライバ５０のような）は、どのシェーダステージがドローコールにおいて実行されるべきかを示す、ドローコールに対するモード番号を生成することができる。ＧＰＵ　３６は、このモード番号を使用して、上で説明されたように、パッチコードを実行すると、シェーディングユニットの動作モードを変更することができる。

【０２４９】

[0266]以下に示される表８は、シェーダステージの種々の組合せに対するモード番号を含む、モード情報を一般に示す。

【表８】

表8:シェーダパイプラインの構成

VS	(HS, *TE*,DS)	GS	SO	PS	ドローモード
オン	オフ	オフ	オフ	オン	モード 0
オン	オフ	オフ	オン	オン/オフ	モード 0
オン	オフ	オン	オフ	オン	モード 1
オン	オフ	オン	オン	オン/オフ	モード 1
オン	オン	オフ	オフ	オン	パス 1: モード 2 パス 2 : モード 3
オン	オン	オフ	オン	オン/オフ	パス 1: モード 2 パス 2 : モード 3
オン	オン	オン	オフ	オン	パス 1: モード 2 パス 2 : モード 4
オン	オン	オン	オン	オン	パス 1: モード 2 パス 2 : モード 4

【０２５０】

[0267]表８に示されるように、各モードは、どのシェーダステージがシェーディングユニットによって実行されるかを決定する。したがって、ＧＰＵ　３６は、シェーダ命令を一緒にひと続きにして、同じシェーディングユニット４０が複数のシェーディング操作を実行することを可能にし得る。すなわち、ＧＰＵ　３６は、実行されているドローコールのモード番号に基づいて、適切なシェーダ命令を一緒にパッチされることができる。

【０２５１】

[0268]このようにして、ＧＰＵ　３６は次いで、第１のシェーディング操作を実行するように指定された同じシェーディングユニット４０を用いて、第２のシェーディング操作を実行することができる（４４４）。たとえば、ＧＰＵ　３６は、上の表８に示されるように、頂点シェーディング操作、ハルシェーディング操作、ドメインシェーディング操作、およびジオメトリシェーディング操作の組合せを実行することができる。

【０２５２】

[0269]図１８に示されるステップは一例として与えられたものにすぎないことを理解されたい。すなわち、図１８に示されるステップは必ずしも示される順序で実行される必要があるとは限らず、より少数の、追加の、または代替のステップが実行され得る。

【０２５３】

[0270]上で説明されたいくつかの例は、頂点シェーディング操作を実行するようにハードウェアシェーディングユニットを最初に指定することと、同じハードウェアシェーディングユニットを用いて他のシェーディング操作を実行することへと移行することとを含むが、本開示の技法はこのように限定されないことを理解されたい。たとえば、ＧＰＵは、種々の他のシェーディング操作を実行するようにハードウェアシェーディングユニットのセットを最初に指定することができる。すなわち、３個の異なるシェーディング操作を実行するようにハードウェアシェーディングユニットをＧＰＵが指定することを可能にするシステムにおいて、ＧＰＵは、頂点シェーディング操作と、ハルシェーディング操作と、ピクセルシェーディング操作とを実行するように、ハードウェアシェーディングユニットを指定することができる。この例では、ＧＰＵは、ハルシェーディング操作を実行するように１つまたは複数のハードウェアシェーディングユニットを最初に指定することができるが、上で説明されたように、同じハードウェアシェーディングユニットによってドメインシェーディング操作とジオメトリシェーディング操作とを実行することもできる。種々の他の操作の組合せも可能である。

【０２５４】

[0271]１つまたは複数の例では、説明された機能は、ハードウェア、ソフトウェア、ファームウェア、またはそれらの任意の組合せで実装され得る。ソフトウェアで実装される場合、機能は、非一時的コンピュータ可読媒体を備える製造品に１つまたは複数の命令またはコードとして記憶され得る。コンピュータ可読媒体はコンピュータデータ記憶媒体を含み得る。データ記憶媒体は、本開示で説明された技法の実装のための命令、コードおよび／またはデータ構造を取り出すために１つまたは複数のコンピュータあるいは１つまたは複数のプロセッサによってアクセスされ得る任意の利用可能な媒体であり得る。限定ではなく、例として、そのようなコンピュータ可読媒体は、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、ＣＤ－ＲＯＭまたは他の光ディスクストレージ、磁気ディスクストレージまたは他の磁気ストレージデバイス、フラッシュメモリ、あるいは、命令またはデータ構造の形態の所望のプログラムコードを搬送または記憶するために使用されコンピュータによってアクセスされ得る、任意の他の媒体を備え得る。本明細書で使用されるディスク（disk）およびディスク（disc）は、コンパクトディスク（disc）（ＣＤ）、レーザーディスク（登録商標）（disc）、光ディスク（disc）、デジタル多用途ディスク（disc）（ＤＶＤ）、フロッピー（登録商標）ディスク（disk）およびブルーレイ（登録商標）ディスク（disc）を含み、ディスク（disk）は、通常、データを磁気的に再生し、ディスク（disc）は、データをレーザーで光学的に再生する。上記の組合せもコンピュータ可読媒体の範囲内に含まれるべきである。

【０２５５】

[0272]コードは、１つまたは複数のＤＳＰ、汎用マイクロプロセッサ、ＡＳＩＣ、ＦＰＧＡ、あるいは他の等価な集積回路またはディスクリート論理回路など、１つまたは複数のプロセッサによって実行され得る。さらに、いくつかの態様では、本明細書で説明される機能は、専用のハードウェアモジュールおよび／またはソフトウェアモジュールの内部で与えられ得る。また、本技法は、１つまたは複数の回路または論理要素中で完全に実装され得る。

