(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3803425号
(P3803425)
(45)発行日　平成18年8月2日(2006.8.2)　(24)登録日　平成18年5月12日(2006.5.12)

(51)Int.Cl.　F I
G06F 12/08 (2006.01)　G06F 12/08 507L
G06F 12/08 559D
G06F 12/08 519Z

請求項の数 2　(全 64 頁)

(21)出願番号　特願平8-124368
(22)出願日　平成8年5月20日(1996.5.20)
(65)公開番号　特開平8-328954
(43)公開日　平成8年12月13日(1996.12.13)
審査請求日　平成14年10月16日(2002.10.16)
(31)優先権主張番号　471,653
(32)優先日　平成7年6月6日(1995.6.6)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　ダレル・エヌ・エモット
アメリカ合衆国80526コロラド州フォート・コリンズ、モス・クリーク・ドライブ　3931

審査官　清木　泰

最終頁に続く

(54)【発明の名称】キャッシュ・ディレクトリ

(57)【特許請求の範囲】

【請求項１】

Ｎが１より大きいとして、Ｎ個の読取りキャッシュ・タグをＸ個の記憶されたキャッシュ・タグと比較して、Ｎ個の読取りキャッシュ・タグの各々がＸ個の記憶されたキャッシュ・タグの１つと一致するか否かを判断する機能を持つキャッシュ・ディクトリを操作する方法であって、(a)主ディレクトリにＸ個のキャッシュ・タグを記憶するステップと、(b)Ｎ以上Ｘ未満であるＭ個のキャッシュ・タグを含む上記Ｘ個のキャッシュ・タグのサブセットを、ミニディレクトリに記憶するステップと、(c)Ｎ個の読取りキャッシュ・タグの各々を、ミニディレクトリに記憶されたＭ個のキャッシュ・タグと同時に比較するステップと、(d)記憶されたＭ個のキャッシュ・タグの１つと一致するＮ個の読取りキャッシュ・タグの各々がＸ個の記憶されたキャッシュ・タグの１つと一致することを標示するステップと、(e)Ｎ個の読取りキャッシュ・タグの少なくとも１つが記憶されたＭ個のキャッシュ・タグのいずれとも一致しない場合、Ｎ個の読取りキャッシュ・タグの少なくとも１つを、主ディレクトリに記憶されたＸ個のキャッシュ・タグと比較するステップと、を含む方法。

【請求項２】

キャッシュ・メモリと連動し、Ｘ個のキャッシュ・タグを記憶し、Ｎが１より大きいとして、１回のキャッシュ・ディレクトリ・アクセスにおいてＮ個の読取りキャッシュ・タグに応答して該Ｎ個の読取りキャッシュ・タグの各々について対応するキャッシュ・ブロック・インデックスを提供するキャッシュ・ディレクトリであって、上記キャッシュ・タグ

の各々がキャッシュ・データの１ブロックに対応し、上記キャッシュ・メモリが上記キャッシュ・データのＸ個のブロックを記憶し、Ｎ個の読取りキャッシュ・タグの各々の１つについて、該キャッシュ・ディレクトリによって提供される上記キャッシュ・ブロック・インデックスがＮ個の読取りキャッシュ・タグの１つに対応するキャッシュ・データ・ブロックが記憶されているキャッシュ・メモリ位置を識別するように構成され、上記Ｘ個のキャッシュ・タグのサブセットでありかつ、Ｎ個以上であるＭ個のキャッシュ・タグを記憶し、その記憶されているＭ個のキャッシュ・タグの各々を上記Ｎ個の読取りキャッシュ・タグの各々と同時に比較し、上記Ｎ個の読取りキャッシュ・タグの少なくとも１つと一致する上記記憶されているＭ個のキャッシュ・タグの各々１つに関して対応するブロック・インデックスを提供する完全連想型のミニディレクトリと、上記Ｘ個のキャッシュ・タグを記憶し、Ｎ個未満であるＰ個の読取りキャッシュ・タグを上記Ｘ個の読取りキャッシュ・タグの各々と同時に比較し、上記Ｐ個の読取りキャッシュ・タグの１つと一致する上記記憶されているＸ個のキャッシュ・タグの各々１つに関して対応するブロック・インデックスを提供する完全連想型の主ディレクトリと、上記Ｎ個の読取りキャッシュ・タグをミニディレクトリと初期的に比較して、ミニディレクトリと一致したＮ個の読取りキャッシュ・タグの各々に関してブロック・インデックスを提供し、また、ミニディレクトリと一致しなかった上記Ｎ個の読取りキャッシュ・タグの各々を主ディレクトリと比較して各々に関するブロック・インデックスを提供する制御手段と、を備えるキャッシュ・ディレクトリ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、一般的に、テクスチャ・マッピングを行うコンピュータ・グラフィックス・システムに関するもので、特にテクスチャ・マッピング・データを記憶するキャッシュ・メモリ・システムに関するものである。

【０００２】

【従来の技術】

コンピュータ・グラフィックス・システムは、２次元表示画面上でオブジェクトのグラフィック表現を表示するために一般に使用される。現在のコンピュータ・グラフィックス・システムは、高度に細密な表現を提供することができ、種々のアプリケーションにおいて使用されている。

【０００３】

典型的なコンピュータ・グラフィックス・システムにおいては、表示画面に表現されるべきオブジェクトは、複数のグラフィックス・プリミティブに分解される。プリミティブは、グラフィックス・ピクチャの基本コンポーネントであって、点、線、ベクトルおよび三角形のような多角形を含む場合がある。典型的ハードウェア／ソフトウェア方式は、画面上に表現される１つまたは複数のオブジェクトの画像を表現するグラフィックス・プリミティブを、２次元表示画面上にレンダリング(rendering)または描画するように実施される。

【０００４】

レンダリングされるべき３次元オブジェクトを定義するプリミティブは、典型的には、ホスト・コンピュータによってプリミティブ・データとして定義され提供される。例えば、あるプリミティブが三角形であるとすれば、ホスト・コンピュータは、三角形の各頂点のｘ，ｙ，ｚ座標および各頂点のＲ、Ｇ、Ｂカラー値として、そのプリミティブを定義する。レンダリング・ハードウェアは、プリミティブ・データを補間して、各ピクセルを表現するため画面上オンにする表示画面ピクセルおよび各ピクセルに関するＲ，Ｇ，Ｂ値を計算する。

【０００５】

初期のグラフィックス・システムは、複雑な３次元のオブジェクトを表現またはモデル化する場合十分に現実的な形態で画像を表示することができなかった。そのようなシステム

によって表示される画像は、極端になめらかな表面をしていて、モデル化されたオブジェクトに存在するテクスチャ、凹凸、スクラッチ、陰影およびその他の表面細部に欠けていた。

【０００６】

このため、表面細部が改善された画像を表示するいくつかの方法が開発された。テクスチャ・マッピング手法はそのような方法の１つであって、これは、先ず、テクスチャ(textureすなわち表面模様)と呼ばれるソース画像を３次元オブジェクトの面にマッピングし、次に、その３次元オブジェクを２次元グラフィックス表示画面にマップして、その結果生成される画像を表示する。一般にマップされたテクスチャの細部の面特性は、カラー、鏡面反射、ベクトル摂動、反射、透明性、陰影、表面不規則性および階調度を含む。

【０００７】

テクスチャ・マッピングは、テクスチャの１つまたは複数のテクセル(texelすなわち点要素)を、テクスチャがマップされる先のオブジェクトの表示部分の点要素（すなわちピクセルまたは画素）に対応させることを意味する。テクスチャ・マッピング・ハードウェアは、テクスチャ・マップにおけるテクセルがオブジェクトを表現する表示画面のピクセルに対応する形態を標示する情報を伝統的に備えている。テクスチャ・マップにおける各テクセルは、２次元テクスチャ・マップにおけるその位置を識別するＳおよびＴ座標によって定義される。各ピクセル毎に、該ピクセルに対応する１つまたは複数のテクセルが、テクスチャ・マップから取り出され、表示画面上でテクスチャ化されたオブジェクトを表現するため該ピクセルに対して生成される最終的Ｒ，Ｇ，Ｂ値と統合される。

【０００８】

オブジェクト・プリミティブのピクセルの各々は、オブジェクトのあらゆる表現についてテクスチャ・マップにおける単一のテクセルと１対１の対応関係でマップすることはできない点理解されるべきである。例えば、オブジェクトが表示画面上で表示される場合表示位置に接近すればする程、オブジェクトは一層大きく表現される。オブジェクトが表示画面上大きく表現される程、テクスチャの表現はより詳細になる。従って、オブジェクトが表示画面の大部分を使う時、オブジェクトを表示画面で表現するため多数のピクセルが使用され、オブジェクトを表現する各ピクセルは、テクスチャ・マップにおける１つのテクセルと１対１の対応関係をもってマップされるか、あるいは、単１のテクセルが複数のピクセルに対応することもある。しかし、逆にオブジェクトが表示画面の相対的に小さい部分を占める時、オブジェクトを表示画面で表現するため相対的に少ないピクセルが使用され、テクスチャの表現は粗くなり、従って、各ピクセルは複数のテクセルに対応することになる。テクスチャがオブジェクトの小さい部分にマップされる場合、各ピクセルは、また、複数のテクセルにマップされる可能性がある。この結果、典型的には、ピクセルの各々毎に、複数のテクセルに対応し、かつ、そのピクセルへ対応するテクセルの平均を表すテクセル・データが計算される。

【０００９】

典型的テクスチャ・マッピング・ハードウェア・システムは、レンダリングされるオブジェクトに関連するテクスチャを表現するデータを記憶するローカル・メモリを含む。上述のように、１つのピクセルが複数のテクセルに対応する場合がある。平均値を生成するためテクスチャ・マッピング・ハードウェアがローカル・メモリから１つのピクセルに対応する多数のテクセルを読み取らねばならないとすれば、多数のメモリ読み出し動作と多数のテクセル値の平均算出演算が必要となり、これは時間浪費的であってシステム処理能力を低下させる原因となるであろう。

【００１０】

この問題を解決するため、各テクスチャ毎に一連のＭＩＰマップを作成し、レンダリングされるテクスチャのＭＩＰマップをテクスチャ・マッピング・ハードウェアのローカル・メモリに記憶する方式が開発された。あるテクスチャに関するＭＩＰマップは、テクスチャ・マップに直接対応する基本マップならびにそれ以外の一連のフィルタされたマップを含み、この場合、連続するマップは前後で、２の自乗のサイズで減少する。図１は１組の

ＭＩＰマップの例を示す。（注：ＭＩＰは、multum in parvoの頭文字をとったもので、「小さい場所の多数の事柄」を意味する）。図１のＭＩＰマップは、サイズ的に８×８テクセルである基本マップ１００の他、それぞれサイズ４×４、２×２ならびに１×１テクセルである一連のマップ１０２、１０４および１０８を含む。

【００１１】

４×４マップ１０２は、基本マップ１００をフィルタ（すなわち比例減少）することによって生成されるが、具体的には、マップ１０２のテクセルの各々は、基本マップ１０の４個のテクセルの平均値に対応する。例えば、マップ１０２のテクセル１１０は、マップ１００のテクセル１１２ないし１１５の平均に等しく、マップ１０２のテクセル１１８および１２０は、マップ１００のテクセル１２１ないし１２４の平均およびテクセル１２５ないし１２８の平均にそれぞれ等しい。２×２マップ１０４は、例えばマップ１０４のテクセル１３０がマップ１０２のテクセル１１０、１１８、１１９および１２０の平均に等しくなるように、マップ１０２をフィルタすることによって同様に生成される。マップ１０８の単一（１×１）テクセルは、マップ１０４の４つのテクセルを平均することによって生成される。

【００１２】

従来技術のグラフィックス・システムは、一般的に、ホスト・コンピュータの主メモリからテクスチャ・マッピング・ハードウェアのローカル・メモリへ、表示画面にレンダリングされるプリミティブに関して使用されるべきテクスチャに対する完全な一連のＭＩＰマップをダウンロードする。従って、テクスチャ・マッピング・ハードウェアは、一連のＭＩＰマップのうちのいずれのマップからもテクスチャ・データをアクセスすることができる。特定のピクセルについてテクセル・データを提供するためどのマップにアクセスすべきかは、当該ピクセルがマップするテクセルの数に基づいて決定される。例えば、ピクセルがテクスチャ・マップの単一のテクセルと１対１の対応関係でマップする場合は、基本マップ１００がアクセスされる。しかし、ピクセルが、４、１６または６４のテクセルへマップするとすれば、マップ１０２、１０４および１０８が、それぞれテクスチャ・マップの４、１６、６４個のテクセルの平均を表すテクセル・データを保持しているので、それらのマップがそれぞれアクセスされる。

【００１３】

１つのピクセルが選択されたマップのどの１つのテクセルにも直接対応せず、複数のテクセルの間に落ちることがある。このような場合、グラフィックス・システムによっては、テクセル・データを正確に生成するため双線形補間(bilinear interpolation)を使う。１つのピクセルが、１つのＭＩＰマップの複数のテクセル・エントリに対応する場合、使用されるテクセル・データは、最も近いテクセル・エントリの加重平均である。このように、ピクセルに対応するテクセル・データは、単一のマップにおける４つのテクセル・エントリの加重平均とすることができる。例えば、ピクセルがマップ１０２の１３２で示される位置に対応する場合、そのピクセルに対応するテクセル・データは、テクセル１１０、１１８、１９９および１２０の加重平均である。

【００１４】

また、ピクセルが一連のＭＩＰマップのどのマップにも直接対応せず、２つのマップの間に落ちることもある。例えば、あるピクセルが、テクスチャ・マップの１より大で４未満の数のテクセルに対応することもある。このような場合、所望のテクセル・データを作成するため、グラフィックス・システムによっては、２つの近接するＭＩＰマップの間で補間を行う。例えば、上述のように、１つのピクセルがテクスチャ・マップの１より大で４未満の数のテクセルに対応するような場合、マップ１００および１０２によって提供されるテクセル・データが補間され、該ピクセルに関するテクセル・データが作成される。単一のマップにおける複数のテクセル・エントリの上述の補間と組み合わせられる時、この方式は、３線形補間(trilinear interpolation)として知られるもので、２つの近接するマップのそれぞれにおける４個の近接するテクセル、すなわち８個のテクセルの加重平均として特定のピクセルに対する所望のテクセル・データを生成することができる。

【００１５】
【発明が解決しようとする課題】
上述のように、従来技術のテクスチャ・マッピング・システムは、システムによってレンダリングされるべきプリミティブに関連するいかなるテクスチャについても、一連のＭＩＰマップを（たとえ一部がアクセスされないとしても）すべてダウンロードする。アクセスされないＭＩＰマップのダウンロードは、アクセスされるマップのうちの使用されない部分のダウンロードと共に、システム資源の浪費であり、その帯域幅（すなわちデータ伝送率）を減少させる。
【００１６】
更に、種々の動作が異なるオブジェクト・プリミティブに関して同時に実行されるように、パイプライン形態を実施するテクスチャ・マッピング・システムもある。しかし、そのようなシスステムでは、１つのテクスチャに関する一連のＭＩＰマップが非常に大きくなる可能性がある。大部分のシステムが使用するローカル・メモリは、そのような大規模のＭＩＰマップを１度に１つしか記憶することができない。従って、プリミティブをレンダリングする際テクスチャの切り替えが行われる場合、システムは新たな一連のＭＩＰマップをダウンロードしなければならない。新しいテクスチャ・データをテクスチャ・マッピング・ハードウェアのローカル・メモリにロードするために使用されるデータ経路は、典型的には、システムのプリミティブ・レンダリング・パイプラインを通過する。従って、新しいテクスチャをマップする必要がある場合、新たな一連のＭＩＰマップがダウンロードされる前にプリミティブ・レンダリング・パイプラインをフラッシュ(flush)することができなければならない。一連のＭＩＰマップがダウンロードされたなら、パイプラインに再びデータを送らなければならない。このように、新しいテクスチャが必要とされる度毎にプリミティブ・レンダリング・パイプラインをフラッシュしなければならないので、システムの帯域幅が低下する。
【００１７】
【課題を解決するための手段】
本発明の１つの実施形態において、キャッシュ・メモリに連動するキャッシュ・ディレクトリが作成される。該キャッシュ・ディレクトリは、Ｘ個のキャッシュ・タグを記憶し、１回のキャッシュ・ディレクトリ・アクセスにおいてＮ個の読み取りキャッシュ・タグに応答して、Ｎ個の読み取りキャッシュ・タグの各々について対応するキャッシュ・ブロック・インデックスを提供する。各キャッシュ・タグは、１つのブロックのキャッシュ・データに対応する。キャッシュ・メモリは、キャッシュ・メモリの位置を識別するＮ個の読み取りキャッシュ・タグのそれぞれ１つについてキャッシュ・ディレクトリによって提供されるブロック・インデックスを用いて、Ｘブロックのキャッシュ・データを記憶する。この場合、記憶されるキャッシュ・データのブロックは、Ｎ個の読み取りキャッシュ・タグの１つに対応する。上記キャッシュ・ディレクトリは、ミニディレクトリ、主ディレクトリおよび制御回路を含む。ミニディレクトリは、Ｘ個のキャッシュ・タグのサブセットである（少なくともＮに等しい）Ｍ個のキャッシュ・タグを記憶する。ミニディレクトリは完全連想型ディクトリであって、Ｍ個の記憶されたキャッシュ・タグの各々をＮ個の読取りキャッシュ・タグの各々と同時に比較する。ミニディレクトリは、Ｎ個の読取りキャッシュ・タグの少なくとも１つと一致するＭ個の記憶されたキャッシュ・タグのそれぞれ１つについて、対応するブロック・インデックスを提供する。主ディレクトリは、Ｘ個のキャッシュ・タグを記憶する。主ディレクトリは完全連想型ディクトリであって、Ｐ個の読み取りキャッシュ・タグをＸ個のキャッシュ・タグの各々と同時に比較する（ここでＰはＮより小である）。主ディレクトリは、Ｐ個の読取りキャッシュ・タグの１つと一致するＸ個の記憶されたキャッシュ・タグのそれぞれ１つについて、対応するブロック・インデックスを提供する。制御回路は、Ｎ個の読取りキャッシュ・タグをミニディレクトリと初期的に比較して、ミニディレクトリと一致するＮ個の読み取りキャッシュ・タグの各々についてブロック・インデックスを提供し、ミニディレクトリと一致しないＮ個の読取りキャッシュ・タグの各々を主ディレクトリと比較し、それぞれについてブロック・インデ

ックスを提供する。
【００１８】
本発明の別の１つの実施形態に従えば、Ｎ個の読取りキャッシュ・タグをＸ個の記憶されたキャッシュ・タグと比較してＮ個の読取りキャッシュの各々がＸ個の記憶されたキャッシュ・タグの１つと一致するか否かを判断するキャッシュ・ディレクトリを動作させる方法が提供される。該方法は、(a)主ディレクトリにＸ個のキャッシュ・タグを記憶するステップ、(b)上記記憶されたＸ個のキャッシュ・タグのサブセットとして、Ｎ以上でかつＸ未満であるＭ個のキャッシュ・タグをミニディレクトリに記憶するステップ、(c)Ｎ個の読取りキャッシュ・タグを、ミニディレクトリに記憶されたＭ個のキャッシュ・タグと比較するステップ、(d)Ｍ個の記憶されたキャッシュ・タグの１つと一致するＮ個の読み取りキャッシュ・タグの各々がＸ個の記憶されたキャッシュ・タグの１つと一致することを標示するステップ、および(e)Ｎ個の読み取りキャッシュ・タグの少くとも１つがＭ個の記憶されたキャッシュ・タグのどれとも一致しない場合、Ｎ個の読み取りキャッシュ・タグの少くとも１つを主ディレクトリに記憶されたＸ個のキャッシュ・タグと比較するステップを含む。
【００１９】
【発明の実施の形態】
Ｉ．システム概要
図２は、テクスチャ・データをローカルに記憶するためのキャッシュ・メモリを備えるテクスチャ・マッピング・ハードウェアを含む本発明のグラフィックス・システムの１つの実施形態のブロック図である。図示されている実施例が、基板ならびにチップの数、細分化の形態、バス幅およびデータ転送速度に関して単なる典型例にすぎないことは理解されるべきである。図示される以外のその他多数の形態を実施することは可能であるる。図２に示されているように、本システムは、フロントエンド基板１０、テクスチャ・マッピング基板１２およびフレーム・バッファ基板１４を含む。フロントエンド基板は、５２ビット幅バス１６を経由してホスト・コンピュータ１５と通信する。フロントエンド基板は、レンダリング（描画）すべきプリミティブを、バス１６経由でホスト・コンピュータから受け取る。プリミティブは、ｘ，ｙ，ｚベクトル座標データ、Ｒ，Ｇ，Ｂカラー・データおよびテクスチャＳ，Ｔ座標によって指定される。これらデータは、すべて、例えばプリミティブが三角形であれば頂点のようなプリミティブの部分に関するデータである。次に、プリミティブを３次元で表すデータが、フロントエンド基板１０によって、テクスチャ・マッピング基板１２およびフレーム・バッファ基板１４に対して８５ビット幅バス１８を経由して与えられる。テクスチャ・マッピング基板は、プリミティブを表現する画面表示ピクセルを計算するため受け取ったプリミティブ・データを補間し、該プリミティブのピクセル毎に対応するテクスチャ・データを決定する。計算結果のテクスチャ・データは、図２では単純化のため１本線として示されている５本の５５ビット幅バス２８を経由してフレーム・バッファ基板に送られる。
【００２０】
フレーム・バッファ基板１４もまた、フロントエンド基板１０から受け取ったプリミティブ・データを補間し、各プリミティブを表現する表示画面上のピクセルを計算し、各ピクセル毎にオブジェクトのカラーを決定する。次に、フレーム・バッファ基板は、上記オブジェクト・カラー値をテクスチャ・マッピング基板から送られたテクスチャ・データとピクセル毎に結合して、各ピクセル毎に画像Ｒ，Ｇ，Ｂを生成する。表示画面（図示されてない）のピクセルを制御するため各ピクセルに関するＲ，Ｇ，Ｂカラー制御信号が、Ｒ，Ｇ，Ｂライン２９経由でそれぞれ与えられ、テクスチャ・マップ・プリミティブを表現する画像が表示画面上に表示される。
【００２１】
フロントエンド基板１０、テクスチャ・マッピング基板１２およびフレーム・バッファ基板１４の各々はパイプライン化され、複数のプリミティブに対して同時に動作する。テクスチャ・マッピングおよびフレーム・バッファ基板が、フロントエンド基板によって前に

提供されたプリミティブに対して動作を行う際、フロントエンド基板は、基板１２および１４のパイプラインがいっぱいにならない限り、新しいプリミティブに対して動作し基板１２および１４へ提供し続ける。フロントエンド基板１０は、分配器チップ３０、３次元（３Ｄ）加速器チップ３２Ａ、３２Ｂならびに３２Ｃ、２次元（２Ｄ）加速器チップ３４および集線器チップ３６を含む。分配器チップ３０は、Ｘ，Ｙ，Ｚ座標およびカラー・プリミティブ・データをバス１６経由でホスト・コンピュータから受け取り、３次元プリミティブ・データを３次元加速器チップ３２Ａ、３２Ｂおよび３２Ｃに均等に分配する。このような形態で、３つのグループのプリミティブが同時に処理されることによって、システムの帯域幅が増加される。データは、４０ビット幅バス３８Ａを経由して３次元加速器チップ３２Ａおよび３２Ｂに送られ、４０ビット幅バス３８Ｂを経由してチップ３２Ｃに送られる。バス３８Ａおよび３８Ｂは、６０ＭＨＺの伝送率でデータを伝送し、２つの３次元加速器チップをサポートするために十分な帯域幅を提供する。２Ｄプリミティブ・データは、４４ビット幅バス４０を経由して４０ＭＨＺの伝送率で２Ｄ加速速器チップ３４へ送られる。

【００２２】

３次元加速器チップの各々は、受け取ったプリミティブを定義するｘ，ｙ，ｚ座標を、対応する画面空間座標に変換し、画面空間座標に対するオブジェクトＲ，Ｇ，Ｂ値およびテクスチャＳ，Ｔ値を決定し、プリミティブの四辺形を三角形へ分解し、各三角形を定義するため三角形平面方程式を計算する。各３次元加速器チップは、また、複数ウィンドウが表示される時、あるいは、プリミティブの一部が表示画面上に表される視野を越えて広がる時、画像の正確な画面表示を確実にするため視野クリッピング動作を実行する。３次元加速器３２Ａおよび３２Ｂからの出力データは４４ビット幅バス４２Ａを経由して、また３次元加速器３２Ｃからの出力データは４４ビット幅バス４２Ｂを経由して、それぞれ、集線器チップ３６へ６０ＭＨＺの伝送率で送られる。２次元加速器３４からの出力データは４６ビット幅バス４４を経由して集線器チップ３６へ４０ＭＨＺの伝送率で送られる。集線器チップ３６は、３次元加速器チップ３２Ａ－３２Ｃから受け取った３次元プリミティブ出力データを結合し、分配器チップ３０による分配の前の元の順序にプリミティブを配列し直し、結合したプリミティブ出力データをバス１８を経由してテクスチャ・マッピング基板およびフレーム・バッファ基板に送る。

【００２３】

テクスチャ・マッピング基板１２は、テクスチャ・マッピング・チップ４６、および、好ましくはキャッシュ・メモリとして構成されるローカル・メモリ４８を備える。本発明の１つの好ましい実施形態において、ローカル・メモリは、後述の理由から、複数のＳＤＲＡＭチップ（すなわち同期ダイナミックＲＡＭ）から形成される。詳細は後述するが、キャッシュ・メモリ４８は、フレーム・バッファ基板においてレンダリングされるプリミティブに関連するテクスチャＭＩＰマップ・データを記憶する。テクスチャＭＩＰマップ・データは、ホスト・コンピュータ１５の主メモリ１７から、バス４０を経由して、２Ｄ加速器チップ３４を通過し、２４ビット幅バス２４を経由して、キャッシュ・メモリ４８にダウンロードされる。

【００２４】

テクスチャ・マッピング・チップ４６は、表示画面上で描画（レンダリング）されるべきプリミティブを表すプリミティブ・データをバス１８経由で連続的に受け取る。上述のように、３次元加速器チップ３２Ａ－３２Ｃから送られるプリミティブは、点、線分および三角形を含む。テクスチャ・マッピング基板は、点または線分に関してはテクスチャ・マッピングを実行せず、三角形プリミティブについてのみ実行する。三角形プリミティブを表現するデータは、少くとも１つの頂点に関するｘ，ｙ，ｚオブジェクト・ピクセル座標、少くとも１つの頂点のオブジェクト・カラーＲ，Ｇ，Ｂ値、少くとも１つの頂点に対応するテクスチャ・マップ部分のＳ，Ｔ座標、および三角形の平面方程式を含む。テクスチャ・マッピング・チップ４６は、オブジェクト・ピクセルｚ座標およびオブジェクト・カラーＲ，Ｇ，Ｂ値を無視する。チップ４６は、ｘ，ｙピクセル座標を補間し、プリミティ

ブを表現する各ｘ，ｙ画面表示ピクセルに対応するＳおよびＴ座標を補間する。各ピクセル毎に、テクスチャ・マッピング・チップは、ピクセルに対応するテクスチャＭＩＰマップ部分をキャッシュ・メモリから取り出し、複数のテクセルの加重平均を含むテクスチャ・データを該ピクセルについて計算する。

【００２５】

１つの典型的実施形態において、キャッシュは１ブロックが２５６×２５６テクセルからなる６４ブロックのテクセルを記憶する。従来技術のシステムのテクスチャ・マッピング・ハードウェアで使われるローカル・メモリと異なって、本発明のキャッシュ・メモリは、レンダリングされるプリミティブに対応する（大規模な）テクスチャの一連のＭＩＰマップ全体を記憶しなくてもよい。むしろ、本発明のキャッシュ・メモリは、ある１時点をとると、その時点でプリミティブをレンダリングする場合一連のＭＩＰマップの実際に使用される特定部分のみを記憶する。従って、ほとんどのアプリケーションの場合、ある１時点で、全テクスチャ・データのうちレンダリングされる画像に関する部分だけがキャッシュ・メモリに記憶される。

【００２６】

各テクスチャに関する完全な一連のＭＩＰマップは、ホスト・コンピュータ１５の主メモリ１７に記憶される。レンダリングされるプリミティブの各ピクセルについて、テクスチャ・マッピング・チップ４６は、キャッシュ・メモリ４８のディレクトリにアクセスして、テクスチャＭＩＰマップの対応する１つまたは複数のテクセルが現在キャッシュに存在するか否かを判断する。対応するテクセルがアクセス時点でキャッシュ・メモリに存在する場合、キャッシュ・ヒット(cache hit)が発生し、テクセルがキャッシュ・メモリから読み取られ、テクスチャ・マッピング・チップ４６によってフレーム・バッファ基板に渡されるテクスチャ・データが計算される。

【００２７】

しかし、プリミティブ・ピクセルについて対応するテクセルがアクセス時点で存在しない場合、キャッシュ・ミス(cache miss)が発生する。キャッシュ・ミスが発生すると、プリミティブをレンダリングするために必要とされるテクスチャＭＩＰマップ部分データが、ホスト・コンピュータ１５の主メモリ１７からキャッシュ・メモリ４８へダウンロードされ、既に記憶されているなにがしかのデータを置き換えることになろう。しかしながら、レンダリングされるプリミティブに関する一連のＭＩＰマップ全体をダウンロードする従来技術のテクスチャ・マッピング・システムと異なって、本発明は、現時点でプリミティブをレンダリングする場合一連のＭＩＰマップの実際に使用される特定部分または現時点でレンダリングされている部分のみをダウンロードする。詳細は後述するが、キャッシュ・ミスが発生すると、ホスト・コンピュータ１５のテクスチャ割り込み管理機構を始動する割込み制御信号が、テクスチャ・マッピング・チップ４６によって生成される。割込み制御信号は、ライン９４を経由して分配器チップ３０へ送られ、次に、ライン９５を経由してホスト・コンピュータへ送られる。