【０２５６】
[0273]本開示の技法は、ワイヤレスハンドセット、集積回路（ＩＣ）、またはＩＣのセット（たとえば、チップセット）を含む、多種多様なデバイスまたは装置において実装され得る。本開示では、開示される技法を実行するように構成されたデバイスの機能的態様を強調するために、様々なコンポーネント、モジュール、またはユニットが説明されたが、それらのコンポーネント、モジュール、またはユニットは、必ずしも異なるハードウェアユニットによる実現を必要とするとは限らない。むしろ、上で説明されたように、様々なユニットが、好適なソフトウェアおよび／またはファームウェアとともに、上記で説明した１つまたは複数のプロセッサを含めて、コーデックハードウェアユニットにおいて組み合わせられるか、または相互動作ハードウェアユニットの集合によって与えられ得る。
【０２５７】
[0274]様々な例が説明された。これらおよび他の例は、以下の特許請求の範囲内に入る。
以下に、本願出願の当初の特許請求の範囲に記載された発明を付記する。
［Ｃ１］
グラフィックスをレンダリングする方法であって、
レンダリングパイプラインの第１のシェーダステージと関連付けられる第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定することと、
前記第１のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えることと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行することと
を備える、方法。
［Ｃ２］
動作モードを切り替えることは、前記第１のシェーディング操作と前記第２のシェーディング操作とを備えるドローコールのモードを決定することを備える、
Ｃ１に記載の方法。
［Ｃ３］
前記第２のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えることと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第３の異なるシェーダステージと関連付けられる第３のシェーディング操作を実行することと
をさらに備える、Ｃ２に記載の方法。
［Ｃ４］
前記第３のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えることと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第４の異なるシェーダステージと関連付けられる第４のシェーディング操作を実行することと
をさらに備える、Ｃ３に記載の方法。
［Ｃ５］
動作モードを切り替えることは、前記第１のシェーディングステージと関連付けられる入力／出力インターフェースを維持しながら動作モードを切り替えることを備える、
Ｃ１に記載の方法。

［Ｃ６］
動作モードを切り替えることは、前記第２のシェーディング操作のためにプログラムカウンタと１つまたは複数のリソースポインタとを切り替えることを備える、
Ｃ１に記載の方法。
［Ｃ７］
前記第１のシェーディング操作と関連付けられる第１の命令は、前記第２のシェーディング操作と関連付けられる第２の命令に依存しないように、前記第１の命令が前記第２の命令とは独立にコンパイルされる、
Ｃ１に記載の方法。
［Ｃ８］
１つまたは複数のシステムにより生成される値のために、ローカルメモリ中の１つまたは複数の所定の位置を確保することをさらに備え、前記システムにより生成される値は、前記第１のシェーディング操作および前記第２のシェーディング操作において使用される、
Ｃ７に記載の方法。
［Ｃ９］
前記第１のシェーディング操作の結果をローカルメモリに記憶することと、前記グラフィックスプロセシングユニットの外部に位置するオフチップメモリにアクセスすることなく、前記第１のシェーディング操作の前記結果に対して前記第２のシェーディング操作を実行することと
をさらに備える、Ｃ１に記載の方法。
［Ｃ１０］
前記第１のシェーディング操作を実行することは、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することを備え、前記第２のシェーディング操作を実行することは、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するジオメトリシェーディング操作を実行することを備える、
Ｃ１に記載の方法。
［Ｃ１１］
前記第１のシェーディング操作を実行することは、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することを備え、前記第２のシェーディング操作を実行することは、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するハルシェーディング操作を実行することを備える、
Ｃ１に記載の方法。
［Ｃ１２］
前記第１のシェーディング操作を実行することは、頂点を生成するためにドメインシェーディング操作を実行することを備え、前記第２のシェーディング操作を実行することは、前記ドメインシェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するためにジオメトリシェーディング操作を実行することを備える、
Ｃ１に記載の方法。
［Ｃ１３］
１つまたは複数のプロセッサを備える、グラフィックスをレンダリングするためのグラフィックスプロセシングユニットであって、前記１つまたは複数のプロセッサは、
レンダリングパイプラインの第１のシェーダステージと関連付けられる第１のシェーディング操作を実行するように、前記グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定することと、
前記第１のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替え、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセ

シングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行することとを行うように構成される、
グラフィックスプロセシングユニット。
［Ｃ１４］
動作モードを切り替えるために、前記１つまたは複数のプロセッサは、前記第１のシェーディング操作と前記第２のシェーディング操作とを備えるドローコールのモードを決定するように構成される、
Ｃ１３に記載のグラフィックスプロセシングユニット。
［Ｃ１５］
前記１つまたは複数のプロセッサは、
前記第２のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えることと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第３の異なるシェーダステージと関連付けられる第３のシェーディング操作を実行することとを行うようにさらに構成される、
Ｃ１４に記載のグラフィックスプロセシングユニット。
［Ｃ１６］
前記１つまたは複数のプロセッサは、
前記第３のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えることと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第４の異なるシェーダステージと関連付けられる第４のシェーディング操作を実行することとを行うようにさらに構成される、
Ｃ１５に記載のグラフィックスプロセシングユニット。
［Ｃ１７］
動作モードを切り替えるために、前記１つまたは複数のプロセッサは、前記第１のシェーディングステージと関連付けられる入力／出力インターフェースを維持しながら動作モードを切り替えるように構成される、
Ｃ１３に記載のグラフィックスプロセシングユニット。
［Ｃ１８］
動作モードを切り替えるために、前記１つまたは複数のプロセッサが、前記第２のシェーディング操作のためにプログラムカウンタと１つまたは複数のリソースポインタとを切り替えるように構成される、
Ｃ１３に記載のグラフィックスプロセシングユニット。
［Ｃ１９］
前記第１のシェーディング操作と関連付けられる第１の命令は、前記第２のシェーディング操作と関連付けられる第２の命令に依存しないように、前記第１の命令が前記第２の命令とは独立にコンパイルされる、
Ｃ１３に記載のグラフィックスプロセシングユニット。
［Ｃ２０］
前記１つまたは複数のプロセッサは、１つまたは複数のシステムにより生成される値のために、前記グラフィックスプロセシングユニットのローカルメモリ中の１つまたは複数の所定の位置を確保することを行うようにさらに構成され、前記システムにより生成される値は、前記第１のシェーディング操作および前記第２のシェーディング操作において使用される、
Ｃ１９に記載のグラフィックスプロセシングユニット。
［Ｃ２１］