【００２８】

要求されたテクスチャ・データが、ホスト・コンピュータによって主メモリから読み出され、３Ｄプリミティブ・レンダリング・パイプラインをバイパスして、バス２４経由でテクスチャ・マッピング基板のメモリ４８へダウンロードされる。このように、キャッシュ・ミス割込みが発生する時、キャッシュ・ミスを引き起こしたプリミティブに関連するテクスチャ・データが主メモリ１７からダウンロードされている間、フロントエンド基板が、３次元プリミティブに関する動作を継続し、テクスチャ・マッピング・チップおよびフレーム・バッファ基板へバス１８を経由してプリミティブ・データを出力することができる。従来技術のテクスチャ・マッピング・システムと対照的に、テクスチャ・マッピング・ハードウエアへのテクスチャ・データのダウンロードが３次元プリミティブ・パイプラインのフラッシングを必要としないので、システムの帯域幅および処理能力が向上する。各ピクセルに関するテクスチャ・データは、テクスチャ・マッピング・チップ４６によって、５本のバス２８を経由してフレーム・バッファ基板へ送られる。５本のバス２８は、

フレーム・バッファ基板に備えられる５つのフレーム・バッファ制御器チップ５０Ａ、５０Ｂ、５０Ｃ、５０Ｄおよび５０Ｅにそれぞれ接続され、それらフレーム・バッファ制御器チップに、計算結果のテクスチャ・データが並列的に送られる。フレーム・バッファ制御器チップ５０Ａ－５０Ｅは、それぞれ対応するＶＲＡＭ（ビデオ・ランダム・アクセス・メモリ）チップ５１Ａ－５１Ｅグループに接続される。更にフレーム・バッファ基板は、４つのビデオ形式チップ（５２Ａ、５２Ｂ、５２Ｃおよび５２Ｄ）およびＲＡＭＤＡＣ（ランダム・アクセス・メモリ・デジタル・アナログ変換器）５４を含む。フレーム・バッファ制御器チップは、表示画面の異なる非上重ねセグメントを制御する。各フレーム・バッファ制御器チップは、バス１８経由でフロントエンド基板からプリミティブ・データを、そして、バス２８経由でテクスチャ・マッピングから計算結果のテクスチャ・マッピング・データを受け取る。フレーム・バッファ制御器チップは、プリミティブ・データを補間して、それぞれの対応するセグメントに関する画面表示ピクセル座標、および各ピクセル座標に関して対応するオブジェクトＲ，Ｇ，Ｂカラー値を計算する。テクスチャ・マッピング基板から計算結果のテクスチャ・データが渡される（例えば三角形のような）プリミティブについて、フレーム・バッファ制御器チップは、ピクセル毎にオブジェクト・カラー値と計算結果のテクスチャ・データを結合して、表示画面上で表示されるべき最終的Ｒ，Ｇ，Ｂ値をピクセル毎に生成する。

【００２９】

オブジェクトとテクスチャ・カラー値の結合は、多くの異なる形態で制御することができる。例えば、置き換えモードでは、オブジェクト・カラー値が、単純にテクスチャ・カラー値によって置き換えられ、テクスチャ・カラー値だけがピクセルのレンダリングに使用される。別の形態の調整モードでは、オブジェクトとテクスチャ・カラー値が乗じられピクセルに関する最終的Ｒ，Ｇ，Ｂ値が生成される。更に、対応するテクスチャ・カラー値とオブジェクト・カラー値との組み合わせ方法を定める比率を指定するカラー制御ワードを各テクセルについて記憶することもできる。カラー制御ワードは、各ピクセルに対応するテクセル・データに関して決定され、フレーム・バッファ制御器チップにバス２８経由で渡されるので、制御器チップは対応する制御ワードによって指定された比率を使用して最終的Ｒ，Ｇ，Ｂ値を各ピクセル毎に決定することができる。

【００３０】

フレーム・バッファ制御器チップ５０Ａ－５０Ｅによって生成され、各ピクセルのＲ，Ｇ，Ｂ値を含む画像ビデオ・データが対応するＶＲＡＭチップ５１Ａ－５１Ｅに記憶される。ＶＲＡＭチップ５１Ａ－５１Ｅの各グループは、４０個のＶＲＡＭチップがフレーム・バッファ基板上に配置されるように、８つのＶＲＡＭチップを含む。ビデオ形式チップ５２Ａ－５２Ｄの各々は、異なるセットの１０個のＶＲＡＭチップに接続され、そこからデータを受け取る。ビデオ・データは、ＶＲＡＭチップから順次シフトされ、６４ビット幅バス５８Ａ、５８Ｂ、５８Ｃおよび５８Ｄを経由して４つのビデオ形式チップ５２Ａ、５２Ｂ、５２Ｃおよび５２Ｄへ３３ＭＨＺ伝送率でそれぞれ送られる。ビデオ形式チップは、ＲＡＭＤＡＣが処理できるような形式にビデオ・データを変換して、形式化データを、３２ビット幅バス６０Ａ、６０Ｂ、６０Ｃおよび６０Ｄを経由して３３ＭＨＺ伝送率でＲＡＭＤＡＣ５４へ送る。次に、ＲＡＭＤＡＣ５４は、デジタル・カラー・データをアナログＲ，Ｇ，Ｂカラー制御信号に変換し、各ピクセルに関するＲ，Ｇ，Ｂ制御信号を、Ｒ，Ｇ，Ｂ制御ライン２９を介して表示画面に送る。

【００３１】

本発明の１つの実施形態において、特定のプリミティブ・レンダリング・タスクが複数のプリミティブに関して並列的に実行されるように、テクスチャ・マッピング基板１２およびフレーム・バッファ基板１４に関するハードウェアが反復配置され、これによって、システムの帯域幅が拡大される。そのような代替的実施形態の１例が図３に示されている。図３は、特定のハードウエアが複製されている本発明のコンピュータ・グラフィックス・システムのブロック図である。図３のシステムは、４つの３次元加速器チップ３２Ａ、３２Ｂ、３２Ｃおよび３２Ｄ、キャッシュ・メモリ４８Ａならびに４８Ｂとそれぞれ連動す

る２つのテクスチャ・マッピング・チップ４６Ａならびに４６Ｂ、および、各々が対応するＶＲＡＭチップを持つ１０個のフレーム・バッファ・チップ５０Ａ－５０Ｊを含む。図３のシステムの動作は、図２のシステムのそれに類似している。図３の実施形態におけるハードウェアの反復配置によって、特定のプリミティブ・レンダリング・タスクが複数のプリミティブに関して並列的に実行されるためシステムの帯域幅が増大する。

【００３２】

II．テクスチャ・マッピング・チップの概要

図４は、テクスチャ・マッピング・チップ４６のブロック図である。チップ４６は、オブジェクトおよびテクスチャ・プリミティブ・データをフロントエンド基板から６４ビット幅バス１８経由で受け取るフロントエンド・パイプライン・インターフェース６０を含む。テクスチャ・マッピング・チップ上で処理される三角形プリミティブは最高５２個の３２ビット・デジタル・ワードによって定義されるが、異なる長さのワードによって定義することもできる。パイプライン・インターフェースは、一組のマスター・レジスタと一組の対応するスレーブ・レジスタを含む。レンダリングの間、マスター・レジスタは、プリミティブを定義する５２個のデジタル・ワード・データで逐次満たされる。次に、適切なレンダリング・コマンドを受領すると、データは、パイプライン・インターフェースのスレーブ・レジスタにシフトされ、これによって、マスター・レジスタはパイプライン方式で別のプリミティブを表現するデータで満たされる。バス１８経由で提供されるプリミティブ・データは、ｘ，ｙ，ｚベクトル座標データ、少なくとも１つの三角形頂点に関するＳ，Ｔテクスチャ座標ならびにＲ，Ｇ，Ｂオブジェクト・カラー・データ、および三角形平面方程式を表すデータを含む。上述のように、テクスチャ・マッピング・チップは、オブジェクト・ピクセルｚ座標およびオブジェクト・カラーＲ，Ｇ，Ｂ値を無視し、その他のデータだけをフロントエンド・パイプライン・インターフェース６０に記憶する。

【００３３】

パイプライン・インターフェース６０のスレーブ・レジスタは、バス６２経由でパラメータ補間器回路６４へプリミティブ・データを転送する。パラメータ補間器回路６４は、各プリミティブ三角形を補間して、三角形を表現する各表示画面ピクセル座標について、ピクセルにマップするＳ，Ｔテクスチャ・マップ座標、および、ＳならびにＴ勾配値を決定する。ＳならびにＴ勾配は、それぞれ、隣接するピクセルの間でのＳなびにＴ座標の変化に等しく、以下に説明される方法で計算される。

【００３４】

パラメータ補間回路６４は、図５を用いて詳細は後述するが、辺ステッパ(edge steper)６６、ＦＩＦＯ（先入れ先出し）バッファ６８、スパン・ステッパ(span stepper)７０、勾配ならびに釣り合い補正回路７２を全て逐次接続形態で含む。辺ステッパは、三角形頂点の１つのｘ，ｙピクセル座標で開始し、三角形平面方程式を利用して、三角形の辺を辿って、三角形の辺を定義するピクセル座標を決定する。各ピクセル座標について、テクスチャ・マップにおけるどのテクセルが各表示画面ピクセル座標に対応するかを識別するように、三角形頂点のＳ，Ｔ値に基づいてテクスチャ・マップのＳならびにＴ座標が決定される。ピクセルおよびテクセル座標は、一時的にＦＩＦＯバッファに記憶され、次にスパン・ステッパに渡される。三角形の辺に沿った各ｘ，ｙピクセル位置毎に、スパン・ステッパは三角形の対応する（ｘ、ｙ間の）スパンに沿って進み、該スパンに沿ったピクセル位置のそれぞれについてＳ，Ｔテクセル座標を決定する。

【００３５】

もしも、ピクセルが、テクスチャに関する一連のＭＩＰマップの中の１つにおける単一のテクセルと１対１の対応関係を持たない場合、表示画面ピクセルに対するＳならびにＴ座標の各々は、整数部分と小数部分を持つであろう。上述のように、テクスチャ・マップと対応付けされる時、各表示画面ピクセルが、テクスチャに関する一連のＭＩＰマップの１つにおける複数のテクセルの間に落ちることがあり、更に、一連のＭＩＰマップのサイズ的に隣接するＭＩＰマップの間に落ちることもある。

【００３６】

勾配および釣り合い補正回路７２は、各表示画面ピクセルに対するＳおよびＴの勾配値（ΔＳおよびΔＴ）を決定する。本発明の１つの実施形態において、勾配ΔＳは、勾配ΔＳｘと勾配ΔＳｙのいずれか大なる方であるように選択される。ここで、勾配ΔＳｘは、表示画面上の隣接ピクセルの間でｘ座標が変化するにつれて変わるテクスチャ・マップにおけるＳ座標の変化であり、勾配ΔＳｙは、表示画面上の隣接ピクセルの間でｙ座標が変化するにつれて変わるテクスチャ・マップにおけるＳ座標の変化である。勾配ΔＴも同様に計算される。１つの表示画面ピクセルに関するΔＳおよびΔＴは、表示画面上のピクセルの対応するＳ，Ｔ軸での変化に対するテクスチャ・マップ内の座標位置の変化率を示し、ピクセルに対するテクスチャ・データを作成するため、どのＭＩＰマップがアクセスされなければならないかを決定するために使用される。例えば、表示画面ピクセルについて２に等しい勾配は、ピクセルが４つの（すなわち、後述されるように2^2の)テクセルにマップすることを示して、当該ピクセルに関するテクスチャ・データを提供できるように基本マップからサイズ的に２だけ減じられたＭＩＰマップ（例えば図１のマップ１０２）をアクセスしなければならない。かくして、勾配が増加するにつれて、ピクセルに対するテクスチャ・データを提供するためアクセスされるＭＩＰマップのサイズは減少する。

【００３７】

本発明の１つの実施形態において、各ピクセルに対する適切なＭＩＰマップを選択するため、勾配が当該ピクセルに関するΔＳｘ、ΔＳｙ、ΔＴｘおよびΔＴｙの最大値に等しくなるように、ΔＳおよびΔＴの大なる方に等しい単一の傾斜が使用される。しかし、勾配は、例えば上記の値の最小値、それら平均値あるいはその他の組合せを選択することによって、異なる形態で代替的に選択することも可能であることは理解されるべきであろう。Ｓ，Ｔ座標の１つだけの変化率を示す単一の勾配が選択されるので、その勾配の平方値は、対応するピクセルにマップするテクセルの数を表す。

【００３８】

勾配を使用して、パラメータ補間回路はピクセルが対応する最も近いマップと、ピクセルがそのマップに直接対応するものからどれほど離れているかを示す値を決定する。最も近いマップは、マップ番号の整数部分によって識別され、ピクセルがそのマップに直接対応するものからどれほど離れているかを示す値は、マップ番号の小数部によって識別される。

【００３９】

再び図４のテクスチャ・マッピング・チップのブロック図を参照して説明すれば、パラメータ補間回路６４からのテクセル・データ出力が、ライン７０経由でタイル作成器／境界検査器(tiler and boundary checker)７２に送られ、そこで、テクセル・データによって指定されるテクスチャ・マップの各々の位置に最も近い４つのテクセルのアドレスが決定され、それらテクセルの各々がテクスチャ境界の内部にあるか否かが検査される。テクセル・データは、補間されたＳ、Ｔマップ座標（整数値と小数値）、およびマップ番号ならびにマップ小数を含む。タイル作成器は、ＳおよびＴ座標の整数部分がパラメータ補間回路６によって計算された整数を使用し、各々の整数部に１を加えて４つの最も近いテクセルのアドレスを生成する。次に、境界検査器が、それら４つのテクセルのいずれかのＳ，Ｔ座標がテクスチャ・マップの境界の外側に落ちるか否かを判断する。もしもある表示画面ピクセルが、テクスチャ・マップの境界の外側に落ちるＳ，Ｔ座標位置に対応する場合、いくつかのマッピング・テクスチャ方式の１つによって、そのピクセルについてテクスチャ・データを生成すべきか、またそのデータをどのように生成すべきかが決定される。そのような方式の例には、ラッピング（wrapping、すなわちテクスチャの繰り返し）、ミラーリング（mirroring、すなわちテクスチャの鏡画像の繰り返し）、境界の外側にあるテクスチャ・マッピングの取り消し、および、境界外での同一調カラー表示などが含まれる。

【００４０】

境界を越えたテクスチャ・マップ位置にピクセルをマップすることを可能にすることによって、テクスチャをオブジェクト・プリミティブにマップする方法に柔軟性が与えられる

。例えば、テクスチャがオブジェクトの複数部分にマップされるように、反復動作でテクスチャをオブジェクトにマップすることが望ましい場合がある。例えば、[0，0]から(10，10)までの範囲のＳ，Ｔ座標を持つテクスチャが定義される場合、ユーザは、そのような範囲のＳ，Ｔ座標へマップするようにオブジェクトの特定部分を指定することができる。上記において、記号[ ]は、指定する範囲が括弧内座標を含み、記号（ ）は括弧内座標を含まないことを表し、以下においても同様の表記法を使用する。ラッピング機構がテクスチャの境界の外側に落ちるＳ，Ｔ座標について動作するように選択される場合、[10，10]から(20，20)までのＳ，Ｔ座標を持つピクセルは、[10，10]から(20,20)までのＳ，Ｔ座標にあるテクセルにそれぞれ対応するであろう。