前記１つまたは複数のプロセッサは、前記第１のシェーディング操作の結果を前記グラフィックスプロセシングユニットのローカルメモリに記憶することと、前記グラフィックスプロセシングユニットの外部に位置するオフチップメモリにアクセスすることなく、前記第１のシェーディング操作の前記結果に対して前記第２のシェーディング操作を実行することとを行うようにさらに構成される、
Ｃ１３に記載のグラフィックスプロセシングユニット。
［Ｃ２２］
前記第１のシェーディング操作を実行するために、前記ハードウェアシェーディングユニットは、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することを行うように構成され、前記第２のシェーディング操作を実行するために、前記ハードウェアシェーディングユニットは、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するジオメトリシェーディング操作を実行することを行うように構成される、
Ｃ１３に記載のグラフィックスプロセシングユニット。
［Ｃ２３］
前記第１のシェーディング操作を実行するために、前記ハードウェアシェーディングユニットは、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することを行うように構成され、前記第２のシェーディング操作を実行するために、前記ハードウェアシェーディングユニットは、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するハルシェーディング操作を実行することを行うように構成される、
Ｃ１３に記載のグラフィックスプロセシングユニット。
［Ｃ２４］
前記第１のシェーディング操作を実行するために、前記ハードウェアシェーディングユニットは、頂点を生成するためにドメインシェーディング操作を実行することを行うように構成され、前記第２のシェーディング操作を実行するために、前記ハードウェアシェーディングユニットは、前記ドメインシェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成することを行うように構成される、
Ｃ１３に記載のグラフィックスプロセシングユニット。
［Ｃ２５］
グラフィックスをレンダリングするための装置であって、
レンダリングパイプラインの第１のシェーダステージと関連付けられる第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定するための手段と、
前記第１のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えるための手段と、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行するための手段と
を備える、装置。
［Ｃ２６］
動作モードを切り替えるための前記手段は、前記第１のシェーディング操作と前記第２のシェーディング操作とを備えるドローコールのモードを決定するための手段を備える、
Ｃ２５に記載の装置。
［Ｃ２７］
前記第２のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えるための手段と、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリング

パイプラインの第３の異なるシェーダステージと関連付けられる第３のシェーディング操作を実行するための手段と
をさらに備える、Ｃ２６に記載の装置。
［Ｃ２８］
前記第３のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えるための手段と、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第４の異なるシェーダステージと関連付けられる第４のシェーディング操作を実行するための手段と
をさらに備える、Ｃ２７に記載の装置。
［Ｃ２９］
動作モードを切り替えるための前記手段は、前記第１のシェーディングステージと関連付けられる入力／出力インターフェースを維持しながら動作モードを切り替えるための手段を備える、
Ｃ２５に記載の装置。
［Ｃ３０］
動作モードを前記切り替えることは、前記第２のシェーディング操作のためにプログラムカウンタと１つまたは複数のリソースポインタとを切り替えるための手段を備える、
Ｃ２５に記載の装置。
［Ｃ３１］
前記第１のシェーディング操作と関連付けられる第１の命令は、前記第２のシェーディング操作と関連付けられる第２の命令に依存しないように、前記第１の命令が前記第２の命令とは独立にコンパイルされる、
Ｃ２５に記載の装置。
［Ｃ３２］
１つまたは複数のシステムにより生成される値のために、ローカルメモリ中の１つまたは複数の所定の位置を確保するための手段をさらに備え、前記システムにより生成される値は、前記第１のシェーディング操作および前記第２のシェーディング操作において使用される、
Ｃ３１に記載の装置。
［Ｃ３３］
前記第１のシェーディング操作の結果をローカルメモリに記憶し、前記グラフィックスプロセシングユニットの外部に位置するオフチップメモリにアクセスすることなく、前記第１のシェーディング操作の前記結果に対して前記第２のシェーディング操作を実行するための手段をさらに備える、
Ｃ２５に記載の装置。
［Ｃ３４］
前記第１のシェーディング操作を実行するための前記手段は、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行するための手段を備え、前記第２のシェーディング操作を実行するための前記手段は、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するジオメトリシェーディング操作を実行するための手段を備える、
Ｃ２５に記載の装置。
［Ｃ３５］
前記第１のシェーディング操作を実行するための前記手段は、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行するための手段を備え、前記第２のシェーディング操作を実行するための前記手段は、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するハルシェーディング操作を実行するための手段を備える、

Ｃ２５に記載の装置。
［Ｃ３６］
前記第１のシェーディング操作を実行するための前記手段は、頂点を生成するドメインシェーディング操作を実行するための手段を備え、前記第２のシェーディング操作を実行するための前記手段は、前記ドメインシェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するジオメトリシェーディング操作を実行するための手段を備える、
Ｃ２５に記載の装置。
［Ｃ３７］
命令を記憶した非一時的コンピュータ可読媒体であって、前記命令は、実行されると、１つまたは複数のプロセッサに、
レンダリングパイプラインの第１のシェーダステージと関連付けられる第１のシェーディング操作を実行するように、グラフィックスプロセシングユニットのハードウェアシェーディングユニットを指定することと、
前記第１のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えることと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第２の異なるシェーダステージと関連付けられる第２のシェーディング操作を実行することと
を行わせる、非一時的コンピュータ可読媒体。
［Ｃ３８］
動作モードを切り替えるために、前記命令は、前記１つまたは複数のプロセッサに、前記第１のシェーディング操作と前記第２のシェーディング操作とを備えるドローコールのモードを決定することを行わせる、
Ｃ３７に記載の非一時的コンピュータ可読媒体。
［Ｃ３９］
前記１つまたは複数のプロセッサに、
前記第２のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えることと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第３の異なるシェーダステージと関連付けられる第３のシェーディング操作を実行することと
を行わせる命令をさらに備える、Ｃ３８に記載の非一時的コンピュータ可読媒体。
［Ｃ４０］
前記１つまたは複数のプロセッサに、
前記第３のシェーディング操作が完了すると、前記ハードウェアシェーディングユニットの動作モードを切り替えることと、
前記第１のシェーディング操作を実行するように指定された前記グラフィックスプロセシングユニットの前記ハードウェアシェーディングユニットを用いて、前記レンダリングパイプラインの第４の異なるシェーダステージと関連付けられる第４のシェーディング操作を実行することと
を行わせる、Ｃ３９に記載の非一時的コンピュータ可読媒体。
［Ｃ４１］
動作モードを切り替えるために、前記命令は、前記１つまたは複数のプロセッサに、前記第１のシェーディングステージと関連付けられる入力／出力インターフェースを維持しながら動作モードを切り替えることを行わせる、
Ｃ３７に記載の非一時的コンピュータ可読媒体。
［Ｃ４２］

動作モードを切り替えるために、前記命令は、前記１つまたは複数のプロセッサに、前記第２のシェーディング操作のためにプログラムカウンタと１つまたは複数のリソースポインタとを切り替えることを行わせる、
Ｃ３７に記載の非一時的コンピュータ可読媒体。
［Ｃ４３］
前記第１のシェーディング操作と関連付けられる第１の命令が、前記第２のシェーディング操作と関連付けられる第２の命令に依存しないように、前記第１の命令が前記第２の命令とは独立にコンパイルされる、
Ｃ３７に記載の非一時的コンピュータ可読媒体。
［Ｃ４４］
前記１つまたは複数のプロセッサに、１つまたは複数のシステムにより生成される値のために、前記グラフィックスプロセシングユニットのローカルメモリ中の１つまたは複数の所定の位置を確保することを行わせる命令をさらに備え、前記システムにより生成される値は、前記第１のシェーディング操作および前記第２のシェーディング操作において使用される、
Ｃ４３に記載の非一時的コンピュータ可読媒体。
［Ｃ４５］
前記１つまたは複数のプロセッサに、前記第１のシェーディング操作の結果を前記グラフィックスプロセシングユニットのローカルメモリへ記憶することと、前記グラフィックスプロセシングユニットの外部に位置するオフチップメモリにアクセスすることなく、前記第１のシェーディング操作の前記結果に対して前記第２のシェーディング操作を実行することとを行わせる命令をさらに備える、
Ｃ３７に記載の非一時的コンピュータ可読媒体。
［Ｃ４６］
前記第１のシェーディング操作を実行するために、前記命令は、前記ハードウェアシェーディングに、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することと、前記第２のシェーディング操作を実行するために、前記命令は、前記ハードウェアシェーディングユニットに、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するジオメトリシェーディング操作を実行することと
を行わせる、Ｃ３７に記載の非一時的コンピュータ可読媒体。
［Ｃ４７］
前記第１のシェーディング操作を実行するために、前記命令は、前記ハードウェアシェーディングユニットに、頂点シェーディングされた頂点を出力するために、入力された頂点をシェーディングする頂点シェーディング操作を実行することと、前記第２のシェーディング操作を実行するために、前記命令は、前記ハードウェアシェーディングユニットに、前記頂点シェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の制御ポイントを生成するハルシェーディング操作を実行することと
を行わせる、Ｃ３７に記載の非一時的コンピュータ可読媒体。
［Ｃ４８］
前記第１のシェーディング操作を実行するために、前記命令は、前記ハードウェアシェーディングユニットに、頂点を生成するドメインシェーディング操作を実行することと、前記第２のシェーディング操作を実行するために、前記命令は、前記ハードウェアシェーディングユニットに、前記ドメインシェーディングされた頂点の１つまたは複数に基づいて１つまたは複数の新たな頂点を生成するジオメトリシェーディング操作を実行することと
を行わせる、Ｃ３７に記載の非一時的コンピュータ可読媒体。