【００４１】

上述のように、１つのピクセルに関して２次元テクスチャ・マップから得られるテクスチャ・データは、８つのテクセル、すなわち最も近い２つのＭＩＰマップにおける最も近い４つのテクセルが結合された結果である。８つのテクセルを結合してテクセル・データを生成する多数の方法がある。例えば最も近いマップにおける最も近い単一のテクセルを選択することによって、平均算出の必要性をなくすことができる。別の方法として、最も近い２つのマップそれぞれの最も近い単一のテクセルが、勾配値に基づいて平均される。このような方法は、８つの最も近いテクセルの平均値が計算される場合のように正確にテクチャをマップしない。

【００４２】

本発明の１つの実施形態においては、単一ピクセルに関するテクスチャ・データを８つのテクセルの加重平均として計算する３線形補間法(trilinear interpolation)が利用される。テクスチャ・データをアクセスする最も近い２つのＭＩＰマップを識別するためＳ，Ｔの変化率を表す勾配が使用され、各々のマップ内の最も近い４つのテクセルがアクセスされる。表示画面ピクセルがマップするＭＩＰマップの位置のＳ，Ｔ座標に最も近いテクセルに基づいて、各マップ内の４つのテクセルの平均が加重される。該ピクセルに関するＳ、Ｔ座標の小数部分が、この加重を実行するために使用される。次に、最も近い２つのＭＩＰマップ各々の上記平均値が、勾配値に基づいて加重される。この加重プロセスにおける使用のため、勾配を基に小数値が計算される。例えば、値３の勾配は、勾配２および勾配４にそれぞれ対応するＭＩＰマップの中間にある。

【００４３】

テクセル補間プロセスは、テクセル補間回路７６によって実行される。各表示画面ピクセルに関するＳおよびＴ座標の小数部分は、タイル作成／境界検査器を経由して、パラメータ補間回路からテクセル補間回路７６へライン７４を介して送られる。テクセル補間回路は小数部分を使用して、複数のテクセルの各々に与えられる加重を決定し、所望のテクセル・データを計算する。

【００４４】

上述のように、レンダリングされるプリミティブに関連するテクスチャＭＩＰマップは、ローカル・キャッシュ・メモリ４８（図２）に記憶される。本発明の１つの実施形態において、キャッシュは完全連想型である。キャッシュは、各インターリーブに２つのＳＤＲＡＭチップが配置される構成で、全体として４つのインターリーブに区分けされる８つのＳＤＲＡＭチップを含む。各インターリーブ内のＳＤＲＡＭチップが同時にアクセスされるように各インターリーブに対応して１つ宛計４つのコントローラが備えられる。各ＳＤＲＡＭチップは２つのメモリ・バンクを含む。上記メモリ・バンクにおいては、従来技術のＤＲＡＭの場合に起きるような２つの異なるページ（すなわち２つの異なる行アドレス）からデータを取り出すことに一般に関連する再ページングの負荷を伴うことなく、メモリの異なるページを連続的読取りサイクルでアクセスすることができる。

【００４５】

テクスチャ・データ（すなわち、ＭＩＰマップ）は、各々が２５６×２５６のテクセルを含むテクセル・データ・ブロックに分割される。キャッシュ・メモリは、一時点で６４個のデータ・ブロックを記憶することができる。各ブロックは、ブロックをユニークに識別

するブロック・タグを持つ。キャッシュは、キャッシュに現在記憶されているデータ・ブロックに対応するブロック・タグを記憶するディレクトリ７８を含む。詳細は後述するが、ブロック・タグの各々は、データ・ブロックが表現する特定のテクスチャを識別するテクスチャ識別子（すなわちテクスチャＩＤ）、当該テクスチャの一連のマップの中からデータ・ブロックが表す特定のＭＩＰマップを識別するマップ番号、および、該特定マップ内の上記データ・ブロックの位置を識別する高位ＳなびにＴ座標を含む。キャッシュ・ディレクトリ内のブロック・タグの物理的位置が、キャッシュ・メモリ内における対応するデータ・ブロックの位置を表す。

【００４６】

異なるテクスチャを区別するテクスチャ識別子を用いて、複数のテクスチャのＭＩＰマップをキャッシュ・メモリに同時に記憶することもできる。一部のＭＩＰマップが２５６×２５６未満のテクセルを含むこともあり、この場合データ・ブロックの一部は使用されない。例えば、一連のＭＩＰマップの小さい方のマップ、または小さいテクスチャの場合大きい方のマップでも、２５６×２５６個のテクセルを越えないことがある。メモリ空間を有効に活用するため、各マップ部分がブロック内のサブブロックに割り当てられるように、複数のマップ部分がテクスチャ・データの１つのブロック内に記憶されるようにすることもできる。１つのブロック内に記憶される複数のマップの各々は、ブロック内のマップの位置を識別するサブテクスチャ識別子（ＩＤ）を持つ。

【００４７】

レンダリングの間、タイル作成／境界検査器７２は、レンダリングされるピクセルに対応するテクスチャ・データ・ブロックに関する読取りキャッシュ・タグを生成する。タグを生成する方法の詳細は後述する。タグは、テクスチャ・データのテクスチャＩＤを表す８ビット、テクスチャ・データのマップ番号を決定する際に使用される１ビット、および、テクスチャ・データの高位７ビットのＳならびにＴ座標を含む２３ビットのフィールドである。キャッシュ・ディレクトリ７８は、タイル作成／境界検査器から送られる読み取りキャッシュ・タグをディレクトリに記憶されているブロック・タグと比較して、レンダリングの際に使用されるべきテクスチャ・データ・ブロックがキャッシュ・メモリに存在するか否かを判断する。レンダリングされるべきプリミティブに対応するテクスチャ・データ・ブロックがキャッシュ・メモリに記憶されている場合（すなわちキャッシュ・ヒットの場合）、キャッシュ・ディレクトリは、ヒットしたタグに対応するキャッシュ内のテクスチャ・データ・ブロックの物理的位置を標示するブロック・インデックスを生成する。ブロック・インデックスの計算の詳細は後述する。キャッシュから読み取られるべき各テクセルについて、ブロック内のテクセルの位置を標示するテクセル・アドレスがまたタイル作成／境界検査器７２によって生成される。テクセル・アドレスは、より大きいサイズ・マップに関する補間されたＳ，Ｔ座標の低位アドレス・ビットを含み、より小さいサイズのマップに関して以下に記述されるアルゴリズムに基づいて計算される。ブロック・インデックスおよびテクセル・アドレスはともに、キャッシュ内のテクセルの位置を示すキャッシュ・アドレスを含む。詳細は後述するが、４つのインターリーブのどこにテクセルが記憶されているかを決定するため、各テクセルに関するＳならびにＴ座標のＬＳＢ（すなわち最下位ビット）がデコードされ、キャッシュ・アドレスの残りのビットは、コマンドと共に、ライン８４経由でテクセル・キャッシュ・アクセス回路８２へ送られ、キャッシュ内の上記アドレス位置に記憶されているテクセル・データが読み取られる。

【００４８】

読取りキャッシュ・タグがキャッシュ・ディレクトリに記憶されてるブロック・タグのいずれとも一致しない場合、すなわちキャッシュ・ミスが発生する場合、キャッシュ・ディレクトリ７８は、ライン９４（図２）経由でフロントエンド基板上に対して割込み制御信号を生成し、これに応答して、分配器チップ３０がライン９５経由でホスト・コンピュータ１５に対する割り込みを生成する。割り込みに応答して、ホスト・コンピュータのプロセッサ１９が、サービス・ルーチン（詳細は後述）を実行することによって、キャッシュ・ミスのあったブロック・タグをキャッシュ・ディレクトリから読み取り、フロントエン

ド基板１０およびテクスチャ・マッピング・チップ４６における３次元プリミティブ・パイプラインをバイパスする形態で、テクスチャ・データの対応するブロックをキャッシュ・メモリにダウンロードする。主メモリからダウンロードされたテクスチャ・データは、バス２４経由で（図４の）テクセル・ポート９２を通ってテクセル・キャッシュ・アクセス回路８２へ送られ、キャッシュ・メモリを形成するＳＤＲＡＭへ書き込まれる。

【００４９】

キャッシュ・ミスが発生する時、テクスチャ・マッピング・チップは、ミスが発生したプリミティブの処理を進める前に、新しいテクスチャ・データがダウンロードされるのを待つ。しかしながら、キャッシュ読み取りに続くパイプラインの処理段階は、ミスのあったプリミティブに先行して受け取ったプリミティブを処理し続ける。同様に、キャッシュ読み取りに先行するパイプラインの処理段階は、新しいテクスチャ・データのダウンロードを待っている間、パイプラインがいっぱいにならない限り、キャッシュ読み取り動作の背後でプリミティブの処理を続行する。

【００５０】

レンダリングの間、フレーム・バッファ基板１４におけるパイプラインの後の方の処理段階は、対応するテクスチャ・データがテクスチャ・マッピング基板から受け取られるまで、プリミティブの処理を進めない。従って、キャッシュ・ミスが発生して、テクスチャ・マッピング・チップが新しいテクスチャ・データのダウンロードを待つ時、フレーム・バッファ基板１４は、同様に、テクスチャ・マッピング・チップから送られてくるテクスチャ・データを待つ。テクスチャ・マッピング・チップの場合と同様に、テクスチャ・マッピング・データの受け取り段階に続くパイプラインの処理段階は、キャッシュ・ミスのあったプリミティブに先立って受け取ったプリミティブの処理を続行し、テクスチャ・マッピング・データを受け取る段階に先行するパイプラインの処理段階はパイプラインがいっぱいにならない限りプリミティブの処理を続行する。

【００５１】

キャッシュ・ミスに応答して新しいテクスチャ・データを待つ時テクスチャ・マッピング基板またはフレーム・バッファ基板いずれかのパイプラインが待機する場合、フロントエンド基板１０のパイプラインもまた同様に待機するする点は理解されるべきであろう。キャッシュ・ミスの発生によって、ホスト・コンピュータの主メモリへのアクセスおよびテクスチャ・データのダウンロードを完了するにはいくつかのサイクルがかかるので、フレーム・バッファ基板のパイプラインが待機させられたことによってテクスチャ・マッピング・チップのパイプラインが待機する必要がないことを確認することが望ましい。従って、本発明の１つの実施形態においては、フレーム・バッファ基板が、テクスチャ・マッピング基板より深いプリミティブ・パイプラインを備えるように構成され、それにより、フレーム・バッファ・パイプラインが使用可能になるのを待つことによるテクスチャ・マッピング・パイプラインの遅延がなくなる。

【００５２】

本発明の１つの実施形態では、上記の機能を備えさせるため、テクスチャ・マッピング機能がオフにされる。これは、ホスト・コンピュータのプロセッサ１９上でソフトウェアを操作して、テクスチャ・マッピング基板１２およびフレーム・バッファ基板におけるレジスタを設定することによって達成される。テクスチャ・マッピングがオフに設定される時、これらのレジスタはそれぞれ、テクスチャ・マッピング・チップ４６がフレーム・バッファ基板１４へテクスチャ・データを送ることを禁止し、テクスチャ・マッピング基板からのテクスチャ・データを待つことなくプリミティブに対するレンダリングを続けるようにフレーム・バッファ基板に命令する。

【００５３】

上述のように、２次元テクスチャ・マップからのテクスチャ・データでレンダリングされる表示画面ピクセルの各々について、（双線形補間の場合）１つのＭＩＰマップから４つのテクセル、または（３線形補間の場合）２つの隣接ＭＩＰマップから８つのテクセルが、キャッシュ・メモリから取り出され、該ピクセルに対するテクスチャ・データが決定さ

れる。キャッシュから読まれたテクセルは（図３の）バス８６経由でテクセル補間回路７６へ送られ、そこで、複数テクセルの補間によって、各ピクセルのテクセル・データが計算される。補間方法は、システムに関して設定されるモードに応じて変り得る。１点標本抽出補間モードが設定される場合、結果として生成されるテクセル・データは、テクスチャ・マップにおけるピクセルのＳ，Ｔ座標によって定義される位置に最も近い１つのテクセルに等しい。別の方法として、双線形補間または３線形補間が用いられる場合、それぞれ１つまたは最も近い２つのマップにおける４または８個の最も近いテクセルの加重平均である。複数のテクセルの各々に与えられる加重は、タイル作成／境界検査器からテクセル補間回路７６へ提供される勾配値およびＳならびにＴ座標の小数部分に基づいて決定される。

【００５４】

表示画面ピクセルに関する計算結果のテクセル・データは、バス８８経由でフレーム・バッファ・インターフェースＦＩＦＯバッファ９０へ順次送られる。フレーム・バッファ・インターフェースＦＩＦＯバッファ９０は、最高６４までの計算結果のテクセルを記憶することができる。

【００５５】

計算結果のテクセルの各々は、Ｒ，Ｇ，Ｂを表現する各８ビット、およびαを表す８ビットを含む３２ビット・ワードである。αバイトは、（図２の）フレーム・バッファ基板１４に対して、テクセルに対応するピクセルについて最終的表示画面Ｒ，Ｇ，Ｂ値を計算する際に、計算結果のテクスチャ・データのＲ，Ｇ，Ｂ値をフレーム・バッファ基板によって生成されたオブジェクト・データのＲ，Ｇ，Ｂ値と結合する方法を標示する。フレーム・バッファ・インターフェースＦＩＦＯバッファ出力Ｔ０－Ｔ４は、（図２の）バス２８を経由してフレーム・バッファ基板１４へ送られる。フレーム・バッファ基板は、各画面表示ピクセルについて最終的Ｒ，Ｇ，Ｂ値を生成するためαによって指定された方法で、計算結果のテクセル・データのＲ，Ｇ，Ｂ値をオブジェクトＲ，Ｇ，値と結合する。

【００５６】

III．キャッシュ・メモリの構成

図６は、本発明の１つの実施形態に従うキャッシュ・メモリのブロック図である。該キャッシュ・メモリは、テクセル・ポート９２、テクスチャ補間回路７６、キャッシュ・ディレクトリ７８およびテクセル・キャッシュ・アクセス回路８２を含むテクスチャ・マッピング・チップ部分に接続する。この実施形態では、キャッシュ・メモリ４８は、４つのインターリーブ２０４Ａ、２０４Ｂ、２０４Ｃおよび２０４Ｄを含む。各インターリーブは、同時にアクセスされることができる２つのＳＤＲＡＭチップ（図示されてない）を含む。各ＳＤＲＡＭは１回の読み取りサイクルの間に８ビットのデータを提供する。従って、各インターリーブは、１読取りサイクルの間に１６ビットのテクセル・データを提供する。インターリーブの各ＳＤＲＡＭの２つの連続する位置のそれぞれに８ビットが記憶される形態で、１つのインターリーブのキャッシュに各々３２ビット・ワードのテクセル・データが記憶される。従って、キャッシュから１つのテクセルを読み取るためには、該当するインターリーブの連続的位置に対する２回の読取りサイクルの実行によって３２ビットのテクセル・データが取り出される。後述されるように、２回の連続サイクルでバース・データ(burst data)を作成するには、（行および桁データを含む）１つのアドレス・ワードだけを各インターリーブ内のＳＤＲＡＭに送出すればよい。バースト・データは、与えられたアドレスから第１のサイクルで渡される１６ビット、同じ行を持つアドレスから第２のサイクルで渡される１６ビット、および１だけ増分される桁を含む。