【図１】

図 1

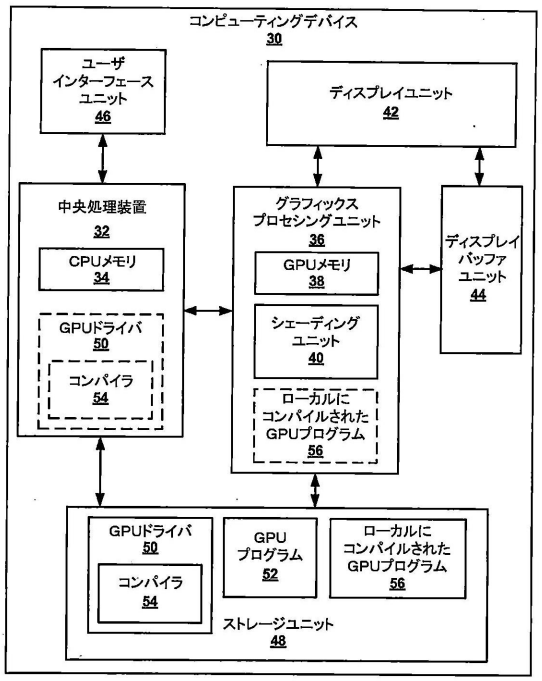

FIG. 1

【図２】

図 2

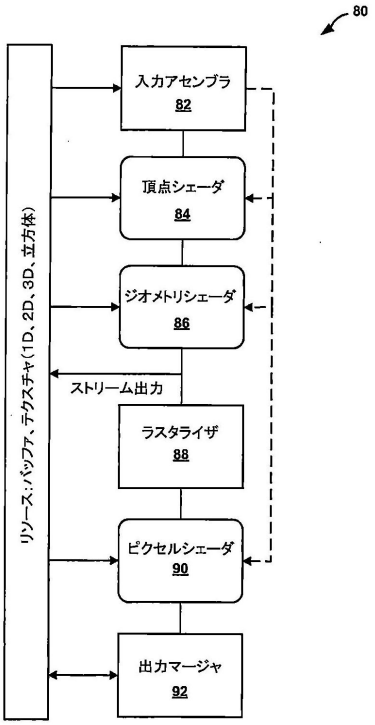

FIG. 2

【図３Ａ】

図 3A

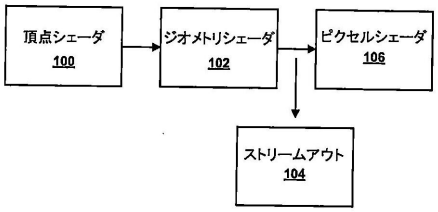

FIG. 3A

【図３Ｂ】

図 3B

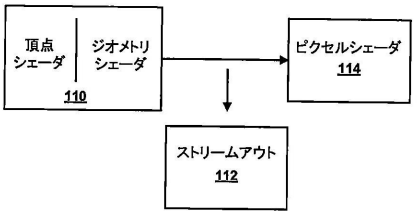

FIG. 3B

【図４】

図 4

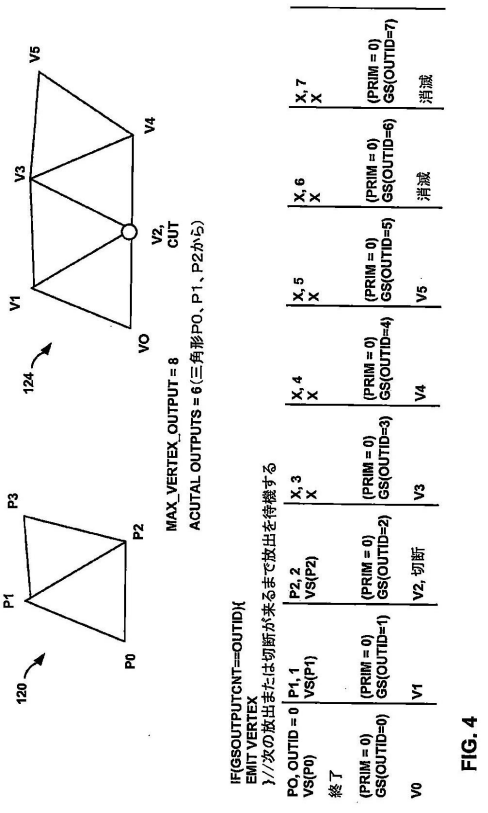

FIG. 4

【図５Ａ】

図 5A

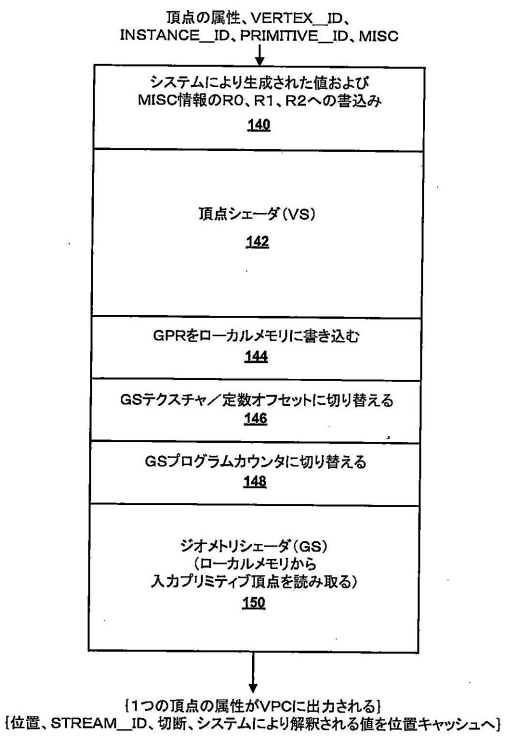

FIG. 5A

【図５Ｂ】

図 5B

```
VS:   'Normal' VS code  //instanceIDを使用でき、initialはprimitiveiD、
                        //misc、rel_primを記憶する
                        //GPRにおける左側のデータ
                        //DX9スタイルのシェーダであれば終了し、
      END_FINAL;        //それ以外の場合は無視し、シェーダが終了する場合にgprを
                        //VPCに書き込むのはSPであることに留意されたい

      LM_Vs_offset = rel_primID*PRIM_SIZE + rel_vertex*VERT_SIZE;

      GPR中の頂点データをLM_Vs_offsetに書き込む

      CHMSK             //第2のステージに対するcov_mask_1を使用し、
                        //第2のシェーダリソースに切り替える（リソースptrを
                        //GSリソースオフセットに切り替える）

      CHSH              //GSプログラムカウンタおよび状態オフセットに切り替える

GS:   For (vertex_id= 0; vertex_id++; vertex_id <= max_input)
        {   Load from LM using  offset = rel_primID*PRIM_SIZE + vertex_id *
            VERT_SIZE
            //HLSQによって計算される基本アドレスを使用してLMからフェッチする
        }
      for(streamid =0;streamid++;streamid<4);
      cut[streamid] = true;
      gsoutount = 0;

 //「通常の」GSコード
      {.. using primitiveID, GSInstanceID
      CUT(streamid):      { Cut[streamid] = true;}
      EMIT(streamid): {
            If(Gsoutcount==GsoutvertID){
                  oPos = vertex.pos; oColor =vertex.clr;
                  oMisc = (cut[streamid],streamid)
                  gsoutcount = maxoutputvertexcount;
                  go to END_FINAL;   // オプション
            }
            cut[streamid] = false;
            gsoutcount++;
      }

      If (gsoutcount < maxoutputvertexcount)
            KILL PRIM;  //任意の「残りの」ファイバーを削除する

END_FINAL;
```