【００５７】

テクセル・キャッシュ・アクセス回路８２は、コントローラＡ（２００Ａ）、コントローラＢ（２００Ｂ）、コントローラＣ（２００Ｃ）およびコントローラＤ（２００Ｄ）と名付けられた４つの独立したコントローラを含む。４つのコントローラＡ、Ｂ、ＣおよびＤは、並列バス２０２Ａ、２０２Ｂ、２０２Ｃおよび２０２Ｄを経由して４つのインターリーブ２０４Ａ、２０４Ｂ、２０４Ｃおよび２０４Ｄのデータに同時にアクセスすることが

できる。上記コントローラは、バス８４Ａ、８４Ｂ、８４Ｃおよび８４Ｄ経由でそれぞれ受け取ったコマンドおよびアドレスに応答してキャッシュ・メモリ４８からテクセル・データを読み取る。

【００５８】

上述のように、各ピクセルは、潜在的に、１つのＭＩＰマップの４つのテクセルに対応するか、あるいは複数のＭＩＰマップの８つのテクセルに対応する。詳細は後述するが、キャッシュにダウンロードされるテクセル・データは、ホスト・コンピュータの主メモリにおいて、各ＭＩＰマップにおけるいかなる４つの隣接するテクセルも並列的アクセスが可能なように別々のインターリーブに位置づけられるように配置される。従って、双線形補間法によってテクセル・データを生成するために必要とされるＭＩＰマップにおけるいかなる４つの隣接するテクセルも、１回の読取り動作で読み取ることができる。３線形補間法が使用される場合は、１組４つからなる２組の（計８つの）テクセルが隣接するＭＩＰマップから２回の読み取り動作で読み取られる。

【００５９】

図７は、キャッシュ・メモリの４つのインターリーブ構成によってある１つのＭＩＰマップにおける任意の４つの隣接テクセルを同時に読み取ることができる利点を活かすようにテクスチャ・データ・ブロックが配置される形態の１例を示している。各テクセルには、該テクセルが記憶されているキャッシュ・メモリのインターリーブを識別するラベルＡ、Ｂ、ＣおよびＤが付けられている。マップ内のいかなる位置もＡ、Ｂ、ＣおよびＤのラベルを持つ４つのテクセルの間に落ちるように、ＡないしＤのラベルのパターンが繰り返されている。このようにして、あるマップの内の任意の位置に対応するピクセルについて、最も近い４つのテクセルは、別々のインターリーブＡないしＤに存在するため、それらテクセルは４つの独立コントローラ２００Ａ－２００Ｄによって同時にアクセスされることができる。例えば、ピクセルＰ０はＡ、Ｂ、ＣおよびＤというラベルの４つのテクセルの間の位置に対応し、ピクセルＰ１はＢ、Ａ、ＤおよびＣというラベルの４つのテクセルの間の位置に対応する。

【００６０】

上述のキャッシュ構成は例示の目的で記述したものであって他の代替構成も実施できることは理解されるべきであろう。例えば、キャッシュを、８つの別々のコントローラを持つ８つの別々のインターリーブの形態で実施して、３線形補間法が使われ時、８つのテクセルが、１回の読み取り動作で同時にアクセスされることができるように構成することもできる。

【００６１】

キャッシュ・メモリ内のＳＤＲＡＭチップの各々は、同時に別々の活動ページ（すなわち、共通の行アドレスを持つメモリ位置グループ）を維持することができる２つの等しいサイズのバンクに内部的に分割される。このようにして、従来技術のＤＲＡＭの場合に起きるような２つの異なるページ（すなわち２つの異なる行アドレス）からデータを取り出すことに一般に関連する再ページングの負荷を伴うことなく、ＳＤＲＡＭチップの２つのバンク内の異なるページにあるデータを連続的読取りサイクルでアクセスすることができる。

【００６２】

更に詳細は後述するが、３線形補間法を使用する際のページ間アクセス負荷を最小限にとどめるこのＳＤＲＡＭ構成の利点を活かすようにテクスチャ・データがキャッシュ・メモリ内に配置される。３線形補間のために必要な８つのテクセルは、２つのＭＩＰマップに収納されている１組４テクセルからなる２組のテクセルを含む。１つのＭＡＰにある１組の４つの隣接するテクセルの各々は、上述のように、同時にアクセスできるようにインターリーブＡ、Ｂ、ＣおよびＤにそれぞれ記憶されている。更に、任意のテクスチャについて一連のＭＩＰマップにおける隣接するＭＩＰマップ上で共通するデータが、キャッシュの異なるＳＤＲＡＭバンクに記憶される。３線形補間が実行される時、第１のバーストの２回の読取りサイクルの間に、１つのＭＩＰマップの４つのテクセルがインターリーブＡ

ないしＤのＳＤＲＡＭバンクの１つから同時に読み取られ、後続のバーストの２回の読取りサイクルの間に、隣接するＭＩＰマップの４つのテクセルが別のＳＤＲＡＭバンクから読み取られる。ＳＤＲＡＭの両方のバンクは同時に行が有効であるので、再ページングの負荷なしに２組のテクセルは連続的バースト・モードで読み取られる。オブジェクトの複数ピクセルがレンダリングされる場合、隣接ピクセルは、該テクスチャに関する同じ２つのＭＩＰマップに対応していることが多いため、キャッシュへの読取りが、２つのマップに共通データを記憶するキャッシュ・ブロックの間で連続的に切り換えを行うことが必要とされる点は理解されるべきであろう。表示画面ピクセルをレンダリングする間２つの隣接ＭＩＰマップの間での切り換えを行う時、各サイクルの再ページングの負荷を伴うことなく３線形補間を実行できるので、２つのページが各ＳＤＲＡＭ内で活動的であることを可能にする本発明のキャッシュ構成は利点がある。

【００６３】

図８は、本発明のキャッシュ・メモリの上述の実施形態のさらに詳細なブロック図である。キャッシュは、各々が２つのＳＤＲＡＭチップを含む４つのインターリーブ２０４Ａ－２０４Ｄに均一に分割されたＳＤ１－ＳＤ８というラベルの８つのＳＤＲＡＭチップを含む。各インターリーブの２つのＳＤＲＡＭは、以下のような共通のラインを共有する。すなわち、１１本のアドレス線（ＡＤＤ）、行・桁ストローブ（ＲＡＳなびにＣＡＳ）、書き込みイネーブル（ＷＥ）、クロック・イネーブル（ＣＫＥ）、およびデータ入出力マスク（ＤＱＭ）である。各インターリーブ内のＳＤＲＡＭは、各読取りまたは書込みサイクルの間８ビットデータがそれぞれ読み書きされる８本の独立データ線に接続される。各ＳＤＲＡＭチップは、各々がテクスチャ・データの１,０４８,５７６個の８ビット・ワードを記憶する２つのメモリ・バンクを含む。

【００６４】

各インターリーブの２つのＳＤＲＡＭは同時にアクセスされ、一方のＳＤＲＡＭがデータ・ビット[１５：０８］を、他方がデータ・ビット［０７：００］をそれぞれ提供することによって両方で１６ビットのデータを提供する。上述のように、１回のバースト・モードの２つの連続的読取りサイクルが、各インターリーブから３２ビット・テクセル・データを読み取る。その個々の８ビット・ワードは、当該テクセルＲ，Ｇ，Ｂおよびα値の各々を表す。

【００６５】

ＳＤＲＡＭチップが、１１本のアドレス線ＡＤＤ上で多重送信された２０のアドレス・ビットを受け取り、各バンク内の１,０４８,５７６個の８ビット・ワードをデコードする。詳細は後述するが、キャッシュからアクセスされるべき各テクセルについて６ビットのブロック・インデックスおよび１６ビットのテクセル・アドレスが計算される。ブロック・インデックスは、６４個のデータ・ブロックのどこにテクセルが位置しているか標示し、テクセル・アドレスは、ブロック内のテクセルの正確なＳ，Ｔ座標アドレスを標示する。１平方データ・ブロックが２５６×２５６テクセルを含むと仮定すれば、８つのＳビットおよび８つのＴビットがテクセル・アドレスを構成する。キャッシュ・アドレスは、ブロック・インデックス（ＭＳＢ６ビット）およびテクセル・アドレス（ＬＳＢ１６ビット）の組合せを含む２２ビット・ワードである。キャッシュ・アドレスは、キャッシュ内の正確なテクセル位置を示す。

【００６６】

レンダリングの間、タイル作成／境界検査器が、テクセル・アドレスの下位Ｓビットおよび下位Ｔビット（すなわち、ＬＳＢ Ｓ座標おｙびＬＳＢ Ｔ座標）をデコードして、テクセルがキャッシュの４つのインターリーブのどれに記憶されているかを決定する。キャッシュ・アドレスの残りの２０アドレス・ビットは、アドレス線ＡＤＤに沿って、該当するインターリーブ内の２つのＳＤＲＡＭチップに対して提供される。２つのＳＤＲＡＭチップに対して提供される２０アドレス・ビットのうち、９ビットは、テクセル・データにアクセスするためＳＤＲＡＭ内の桁を選択するために使用され、１１ビットは行を選択するために使用される。当業者によって理解されるように、桁および行アドレス・ビットは、

異なるサイクルでＳＤＲＡＭにラッチされ、ＲＡＳおよびＣＡＳストローブは、従来技術の方法でデータにアクセスするために使用される。

【００６７】

２サイクルのバースト・モードの間に、第１のサイクルの間に同じインターリーブ内の２つのＳＤＲＡＭのアドレス指定された位置から１６ビットが読み取られ、次に、別のアドレスを用意することなく、第２のサイクルで、２つのＳＤＲＡＭのもう１つ別の位置から１６ビットが読み取られる。第２のサイクル中のアドレスは、同じ行アドレスおよび１増分された桁アドレスを含む。一旦１つのページ（すなわち特定の行）のアドレスが起動されれば、異なる行アドレスが与えられるまでその行は活動的であるという点は理解されるべきであろう。従って、同じインターリーブからアクセスされるべき連続的テクセルが（同じ行アドレスを含む）同じページにあるならば、行アドレスは、連続バーストの最初に１度だけ提供されればよい。

【００６８】

加えて、ＲＡＳ、ＣＡＳおよびＷＥラインは、従来技術の方法でアドレス指定しＳＤＲＡＭチップへデータを書き込むため使用される。クロック・イネーブル信号ＣＫＥがオフにされると、内部クロックは中断される。ＳＤＲＡＭは、この信号に応答してデータを処理しない状態に保ち、両方のバンクをアイドル状態にする。データ入出力マスクＤＱＭ信号は、読取りサイクルの間出力イネーブルとして機能し、書き込みサイクルの間入力データ・マスクとして機能する。

【００６９】

従来技術におけるＳＤＲＡＭの使用方法では、ＳＤＲＡＭは、現在ページから現在のデータをアクセスしている間に後続のデータをどのページからアクセスするかを決定して現在データ読み取りサイクルが完了する前にその将来のページを起動させる。ＳＤＲＡＭが、２つの異なるページをイネーブルして同時に活動的にさせるので、上記従来技術のＳＤＲＡＭの使用は、従来技術のＤＲＡＭの使用におけるようなデータを異なるページからアクセスする場合に派生する再ページングの負荷を回避する。しかしながら、多数の連続読取りサイクルで読み取られるべきデータが異なるページに位置する場合、将来ページを前もって調べ起動するために複数のサイクルが必要とされるため、従来技術のＳＤＲＡＭ使用法は上記の利点を提供しない。本発明のテクスチャ・データ記憶方法は、再ページングなしで異なるページからの複数の連続的ＳＤＲＡＭ読取りサイクルを実行可能とさせることによって、従来技術のＳＤＲＡＭ使用に比較して利点を持つ。特に、（３線形補間を実行する場合連続的読取りサイクルの間のアクセスを必要とする）テクスチャの隣接ＭＩＰの共通データをＳＤＲＡＭの別々のバンクに記憶することによって、別々のバンクからのデータが、連続的な読取りサイクルで再ページングの負荷なしにアクセスされることができる。ＳＤＲＡＭ処理性能を向上させるための本発明のデータ記憶配置方法をテクスチャ・マッピング・データに関して以上説明したが、本発明の方法がそのようなテクスチャ・マッピング・データに関するものに限定されない点は理解されるべきであろう。特に、複数の連続的読取りサイクルが異なるメモリ位置からデータをアクセスするようなタイプのデータのすべてを割り当てる場合に本発明の方法は応用できる。

【００７０】

Ⅳ．キャッシュ制御ＦＩＦＯ

図９は、境界検査器７２、キャッシュ・ディレクトリ７８、キャッシュ・アクセス回路８２、キャッシュ・メモリ４８およびテクセル補間回路７６を含むテクスチャ・マッピング・チップの一層詳細なブロック図である。テクセル・キャッシュ・アクセス回路８２は、４つのキャッシュ・アクセス・コマンドＦＩＦＯ２０６Ａ、２０６Ｂ、２０６Ｃおよび２０６Ｄを含む。キャッシュ・アクセス・コマンドＦＩＦＯ２０６Ａ－２０６Ｄは、１６ビット・バス８４Ａ、８４Ｂ、８４Ｃおよび８４Ｄを経由して境界検査器からそれぞれ受け取るキャッシュ・アクセス・コマンドを記憶する。キャッシュ・アクセス・コマンドＦＩＦＯ２０６Ａ－２０６Ｄは、図６に示されるコントローラ２００Ａ－２００Ｄにそれぞれ対応する。例えば、ＦＩＦＯ２０６Ａコマンドは、インターリーブ２０４Ａ内のＳＤＲＡ

Mのキャッシュ・アクセスを起動する。この実施形態においては、各キャッシュ・アクセス・コマンドＦＩＦＯは、８つの１６ビット・コマンドを一時的に記憶する能力を持つ。このように、システムのパイプライン性能を向上させるため、キャッシュ・アクセス回路が働く前に、８つのコマンドがキャッシュ・アクセス・コマンドＦＩＦＯの各々に記憶される。

【００７１】

上述のように、レンダリングの間、境界検査器７２は、対象のピクセルに対応するテクスチャ・データのブロックの各々に関する読取りキャッシュ・タグをキャッシュ・ディレクトリ７８に記憶されているブロック・タグの各々と比較して、テクセルがキャッシュにあるかどうかを判定する。ヒットが発生すれば、キャッシュ内のテクスチャ・データの対応するブロックの位置を表すブロック・インデックスが生成される。タイル作成／境界検査器は、補間Ｓ，Ｔ座標、テクスチャＩＤ、特定テクセルのサブテクスチャＩＤ、テクセルをアクセスすべきマップのマップ番号およびテクスチャの基本マップのサイズを用いてテクセル・アドレスを決定するルーチンを同時に実行する。詳細は後述する。（キャッシュ・アドレスを構成する）ブロック・インデックスおよびテクセル・アドレスを用いて、タイル作成／境界検査器がテクセルが記憶されているキャッシュの特定のインターリーブおよびそのインターリーブのＳＤＲＡＭチップの行および桁アドレス・ビットを決定する。アドレス情報は、キャッシュ読み取りコマンドとともに、対応するキャッシュ・アクセス・コマンドＦＩＦＯに送られる。

【００７２】

テクセル補間回路７６は、８つのテクセル・データＦＩＦＯ２１４Ａ０、２１４Ａ１、２１４Ｂ０，２１４Ｂ１，２１４Ｃ０，２１４Ｃ１，２１４Ｄ０および２ｌ４Ｄ１を含む。テクセル・データＦｌＦＯ２ｌ４Ａ０ならびに２ｌ４Ａ１は、キャッシュ・メモリのインターリーブ２０４Ａに対応し、ＦｌＦＯ２ｌ４Ｂ０ならびに２ｌ４Ｂ１は、インターリーブ２０４Ｂに対応し、ＦｌＦＯ２ｌ４Ｃ０ならびに２ｌ４Ｃ１は、インターリーブ２０４Ｃに対応し、ＦｌＦＯ２ｌ４Ｄ０ならびに２ｌ４Ｄ１は、インターリーブ２０４Ｄに対応する。

【００７３】

先に述べたように、キャッシュ・メモリの４つのインターリーブの各々は、別々のキャッシュ・アクセス経路を通して同時にアクセスされることができる。レンダリングの間、テクセル・キャッシュ・アクセス回路８２がキャッシュ・メモリ４８からテクセル・データをアクセスする時、テクセル・アクセス制御ワードが、バス２０８Ａ、２０８Ｂ、２０８Ｃおよび２０８Ｄを経由してキャッシュ・メモリ４８へ与えられる。２つの連続する１６ビットの読取りサイクルの間に４つのテクセルが同時に４つのインターリーブからアクセスされる。４つのテクセルは、バス２１０Ａ、２１０Ｂ、２１０Ｃおよび２１０Ｄ経由で、テクセル・データＡのＦＩＦＯの１つ（２１４Ａ０または２１４Ａ１）に、テクセル・データＢのＦＩＦＯの１つ（２１４Ｂ０または２１４Ｂ１）に、テクセル・データＣのＦＩＦＯの１つ（２１４Ｃ０または２１４Ｃ１）に、テクセル・データＤのＦＩＦＯの１つ（２１４Ｄ０または２１４Ｄ１）にそれぞれ送られる。各インターリーブＡ－Ｄに対応するＦＩＦＯのペア（すなわち０および１）は交互にロードされる。例えば、インターリーブＡから読み取られる第１のテクセルがテクセル・データＦＩＦＯ２１４Ａ０に記憶され、インターリーブＡから読み取られる第２のテクセルがテクセル・データＦＩＦＯ２１４Ａ１に記憶され、インターリーブＡからの第３のテクセルがテクセル・データＦＩＦＯ２１４Ａ０に記憶されるというようにテクセルが交互に記憶される。このような交互方式を使用する理由は以下の通りである。

【００７４】

テクセル・データＦＩＦＯの各々は、幅３２ビットで深さ８段階である。組合せれば、８つのＦＩＦＯ２１４は、８つのパイプライン化された段階を記憶する。各段階は、３線形補間の間所望のテクセル・データを決定するために使用される８つのテクセルを含む。バス２１０Ａ、２１０Ｂ、２１０Ｃおよび２１０Ｄは、幅１６ビットである。各インターリ

ーブにおける各ＳＤＲＡＭペアは、各読取りサイクルの間に１６ビットのデータを提供する。各バースト読み取りの間、第１の１６ビットが、各ＳＤＲＡＭペアから第１の１６ビット・レジスタ（図示されてないない）に送られ、次の１６ビットが、各ＳＤＲＡＭペアから第２の１６ビット・レジスタ（やはり図示されてないない）に送られる。バースト読み取りの第２サイクルの終了時点で、両方のレジスタからのデータが、対応する３２ビット・バス２１２Ａ、２１２Ｂ、２１２Ｃまたは２１２Ｄ上へ送られる。任意のピクセルに関する所望のテクセル・データを決定するため、テクセル補間回路７６が、ＦＩＦＯにアクセスして次の段階の８つのテクセルを読み取り、上述の方法でそれらのテクセルを補間する。補間結果のテクセル・データが、バス２８経由でフレーム・バッファ基板１４（図２）へ送られ、そこで、上述の方法で表示画面ピクセルをレンダリングするために使用される。

【００７５】

３線形補間が実行される時、任意のピクセルに関する所望のテクセル・データは、ある１つのＭＩＰマップの４つのテクセルを補間し、隣接する別のＭＩＰマップの４つのテクセルを補間して得られる。隣接する表示画面ピクセルは、一般的には連続的にレンダリングされる。隣接する表示画面ピクセルは、頻繁に、１つのテクスチャＭＩＰマップの隣接位置に対応する。この結果、連続的にレンダリングされるプリミティブに関して所望のテクセル・データを補間する際いくつかの共通のテクセル・データが使用されることがよくある。本発明の１つの実施形態において、多数の近接した読み取りサイクル内で共通のテクセル・データが多数回アクセスされる場合、キャッシュは、最初の読取りについてのみアクセスされるだけで、後続の読取りの各々についてはキャッシュ読取りサイクルを節約する。最も最近読まれたテクセルが、テクセル・データＦＩＦＯ内に記憶される。このように、それらテクセルに対する後続のアクセスは、キャッシュではなくＦＩＦＯからなされる。これによって、必要とされるキャッシュ・アクセス数が減り、システムの帯域幅が増大する。

【００７６】

テクセル・データ経路Ａ、Ｂ、ＣおよびＤの各々について、前回のピクセルに関してテクセル・データＦＩＦＯ０または１の１つに最も最近書き込まれたテクセル・データが、キャッシュをアクセスするためパイプライン位置に現在あるテクセル・データと一致する場合、キャッシュ・アクセス・コマンドは、対応するキャッシュ・アクセスＦＩＦＯ２０６Ａ、Ｂ、ＣまたはＤに送られない。代わりに、テクセル・データが対応するテクセル・データＦＩＦＯ２１４Ａ、Ｂ、ＣまたはＤの最も最近書かれた位置に記憶されていることを示すコマンドがテクセル補間器に送られる。キャッシュをアクセスするためパイプライン位置に現在あるテクセル・データに対応するテクセル・データが、対応するテクセル・データＦＩＦＯの最も最近書き込まれた位置のデータと一致しない場合、経路Ａ、Ｂ、ＣおよびＤのいずれについても、テクセル・キャッシュ・アクセス・コマンドが対応するテクセル・キャッシュ・アクセス・コマンドＦＩＦＯに送られ、キャッシュ・メモリ４８からそのテクセル・データが読み取られる。

【００７７】

キャッシュ・アクセスを考慮しなければならない現在パイプライン位置にあるピクセルについてインターリーブＡないしＤのいくつかが異なる結果を生み出す点は理解されるべきであろう。例えば、連続的ピクセルに関する共通のテクセル・データがインターリーブＡには存在するがインターリーブＢ－Ｄには存在しないことがある。そのような状況においては、キャッシュからテクセル・データをアクセスするためにパイプライン位置にある第２の連続ピクセルに関してテクセル・データがインターリーブＢ－Ｄから読み取られるであろうが、その第２のピクセルに関するインターリーブＡからのテクセル・データは、テクセル・データＦＩＦＯ２１４Ａ０または２１４Ａｌの１つの同じ位置から読み取られるであろう。キャッシュをアクセスせずに複数のピクセルに関してテクセル・データＦＩＦＯからテクセルが再読み取りされる場合、本方式は帯域幅を節約する。

【００７８】

テクセル補間回路７６は、５３ビット・コマンドを境界検査器７２から５３ビット・バス２１８経由で受け取るテクセル補間回路コマンドＦＩＦＯ２１６を含む。テクセル補間回路コマンドＦＩＦＯは、各サイクルの間所望のテクセル・データを補間する際に使用されるべきテクセル・データをどこのテクセル・データＦＩＦＯ位置が含むかを補間回路に標示する最高１６のコマンドを記憶することができる。補間回路コマンドは、また、（点標本抽出、双線形または３線形などの）補間モードを示し、補間の際に各テクセルが加重される方法を指定するＳおよびＴ座標の勾配および小数値を含む。コマンドは、（双線形補間の場合）４個または（３線形補間の場合）８個のテクセルがＦＩＦＯ２１４Ａ０、Ａ１、Ｂ０、Ｂ１、Ｃ０、Ｃ１、Ｄ０またはＤ１のいずれから読み取られるべきか、また、該テクセルが新しいか古いかを示すデータを含む。テクセル・データがその経路のいずれかのテクセル・データＦＩＦＯの位置に最も最近書き込まれたテクセル・データと異なる場合は、そのテクセル・データは新しいという。新しい場合キャッシュ読取りが必要とされる。テクセル・データがいずれかのテクセル・データＦＩＦＯの位置に最も最近書き込まれたものと同じ場合は、そのテクセル・データは古いという。古い場合キャッシュ読取りは必要とされない。テクセル・データが新しい時、ＦＩＦＯ読取りポインタが、ＦＩＦＯの内の次の位置へ移動されなければならず、一方、テクセル・データが古い時、同じデータが同じＦＩＦＯ位置から読み取られ、読取りポインタを移動する必要はない。

【００７９】

図９に示されたテクセル・アクセス回路の動作を図１０および図１１を参照しながら以下に更に例示する。図１０は、上位ＭＩＰマップの複数のテクセルおよび（サイズ的に小さい）下位ＭＩＰマップの複数のテクセルを示す。テクセルは、上記図８の場合に使用した標記法と同様に、Ａｎ、Ｂｎ、ＣｎおよびＤｎというラベルをつけられている（但しｎは整数）。レンダリングされるべき７つのピクセルに、Ｐ０、Ｐ１、Ｐ６というラベルがつけられている。図に示されているように、レンダリングされるべき（複数）ピクセルは、ＭＩＰマップのテクセルに直接対応していない。この例においては、３線形補間法が実行され、上位マップから４つのテクセルおよび低位マップから４つのテクセルがアクセスされ各ピクセルごとに補間される。進行方向は、レンダリングの方向であって、ピクセルに付けられた数字番号に対応する。

【００８０】

図１１は、キャッシュ・アクセス・コマンドＦＩＦＯ（２０６Ａ）、テクセル・データＦＩＦＯＡ０（２１４Ａ０）、テクセル・データＦＩＦＯＡ１（２１４Ａ１）およびテクセル補間回路コマンドＦＩＦＯ２１６を示す。他のテクセル・データ経路Ｂ、ＣおよびＤの各々に関するＦＩＦＯも同じ形態でで動作するので、便宜上テクセル・データ経路Ａに関連するＦＩＦＯだけが示されている。各ＦＩＦＯバッファは、データが読み書きされるべきＦＩＦＯ内の単一の位置をそれぞれポイントする書込みポインタおよび読取りポインタを含む。両ポインタは、本実施例では１回につき１位置移動することができる。

【００８１】

ピクセルＰ０は、上位マップにおけるテクセルＡ０、Ｂ０、Ｃ０およびＤ０に、そして下位マップにおけるテクセルＡ０、Ｂ０、Ｃ０およびＤ０に対応しているので、これら８つのテクセルが補間され、ピクセルＰ０に対するテクセル・データが生成される。ピクセルＰ０について、キャッシュから読み取られるテクセル・データをテクセル・データＦＩＦＯ２１４Ａ０に書き込むべきアドレスと共に、上位マップのテクセルＡ０のアドレス（図１０でｕＡ０と標記されている）が、キャッシュ・アクセス・コマンドＦＩＦＯ２０６Ａの最初の位置に書き込まれる。次に、キャッシュ・アクセス・コマンドＦＩＦＯ２０６Ａの書込みポインタが１位置移動され、キャッシュから読み取られるテクセル・データをテクセル・データＦＩＦＯ２１４Ａ１に書き込むべきアドレスと共に、下位マップのテクセルＡ０のアドレス（図１０でｌＡ０と標記されている）がＦＩＦＯの次の位置に書き込まれる。このように、上述の理由からテクセル・データＦＩＦＯ０と１は交互に使用される。キャッシュ・アクセス・コマンドＦＩＦＯ２０６Ｂ－２０６Ｄは低位マップのテクセルＢ０、Ｃ０およびＤ０に関して同様の方法で更新される。

【００８２】
ピクセルＰ１について、上位および下位マップのテクセルＡ１がアドレスｕＡ１およびｌＡ１にそれぞれ補間のため記憶される。上位および低位マップのテクセルＡ１は新しいテクセルであり、前のピクセルＰ０からのテクセルに対応していないので、それらはキャッシュからアクセスされる。このようにして、これらテクセルのテクセル・アドレスが、それらのアドレスから読み取られるテクセル・データがテクセル・データＦＩＦＯ２１４Ａ０および２１４Ａ１に書き込まれるべきことをそれぞれ示す対応するアドレスと共に、キャッシュ・アクセス・コマンドＦＩＦＯの後続の２つの位置に付加される。図１１は、上記情報で更新された後のキャッシュ・アクセス・コマンドＦＩＦＯ２０６Ａを示す。

【００８３】
最初の２つのピクセルＰ０およびＰ１については共通のＡアドレス指定テクセルが存在しないので、両者に関するテクセル・データを取り出すためキャッシュ・メモリがアクセスされる。最初のコマンドがキャッシュ・アクセス・コマンドＦＩＦＯ２０６Ａから読み取られ、アドレスｕＡ０にあるテクセル・データがキャッシュ・メモリから読み取られテクセル・データＦＩＦＯ２１４Ａ０の最初の位置に書き込まれる。そして、次のコマンドがキャッシュ・アクセス・コマンドＦＩＦＯから読み取られ、アドレスｌＡ０にあるテクセル・データがキャッシュ・メモリから読み取られテクセル・データＦＩＦＯ２１４Ａ１の最初の位置に書き込まれる。次のコマンドがキャッシュ・アクセス・コマンドＦＩＦＯから読み取られ、アドレスｕＡ１にあるテクセル・データがキャッシュ・メモリから読み取られテクセル・データＦＩＦＯ２１４Ａ０の次の位置に書き込まれる。最後に、第４番目のコマンドがキャッシュ・アクセス・コマンドＦＩＦＯから読み取られ、アドレスｌＡ１にあるテクセル・データがキャッシュ・メモリから読み取られテクセル・データＦＩＦＯ２１４Ａ１の次の位置に書き込まれる。

【００８４】
次のピクセルＰ２をレンダリングするため、アドレスｕＡ１およびｌＡ１のテクセルが補間される必要がある。これらのテクセルが前にレンダリングされたピクセルＰ１についてアクセスされたものなので、それらは、テクセル・データＦＩＦＯ２１４Ａ０および２１４Ａ１の最も最近書き込まれたエントリにそれぞれ記憶されている。従って、それらのテクセルについて新しいキャッシュ・アクセス・コマンドがキャッシュ・アクセス・コマンドＦＩＦＯ２０６Ａに送られることはない。その代わり、ピクセルＰ１に関する所望のテクセル・データが補間された後、アドレスｕＡ１およびｌＡ１に記憶されたテクセル・データが、テクセル補間回路によってテクセル・データＦＩＦＯ２１４Ａ０および２１４Ａｌの最も最近読まれた位置からそれぞれアクセスされ、キャッシュへのアクセスは必要とされない。直接ＦＩＦＯバッファからデータを読み取る方が、キャッシュ・メモリからデータをアクセスする場合に比較して所要時間は少ない。従って、キャッシュ・アクセスを減らす本発明のＦＩＦＯバッファはシステムの帯域幅を増加させる。