FIG. 5B

【図６】

図 6

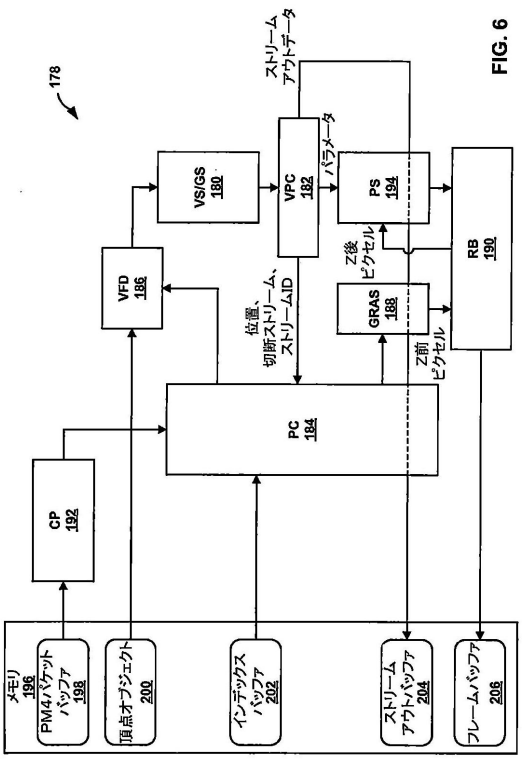

FIG. 6

【図７】

図 7

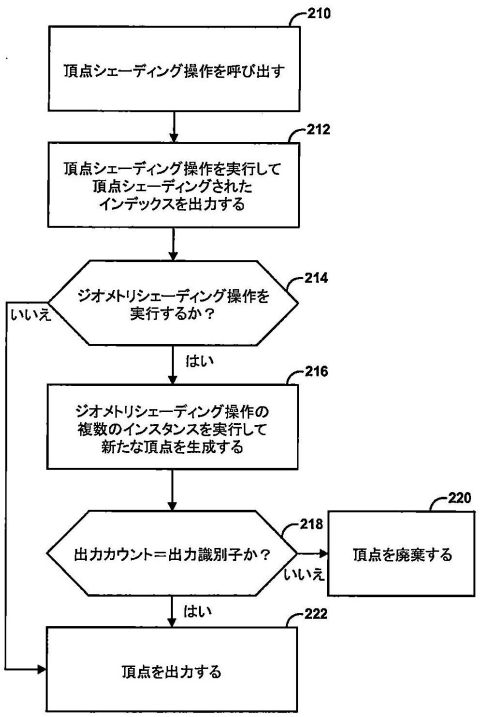

FIG. 7

【図８】

図 8

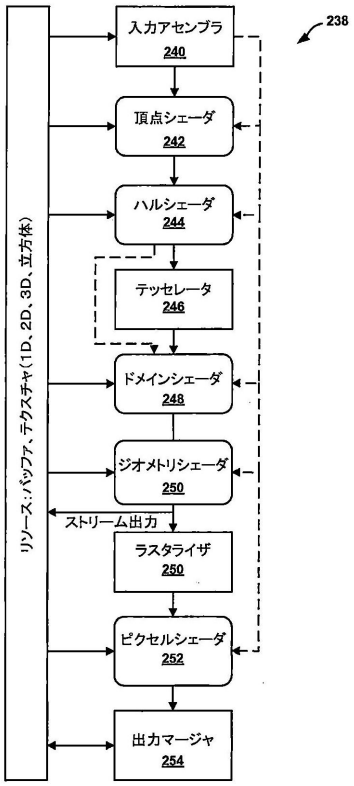

FIG. 8

【図９】

図 9

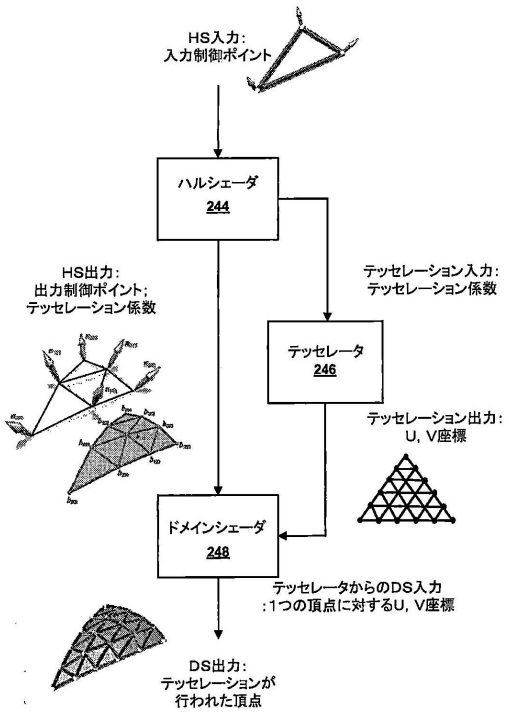

FIG. 9

【図１０Ａ】

図 10A

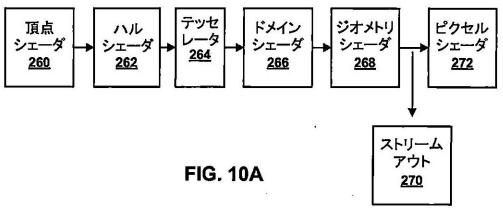

FIG. 10A

【図１０Ｂ】

図 10B

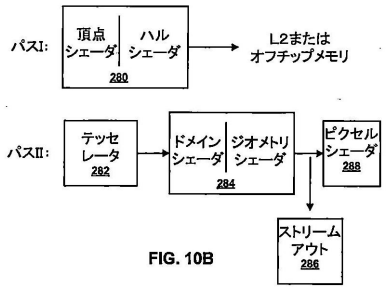

FIG. 10B

【図１１】

図 11

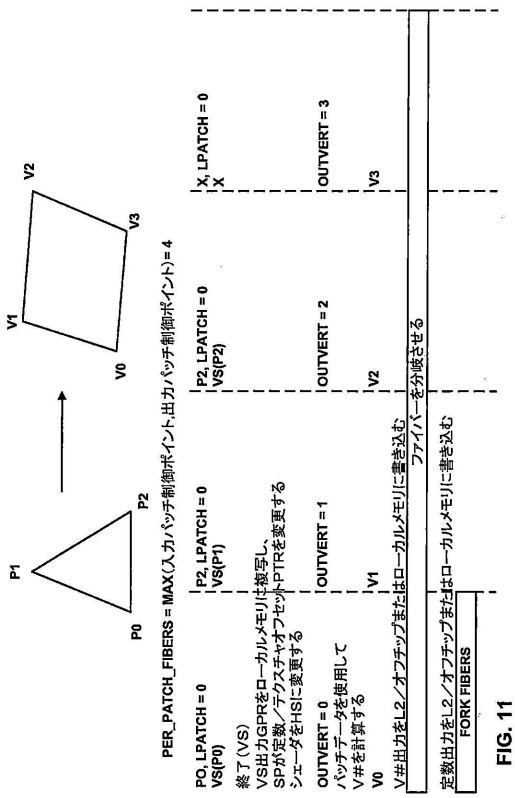

FIG. 11

【図１２Ａ】

図 12A

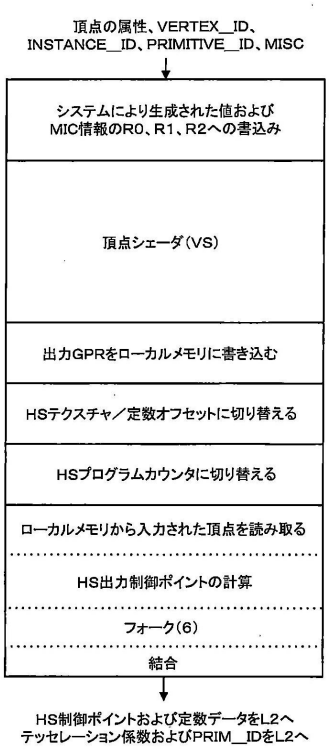

FIG. 12A

【図１２Ｂ】

図 12B

```
//VFDが頂点インデックスおよび有効な頂点に基づいて頂点データをフェッチする。
//VFDは頂点データを有効な頂点にロードしなければならない
VS: 'Normal' VS  code //instanceIDを使用できる
                      //GPRにおける左側のデータ
    END_FINAL;        //DX9スタイルのシェーダであれば終了し、それ以外の場合は無視する

    LM_Vs_offset = rel_primID*PRIM_SIZE + rel_vertex*VERT_SIZE;

    GPR中の頂点データをLM__Vs__offsetに書き込む

    CHMSK;    //第2のステージに対するcov__mask__1を使用し、
              //第2のシェーダリソースに切り替える（リソースptrを
              //GSリソースオフセットに切り替える）

    CHSH      //シェーダプログラムカウンタおよび状態オフセットに切り替える
              //（HS  PC／オフセット）

HS:  For  (vertex_id= 0; vertex_id++; vertex_id <= max_input)
     {    Load from LM using  offset = rel_primID*PRIM_SIZE + vertex_id *
          VERT_SIZE
          //HLSQによって計算される基本アドレスを使用してLMからフェッチする
     }

     For (hsoutcount =0;hscountcount++;hsoutcount<=maxout) //これはfork
                                   //（#__output__pnts）を使用して行われ得る
     {    //HS出力頂点を計算するHSコード
          ... primitiveIDを使用して
       Mem_offset = rel_patchid*OUTPUT_PRIM_SIZE +  outVertID *
       OUTPUT_VERTEX_SIZE + SCRATCH_MEMORY_CONTROL_BASE

       If (hscount == outVertID)        //パッチ出力点が順番に
                                        //処理されれば完了
          (export computed vertex to mem_offset)
     }
     For (fork_id = outVertID;fork_id += WAVE_OUT_SIZE;fork_id <n)
     {
          Fork(fork_id)      //並列計算のためにfibers__per__primの間に
                             //fork（n）命令を広める
     }
       if (outVertID == 0)
     {    Join               //outVertID＝0を伴う1つのみのファイバーを使用する
          Tf_offset = rel_patchid* TF_OUT_SIZE+ tess_attr +