【００８５】
上述のように、インターリーブＡ－Ｄの各々に対応するテクセル・データＦＩＦＯ２１４は、別々に制御されるＦＩＦＯゼロおよび１を含む。ＦＩＦＯは、このような形態で３線形補間を実行するために能率的に分割される。上述から理解されるように、上記の実施形態において、テクセル・データＦＩＦＯ２１４の各々は、後続の読取りが同じエントリをポイントできるポインタを維持することによって、その最も最近読み取られたエントリへのアクセスを提供する。このため、連続的な読取りサイクルの間、各インターリーブが２つのマップの間で交互に読み取りを行うが、独立したＦＩＦＯが、１つのマップ内で連続的読み取りを実行することができるので、ＦＩＦＯへの連続アクセスにおいて読取りポインタが同じテクセル・データをポイントすることが可能となる。

【００８６】
各ピクセルがタイル作成／境界検査器７２によって処理されコマンドがキャッシュ・アクセス・コマンドＦＩＦＯに送られる場合、コマンドはまたテクセル補間回路コマンドＦＩＦＯ２１６に書き込まれる。例えば、ピクセルＰ０についてアドレスｕＡ０でテクセルに

アクセスすべきコマンドがキャッシュ・アクセス・コマンドＦＩＦＯに送られると、コマンドＮｅｗ０が、テクセル補間回路コマンドＦＩＦＯ２１６の最初の位置に送られる。コマンドＮｅｗ０は、インターリーブＡからの次のテクセル・データがキャッシュからアクセスされてテクセル・データＦＩＦＯ２１４Ａ０に渡されること、および、ＦＩＦＯからテクセル・データを読むためにテクセル補間回路は最も最近読み取られた位置から１位置ＦＩＦＯ読取りポインタを移動させなければならないことをテクセル補間回路に示す。

【００８７】

キャッシュ・アクセス・コマンドＦＩＦＯに送られるテクセル・アドレスｌＡ０に対応する次のコマンドについて、コマンドＮｅｗ１が、テクセル補間回路コマンドＦＩＦＯの次の位置に書き込まれる。コマンドＮｅｗ１は、インターリーブＡからの次のテクセル・データも新しいもので、テクセル・データ補間回路２１４Ａ１から読み取らなければならないことをテクセル補間回路に示す。同様に、ピクセルＰ１に対応するテクセル・アドレスｕＡ１およびｌＡ１に関連するコマンドに関して、コマンドＮｅｗ０およびＮｅｗ１がそれぞれテクセル補間器コマンドＦＩＦＯ２１６の次の２つの位置に書き込まれる。

【００８８】

ピクセルＰ２については、アドレスｕＡ１およびｌＡ１のテクセル・データが前のピクセルＰ１のためＦＩＦＯに書かれたデータと同一であるので、テクセル補間器コマンドＦＩＦＯ２１６の次の２つの位置に書かれるコマンドはＯｌｄ０およびＯｌｄ１であり、次のテクセル・データはテクセル・データＦＩＦＯ２１４Ａ０および２１４Ａ１の最も最近読まれた位置から再読み取りされるべきであることをテクセル補間器に対してそれぞれ標示する。Ｏｌｄ０およびＯｌｄ１コマンドは、ＦＩＦＯから次のテクセル・データを読むため最も最近読み取られた位置からＦＩＦＯ読取りポインタを移動する必要がないことをテクセル補間回路に示す。

【００８９】

図１０は、次の３つのテーブルをリストしている。第１のテーブルはピクセルの各々について補間される必要があるテクセルを示し、第２のテーブルは、テクセル・データＦＩＦＯＡ０、Ｂ０、Ｃ０およびＤ０に記憶される必要がある別々のテクセル・データ値をリストし、第３のテーブルは、テクセル・データＦＩＦＯＡ１、Ｂ１、Ｃ１およびＤ１に記憶される必要がある別々のテクセル・データ値をリストする。ブランク空間は、キャッシュから再び読まれる必要がなく、ＦＩＦＯからアクセスされることができるようにキャッシュから既に読み込まれた共通テクセル・データを示す。図が示す通り、複数のピクセルについて所望のテクセル・データが補間される時、本発明のＦＩＦＯ方式によって多数のキャッシュ・アクセスが節約され、この結果システムの帯域幅が増大する。

【００９０】

図１２は、各インターリーブにおいて、あるピクセルについて読み取られるべきテクセル・データが最も最近レンダリングされたピクセルについて読み込まれたか否かを判断するため、テクスチャ・マッピング・チップによって使用される回路のブロック図である。この回路は、新しいデータをキャッシュから読み取るように指示する新しいコマンドをキャッシュ・アクセス・コマンドＦＩＦＯの１つに書き込むべきか、あるいは、テクセル・データは古いのでテクセル・データＦＩＦＯの１つから読み取られるべきことを示すコマンドをテクセル補間器コマンドＦＩＦＯに書き込むべきか判断するために使用される。図１２は、インターリーブＡに対応する１つの回路だけを示しているが、インターリーブＢ、ＣおよびＤに対して、同様の回路が用意される。この回路は、タイル作成／境界検査器の最適化エレメントの範囲内に配置される。補間されるべき各テクセルについてタイル作成／境界検査器によって受け取られる補間されたＳ，Ｔ値を用いて、最適化エレメントは、バス２２０Ａ上に（ブロック・タグおよびテクセル・アドレスを含む）テクセル・アドレスを出力する。テクセル・データＦＩＦＯ２１４Ａ０および２１４Ａ１に割り当てられた最も最近処理されたテクセルのアドレスは、アドレス・レジスタ２２２Ａ０および２２２Ａ１にそれぞれ記憶されている。現在のテクセル・アドレスが、比較器２２４Ａ０および２２４Ａ１によって、レジスタ２２２Ａ０および２２２Ａ１に記憶されているテクセル・

アドレスとそれぞれ比較される。

【００９１】

現在のテクセル・アドレスが、レジスタ２２２Ａ０および２２２Ａ１に記憶されているアドレスのいずれとも一致しない場合、そのテクセル・アドレスに対応するテクセル・データが、キャッシュ・メモリからアクセスされる必要があり、適切なコマンドがキャッシュ・アクセス・コマンドＦＩＦＯに書かれる。しかし、テクセル・アドレスが、レジスタ２２２Ａ０および２２２Ａ１に記憶されているアドレスと一致する場合、テクセル・データはテクセル・データＦＩＦＯ２１２Ａ０または２１２Ａ１にそれぞれ記憶されていて、そのアドレスに対応するテクセル・データをアクセスする直前にテクセル補間器によって読み取られる。従って、キャッシュ・アクセス・コマンドはキャッシュ・アクセス・コマンドＦＩＦＯに書かれず、テクセル・データは古いので、読取りポインタを動かすことなく最も最近読まれたＦＩＦＯ位置からアクセスされるべきことを示すコマンドが、対応するテクセル補間器コマンドＦＩＦＯに書き込まれる。

【００９２】

Ⅴ．テクスチャ・データ・ブロックの構成

図１は、８×８テクセルの基本マップ１００を含む一連の平方テクスチャＭＩＰマップを示す。基本マップを基に、サイズ的にフィルタして、最小サイズのマップ１０８まで連続的マップの各々が作成される。最小サイズのマップ１０８にはマップ番号ゼロが割り当てられ、サイズが大きくなるマップ毎に番号を１ずつ増分する。従って，本例の場合の基本マップ１００はマップ番号３を持つ。マップ番号は、後述する方法で、テクスチャ・データの各ブロックに対するブロック・タグを決定する際に使用される。このマップ番号付け方式に従って、１×１テクスチャ基本マップを仮定すると、マップ番号１０は１０２４×１０２４テクセルのマップに対応し、マップ番号９は５１２×５１２テクセルのマップに、マップ番号８は２５６×２５６テクセルのマップにというようにそれぞれ対応する。テクスチャ基本マップが１×１でなければ、マップ番号１０は、１０２４テクセルより大きい次元を持つマップに対応する。ここでの記述は、正方形のテクスチャ・ベース・マップを仮定しているが、長方形のマップも可能である。長方形の場合、マップ番号は、マップの長い方の次元のテクセル数によって決定される。例えば、マップ番号１０を持つ長方形のマップは、長次元に１０２４以上のテクセルを持つ。上記以外のマップ番号付け法を使用できる点は理解されるべきであろう。

【００９３】

マップ番号１０を持つ正方形１０２４×１０２４テクセル・マップは、マップ内の各テクセル位置をユニークに識別するため１０ビットのＳ座標Ｓ［９：０］および１０ビットのＴ座標Ｔ［９：０］を必要とする。同様に、マップ番号９を持つマップは、マップ内の各テクセル位置を識別するため９ビットのＳおよびＴ座標を必要とし、マップ番号８を持つマップは、マップ内の各テクセル位置を識別するため８ビットのＳおよびＴ座標を必要とするというように、以下のマップ番号について同様となる。任意のピクセルに対応するＭＩＰマップのテクセルの位置をユニークに識別するＳおよびＴ座標は上述の方法で補間される。

【００９４】

詳細は後述するが、テクスチャ・データは、（図２の）ホスト・コンピュータ１５の主メモリ１７に２５６×２５６テクセルのブロックの形式で記憶される。キャッシュ・ミスが発生すると、キャッシュ・ミスのあったテクスチャ・データのブロックを識別するキャッシュ・タグが、ホスト・コンピュータによって読み取られ、次に、そのブロックのテクスチャ・データがテクスチャ・マッピング基板のキャッシュ・メモリ４８へダウンロードされる。本発明の実施形態において、任意の１時点で、６４ブロックのテクスチャ・データがキャッシュ・メモリに記憶されることができる。これらの６４ブロックのテクスチャ・データは、１つまたは複数のテクスチャの複数のＭＩＰマップからのデータを含むことができる。各ブロックは、それをユニークに識別するブロック・タグを持つ。９以上のマップ番号を持つＭＩＰマップは、２５６×２５６を超えるテクセルを含み、従って，複数の

ブロックの形態で記憶される。複数ブロックの形態で記憶されるマップに対する高位Ｓ，Ｔ座標は、マップを記憶するデータ・ブロックに関するブロック・タグに含められる。

【００９５】

例えば、マップ番号９を持つＭＩＰマップは、５１２のテクセルに等しい１つの次元を持ち、正方形の場合は、サイズ的に５１２×５１２テクセルである。（正方形マップを仮定すると）マップは１ブロック２５６×２５６テクセルの４つのブロックに分割される。従って、それらのブロックの各々に対するブロック・タグは、マップの範囲内でのブロックの位置を識別する１つの高位Ｓ座標ビット（Ｓ［８］）および１つの高位Ｔ座標ビット（Ｔ［８］）を含む。同様に、マップ番号１０を持つＭＩＰマップはサイズ的に１０２４×１０２４テクセルであり、１ブロック２５６×２５６テクセルの１６のブロックに分割される。従って、それらのブロックの各々に対するブロック・タグは、マップの範囲内でのブロックの位置を識別する２つの高位Ｓ座標ビット（Ｓ［９：８］）および２つの高位Ｔ座標ビット（Ｔ［９：８］）を含む。

【００９６】

後述するが、補間の間システムの帯域幅を減らすため、隣接するＭＩＰマップの同じ部分が反対側のＳＤＲＡＭバンクに記憶されるように、テクスチャＭＩＰマップは更に小さく分割されてメモリに記憶される。加えて、キャッシュ・メモリ内のメモリ空間を効率的に利用するため、２５６×２５６未満テクセルの複数マップをキャッシュ・メモリの１つのブロックの中に記憶することができる。

【００９７】

図１３は、

ＬＡ

９５

という面画像を含む特定テクスチャに関する一組のテクスチャＭＩＰマップを示す。図１３に示されるように、あるテクスチャに関する一連のＭＩＰマップにおけるＭＩＰマップの各々は、１つの平方テクスチャ・マップに対して等しいサイズの４つの象限に分割される。図１２に示される例においては、基本マップは、マップ番号９を持ち、（画像Ｌを含む）９Ｑ１、（画像Ａを含む）９Ｑ２、（画像９を含む）９Ｑ３および（画像５を含む）９Ｑ４の象限に分割されている。同様にマップ番号８は、それぞれＬ，Ａ，９および５を含む象限８Ｑ１，８Ｑ２，８Ｑ３および８Ｑ４に分割されている。同様にマップ番号７は、それぞれＬ，Ａ，９および５を含む象限７Ｑ１，７Ｑ２，７Ｑ３および７Ｑ４に分割されている。同様に、更に小さいマップは同様の象限に小分割されている。

【００９８】

各ＭＩＰマップの２つの象限が、キャッシュを形成するＳＤＲＡＭの１つのバンクに記憶され、残りの２つの象限が反対側のバンクに記憶される。本発明のテクスチャ・データ配置方式に従えば、８以上の番号の（すなわちサイズが２５６×２５６テクセル以上の）基本マップを持つテクスチャに関しては、そのテクスチャのＭＩＰマップすべての象限のすべてについてメモリ空間のブロック内のメモリ位置はあらかじめ定められている。例えば、図１４に示されるように、マップ番号９の象限９Ｑ１および９Ｑ４は、キャッシュ・バンク１内の別々のブロックに記憶され、象限９Ｑ２および９Ｑ３は、キャッシュ・バンク０内の別々のブロックに記憶される。隣接するＭＩＰマップの対応する象限は、反対側のバンク内のブロックに記憶される。この例において、それぞれ象限９Ｑ１および９Ｑ４をフィルタしたデータを含む象限８Ｑ１および８Ｑ４は、キャッシュ・バンク０内の同じブロックに記憶される。同様に、それぞれ象限９Ｑ２および９Ｑ３をフィルタしたデータを含む象限８Ｑ２および８Ｑ３は、キャッシュ・バンク１内の同じブロックに記憶される。図１４は、図１３に対してスケールが合うように描かれてはいない。図１３のマップの象限が、対応する図１４のものと同じ大きさであることは理解されなければならない。

【００９９】

マップのそれぞれのサイズに従って、マップ番号９の各象限は完全な２５６×２５６テクセル・ブロックを占めるが、マップ番号８の４象限は各々ブロックの１／４だけを占める

。従って、象限８Ｑ２および８Ｑ３は合わせて同じブロックの１／２を占め、象限８Ｑ１および８Ｑ４は、反対のバンク内のもう１つのブロックの１／２を占める。キャッシュ・メモリ空間を効率的に割り当てるため、それらブロックの各々の中で空いている位置は、マップ番号７以下の適切な象限によって占められる。従って、ゼロないし８の番号を持つマップのすべては、それぞれ別のバンクにある２つのブロックを占める。