          SCRATCH_MEMORY_TESS_BASE; //テッセレーション係数メモリに
                                    //テッセレーション係数を書き込む
          Write primitiveID
          Write tessellation factors
     }
      END_FINAL;
```

FIG. 12B

【図１３Ａ】

図 13A

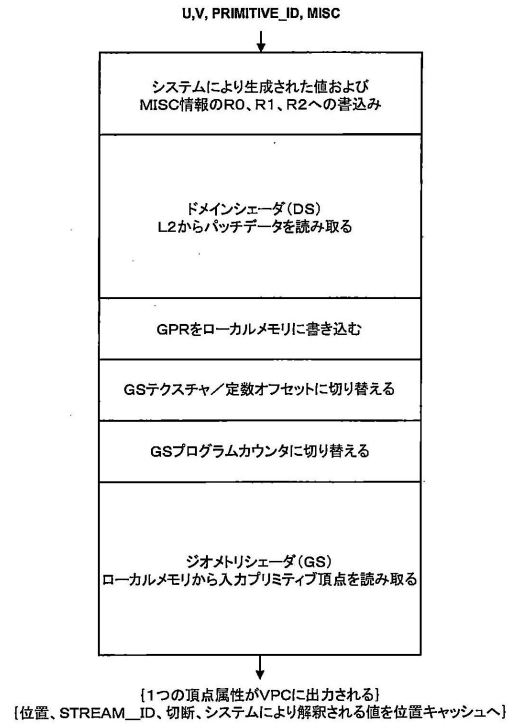

FIG. 13A

【図１３Ｂ】

図 13B

```
//VFDは何らデータをフェッチしない
// InstanceID = rel_patchID
//（u, v）は異なるgprで来る
//このgprに対するrel__patchIDを実際に表すinstanceIDをSPがロードできるように、
//gpr中にpatchIDがあることを予想してコンパイラはDSコードをコンパイルした
DS:  Compute  patch_ptr  =  rel_patchid  *  OUTPUT_PRIM_SIZE  +
     SCRATCH_MEMORY_CONTROL_BASE.
     'Normal' DS  code
     //patch__ptr＋OUTPUT__VERTEX__SIZE×control__pointIDを
       使用してHS出力制御ポイントにアクセスする
     //patch__ptr＋OUTPUT__VERTEX__SIZE×num__output__control__pts
       を使用して、（u, v）を使用してHS定数データにアクセスして、テッセレーション
       が行われた点を計算する
          //GPR中の左側のデータ
END_FINAL; //GSを伴わないDX11のDSシェーダであれば終了し、それ以外の場合は無視し、
           //シェーダが終了する場合にgprをVPCに書き込むのはSPであることに留意されたい

     LM_Vs_offset = rel_primID*PRIM_SIZE + rel_vertex*VERT_SIZE;

     Write vertex data  in GPR to LM_Vs_offset

     CHMSK;    //リソースptrをGSリソースオフセットに切り替える
               //第2のステージに対するcov__mask__1を使用する

     CHSH      //GSプログラムカウンタおよび状態オフセットに切り替える

GS:  For  (vertex_id= 0; vertex_id++; vertex_id <= max_input)
       {    Load from LM using  offset = rel_primID*PRIM_SIZE + vertex_id *
            VERT_SIZE
            //HLSQによって計算される基本アドレスを使用してLMからフェッチする
       }
     for(streamid =0;streamid++;streamid<4);
     cut[streamid] = true;
     gsoutount = 0;

//「通常の」GSコード
     {.. using primitiveID, GSInstanceID
     CUT(streamid):      { Cut[streamid] = true;}
     EMIT(streamid): {
          If(Gsoutcount==GsoutvertID){
               oPos = vertex.pos; oColor =vertex.clr;
               oMisc = (cut[streamid],streamid)
               gsoutcount = maxoutputvertexcount;
               go to END_FINAL;  // オプション
          }
          cut[streamid] = false;
          gsoutcount++;
     }

     If (gsoutcount < maxoutputvertexcount)
          KILL PRIM;  //eliminate any 'leftover' fibers

END_FINAL;
```

FIG. 13B

【図１４】

図 14

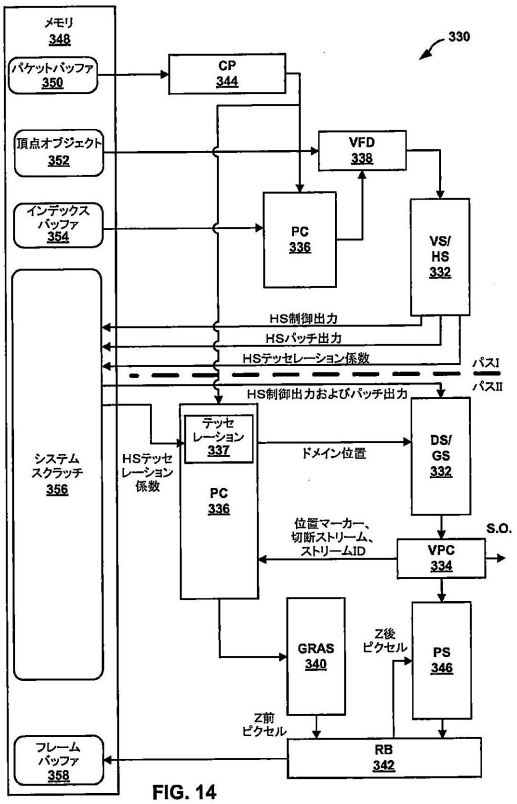

FIG. 14

【図１５】

図 15

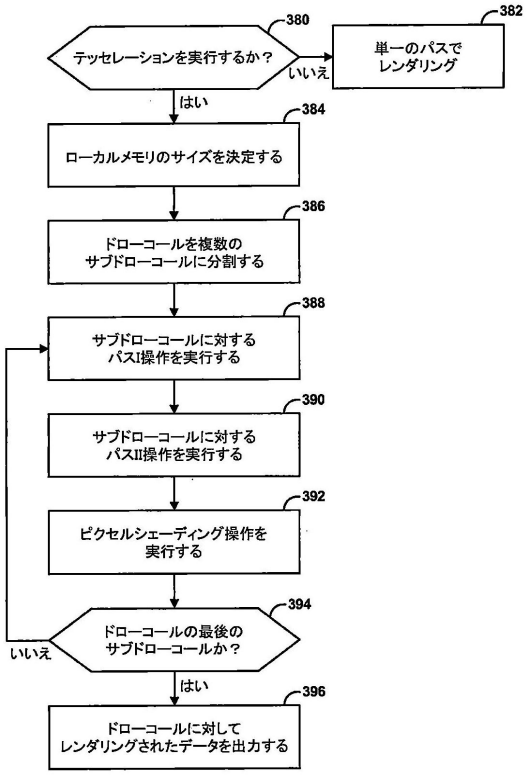

FIG. 15

【図１６】

図 16

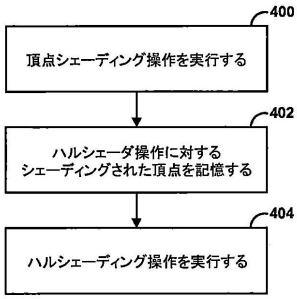

FIG. 16

【図１７】

図 17

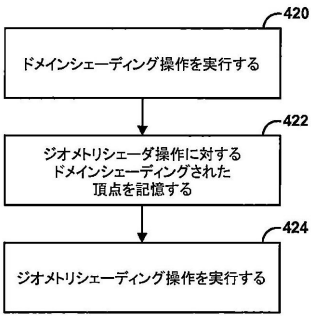

FIG. 17

【図１８】

図 18

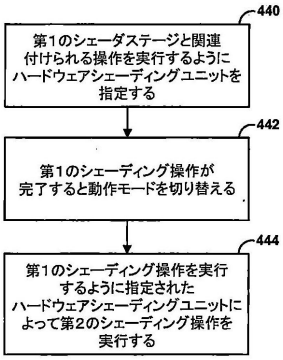

FIG. 18

フロントページの続き

(31)優先権主張番号 61/620,333
(32)優先日 平成24年4月4日(2012.4.4)
(33)優先権主張国 米国(US)
(31)優先権主張番号 13/829,900
(32)優先日 平成25年3月14日(2013.3.14)
(33)優先権主張国 米国(US)

(74)代理人 100153051
弁理士 河野 直樹
(74)代理人 100140176
弁理士 砂川 克
(74)代理人 100158805
弁理士 井関 守三
(74)代理人 100179062
弁理士 井上 正
(74)代理人 100124394
弁理士 佐藤 立志
(74)代理人 100112807
弁理士 岡田 貴志
(74)代理人 100111073
弁理士 堀内 美保子
(72)発明者 ゴエル、ビネート
アメリカ合衆国、カリフォルニア州 ９２１２１、サン・ディエゴ、モアハウス・ドライブ ５７７５
(72)発明者 グルバー、アンドリュー・イー.
アメリカ合衆国、カリフォルニア州 ９２１２１、サン・ディエゴ、モアハウス・ドライブ ５７７５

審査官 岡本 俊威

(56)参考文献 FOLEY T ET AL, SPARK: MODULAR, COMPOSABLE SHADERS FOR GRAPHICS HARDWARE, ACM TRANSACTIONS ON GRAPHICS (TOG), 米国, ACM, ２０１１年 7月, VOL:30 NR:4, PAGE(S):107.1 - 107.12, ＵＲＬ, http://dx.doi.org/10.1145/1964921.1965002

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｔ １５／００－１５／８７