【０１００】

８以下のマップ番号を持つマップに関する４象限の位置は、（８以上のマップ番号を持つ基本マップを所与として）、図１４に示される形態にあらかじめ定められる。図に示されているように、右上の象限８Ｑ２および左下象限８Ｑ３は同じ物理的関係を維持して、それぞれ第１のブロックの右上および左下の象限を占め、左上の象限８Ｑ１および右下象限８Ｑ４も同じ物理的関係を維持して、第１のブロックとは異なるバンクにある第２のブロックの左上および右下の象限をそれぞれ占めている。また、象限７Ｑ１および象限７Ｑ４は同じ物理的関係を維持して、それぞれ第１のブロックの左上の象限を占め、象限７Ｑ２および象限７Ｑ３は同じ物理的関係を維持して、第２のブロックの右上の象限をそれぞれ占めている。

【０１０１】

３線形補間の間、１つのピクセルが、１つのＭＩＰマップの中の４つのテクセルと隣接するＭＩＰマップの中の４つのテクセルの間にあるテクスチャ・マップの位置に対応すれば、すべての８つのテクセルがキャッシュからアクセスされる。両方のＭＩＰマップからアクセスされるテクセルは、大きい方のマップのデータをフィルタリングした小さい方のマップのデータと共に、共通のテクスチャ・データを含む。上述のように、オブジェクトのピクセルがレンダリングされる時、隣接ピクセルは、そのテクスチャについて同じ２つのＭＩＰマップに対応することがしばしばあり、２つのマップを記憶するキャッシュ・ブロックの間でキャッシュへの読取りを連続的に切り換える必要が生じる。キャッシュＳＤＲＡＭチップの異なるバンクに隣接ＭＩＰの共通データを記憶することによって、連続的な読取りサイクルの間２つのＭＩＰマップの間でのキャッシュ読取り切り換えによる再ページングの負荷が発生しない。これは、３線形補間の効率的な実施を提供する。

【０１０２】

上述の説明から理解されるように、テクスチャが８以上のマップ番号を持つ基本マップを含む場合、そのテクスチャに対するブロック間のＭＩＰマップ割り付けは、本発明の上述の実施例に従って、あらかじめ定められている。すなわち、マップ番号８を持つマップの２つの象限が、図１４に関して上述したように、バンクの１つの範囲内の第１のブロックの予め定められた位置を占め、マップ番号８を持つマップの別の２つの象限が、反対バンクの別の１つのブロックの範囲内の予め定められた反対の位置を占める。しかし、マップ番号７以下の基本マップを持つテクスチャについては、（各バンクに１つのブロックの）２つのブロック内の複数の位置がマップを記憶するために使用可能であり、ホスト・コンピュータによって選択される。複数のマップ部分が単一ブロックのデータを共有する時、共有されたブロック内の各マップの位置を識別するため、以下に記述される方法で、サブテクスチャ識別子（ＩＤ）が割り当てられる。

【０１０３】

図１３の一連のＭＩＰマップの構成に加えて、図１４は、異なるテクスチャからの第２の一連のＭＩＰマップ（図でチェッカー盤模様部分）がメモリ・ブロックの間に割り当てられる。この第２のテクスチャのＭＩＰマップは小分割され、第１のテクスチャと同じ方法で別々のブロックに記憶される。図１４の構成が別々のブロックに構成される異なるテクスチャのＭＩＰマップを示してはいるが、２つの異なるテクスチャからのテクスチャ・データを同じブロック内に記憶することもできる点は理解されるべきであろう。

【０１０４】

上述のとおり、本発明の１つ実施形態において、キャッシュ・メモリは、テクスチャ・マッピング・データの最高６４までのブロック（各ブロックは２５６×２５６テクセルを含む）を記憶することができる。キャッシュ・メモリは、ブロック０－３１を収納するバン

ク０およびブロック３２－６３を収納するバンク１という２つのバンクに区分される。キャッシュ・ディレクトリは、キャッシュのブロックに対応する最高６４までのブロック・タグ・エントリを含む。キャッシュ・ディレクトリ内の各ブロック・タグの物理的な位置は、キャッシュ・メモリ内のテクスチャ・データの対応するブロックの物理的な位置を識別する。ブロックの位置を示すブロック・タグから、ブロック・インデックスが生成される。キャッシュのテクセルに関するキャッシュ・アドレスは、ブロックに対するブロック・インデックスおよびキャッシュ・メモリ内のテクセル・アドレスによって形成される。テクセル・アドレスは、テクセルに関する補間された低位Ｓ，Ｔ座標を含み、また場合によっては以下に述べるようにサブテクスチャＩＤのビットを含む。

【０１０５】

図１５は、４象限に小区分されているマップ番号９を持つテクスチャＭＩＰマップの１例を示す。ＭＩＰマップは５１２×５ｌ２テクセルであり、従って、各象限はサイズ２５６×２５６テクセルでありメモリの１ブロックに対応する。本発明の１つの実施形態に従って、ＭＩＰマップの各象限に割り当てられるべきキャッシュ・バンクを決定する簡単な方式がホスト・コンピュータによって実施される。ＭＩＰマップの４象限の各々について、象限に関するＳおよびＴ座標の最上位ビットの値に対する論理的排他ＯＲ演算の結果が、象限が割り当てられるキャッシュＳＤＲＡＭバンクを指し示す。

【０１０６】

５１２×５ｌ２テクセルのマップについては、９つのＳ座標ビットＳ［８：０］および９つのＴ座標ビットＴ［８：０］がマップ内の各テクセルの位置を指定する。象限境界は、ＳおよびＴ座標ビットＳ［８］およびＴ［８］によって表されるＳおよびＴ次元両方の中間点に定められる。従って、マップ番号９を持つＭＩＰマップの４つの象限の各々に関するキャッシュ・バンクを決定するため、各象限の対応する最上位ＳおよびＴ座標ビットＳ［８］およびＴ［８］の値に対する論理的排他ＯＲ演算が実行される。同様に、マップ番号１０を持つＭＩＰマップに関しては、そのの４つの象限の各々に関するキャッシュ・バンクは、各象限の対応する最上位ＳおよびＴ座標ビットＳ［９］およびＴ［９］の値に対する論理的排他ＯＲ演算によって決定される。奇数のマップ番号を持つＭＩＰマップについては、隣接マップからの共通データが異なるバンクに記憶されるようにするため排他ＯＲ演算の結果が反転される。

【０１０７】

図１５で示される例において、ブロック１ないしブロック４は、それぞれ、左上象限、右上象限、左下象限および右下象限の５１２×５１２テクセル・マップに対応する。ブロック１ないしブロック４について、ビットＳ［８］、Ｔ［８］はそれぞれ［０，０］、［１，０］、［０，１］および［１，１］に等しい。従って、ブロック１について ＸＯＲ Ｓ［８］ＸＯＲ Ｔ［８］演算の結果はゼロとなる。マップが奇数マップ番号（すなわち９）を持つので、この結果の反転値（すなわち１）によって、ブロック１はキャッシュ・バンク１に記憶されるべきことが標示される。ブロック２については、Ｓ［８］ＸＯＲ Ｔ［８］演算の結果の反転がゼロであって、ブロック２はキャッシュ・バンク０に記憶されるべきことが標示される。ブロック３およびブロック４については、 Ｓ［８］ＸＯＲ Ｔ［８］演算の結果の反転がそれぞれ１およびゼロであって、ブロック３はキャッシュ・バンク１に、ブロック４はキャッシュ・バンク０にそれぞれ記憶されるべきことが標示される。

【０１０８】

図１５の例で示されているものと同じテクスチャについてマップ番号１０を持つマップに関する限り、そのマップのサイズが１０２４×１０２４テクセルであるため１６個の２５６×２５６テクセル・ブロックに区分けされる。各ブロック毎に、Ｓ［９］ＸＯＲ Ｔ［９］演算の結果がその特定ブロックに対するバンク番号を標示する。マップ番号１０を持つマップの各ブロック毎のＸＯＲ演算の結果は、マップ番号９を持つ隣接マップの場合のように反転されずに、これら２つの対応する象限は異なるキャッシュ・バンクに記憶される。

【０１０９】
マップのサイズに応じて、マップを表すテクスチャ・データ・ブロックのブロック・タグは、特定のＭＩＰマップ内のブロックの位置を示す少くとも１つの高位Ｓ座標ビットおよび高位Ｔ座標ビットを含む。マップ番号９を持つ５１２×５１２テクセルＭＩＰマップについては、ＭＩＰマップ内の各ブロックの位置を示すためにブロック・タグ内にただ１つのＳ座標ビットおよびＴ座標ビットが必要とされる。マップ番号１０を持ち、１６ブロックのデータを含む１０２４×１０２４テクセルＭＩＰマップについては、ＭＩＰマップ内の各ブロックの位置を示すためにブロック・タグ内に２つのＳ座標ビットおよびＴ座標ビットが必要とされる。８以下のマップ番号を持つマップに関する限り、ブロック・タグにＳおよびＴビットは必要とされない。テクスチャＭＩＰマップ・データをホスト・コンピュータの主メモリからキャッシュ・メモリへダウンロードする際、ホスト・コンピュータは、上述の排他的ＯＲ演算方式を使用してブロック・タグの上位ＳおよびＴ座標ビットをデコードして、各データ・ブロックが書き込まれるべき特定バンクを決定する。

【０１１０】
未使用メモリ空間を最小にするようにテクスチャ・データを割り当てるために、各データ・ブロックは、１サブブロックが６４×６４テクセルである１６個のサブブロックにさらに小区分される。テクスチャ・データの各サブブロックは、ブロック内の特定サブブロックの位置を識別するサブテクスチャＩＤを含む。サブテクスチャＩＤは、２つのＳビットＳ［１：０］および２つのＴビットＴ［１：０］を含む。１つまたは複数のテクスチャの１つまたは複数ＭＩＰマップからの複数サブテクスチャを１つのブロックに記憶することも可能である。

【０１１１】
図１６において、ブロック１およびブロック２が、各々１６個の６４×６４テクセル・サブブロックに小区分されているキャッシュのバンク０および１にそれぞれ割り当てられている。各ブロックのサブテクスチャは、ＳＴ０ないしＳＴ１５という符号をつけられ、２つのＳ座標ビットおよび２つのＴ座標ビットを含むサブテクスチャＩＤによって識別される。サブテクスチャは、上述のメモリ割り当て方式と整合性がとれるように、一貫した符号が付けられるが２つのキャッシュ・バンク内で鏡面反射位置を持つ。６４×６４テクセルのサブテクスチャのサイズは例示のため選択したもので、変えることはできる。一層小さいサイズのサブテクスチャは同じブロック内に更に多くのテクスチャを詰め込むことができる。サブテクスチャのサイズを小さくすればサブテクスチャＩＤが一層多くのビット必要とする点は理解されるべきであろう。

【０１１２】
レンダリングの間、一連のテクセルを補間するため、テクスチャＩＤ、サブテクスチャＩＤおよび当該テクスチャに関する基本マップのサイズを表す８ビット・ワードが、３Ｄパイプラインを経由して、それらデータを２０ビット・レジスタ（図示されてない）に一時的に記憶するタイル作成／境界検査器へ送られる。補間されるべきテクセルが異なるサブテクスチャＩＤまたはテクスチャＩＤを持つ場合、新しいデータがタイル作成／境界検査器へ送られ、レジスタに記憶される。サブテクスチャＩＤは以下に述べるようにテクセル・アドレスの一部として使用される。

【０１１３】
テクセル・アドレスがサブテクスチャＩＤの下位Ｓ，Ｔ座標ビットを含むか否かは、アドレス指定されているマップのサイズおよびそのテクスチャの基本マップのサイズに依存する。アドレス指定されているマップが、７以下のマップ・サイズであり、また、その対応する基本マップもまた７以下のサイズである場合、以下に詳細を説明するように、ブロック内のサブテクスチャの位置のアドレスを示すため、テクセル・アドレスの特定の上位アドレス・ビットが、サブテクスチャＩＤのビットを含む。上述のように、基本マップがマップ番号８以上をもつ場合、それぞれのデータ・ブロックの範囲内のそのテクスチャに関するＭＩＰマップ象限のすべての位置はあらかじめ定義されている。従って、マップ番号８以上を持つマップの１つからテクスチャが取り出される時、サブテクスチャを使用せず

、既知のあらかじめ定められた位置を使用して各象限に関するテクセル・アドレスの上位ビットが生成される。しかし、テクスチャの基本マップが７以下のマップ番号を持つ時、ＭＩＰマップ象限の位置はあらかじめ定められてなく、サブテクスチャＩＤビットをテクセル・アドレスの上位ビットとして使用してサブテクスチャの位置を決定する。

【０１１４】

上述のように、異なるテクスチャからの複数のマップは、テクスチャの基本マップが十分小さい限り、単一のデータ・ブロックの異なるサブテクスチャ内に記憶することができる。この場合、各マップについてのテクスチャ・アドレスが下位テクスチャＩＤビットを含む。例えば、４つの異なるテクスチャからのマップ番号７を持つ４つの異なるマップは、１つのブロック内で異なるサブテクスチャ内に割り当てられ、各テクスチャの基本マップのマップ番号が７である場合、サブテクスチャＩＤの１Ｓ座標ビットおよび１Ｔ座標ビットが、テクスチャを見分けるテクセル・アドレスの一部である。タイル作成／境界検査器がテクセル・アドレスを計算するルーチンは、図１８を参照して後述される。

【０１１５】

本発明の実施形態において、テクスチャＭＩＰマップ・データは、一度に１ブロックずつダウンロードされる。しかし、サブテクスチャが主メモリからダウンロードされることができるようにサブテクスチャＩＤをブロック・タグに含めることができる点は理解されるべきであろう。また、本実施形態で記述されるブロックおよびサブテクスチャのサイズは単に例示の目的のものにすぎず、アプリケーションにとって都合のよいように変更することは可能である。

【０１１６】

VI．キャッシュ・ブロック・タグおよびブロック・インデックス

キャッシュ・ディレクトリは、６４個のエントリの各々に関するブロック・タグを含み、各エントリ毎に対応するブロック・インデックスを識別する。ブロック・インデックスは、テクスチャ・データの対応するブロックの先頭が記憶されるキャッシュの物理的な位置を識別する。ブロック・タグは、図１７に示される方法でテクスチャ・データの各ブロックをユニークに識別する２３ビットの識別子である。

【０１１７】

テクスチャ・データの任意のテクセルをユニークに識別するため、そのテクセルが対応するテクスチャが識別されなければならない。本発明の１つの実施形態において、テクスチャ・マッピング・ハードウェアは、１つのテクスチャをユニークに識別する８ビット・テクスチャＩＤを導入する。加えて、同じブロック内に記憶される異なるテクスチャからのテクスチャ・データについて、４ビットのサブテクスチャＩＤが、テクスチャを識別するハードウェアによってサポートされる。このように、本発明のテクスチャ・マッピング・ハードウェアは、任意の１時点で使用可能な2^{12}すなわち４０９６個のユニークなテクスチャをサポートする。

【０１１８】

上述のとおり、各テクスチャは、一連のＭＩＰマップによって表現され、本発明の１つの実施形態において、ＭＩＰマップの各々は、一連のＭＩＰマップにおける位置を示すマップ番号を備えている。このように、任意のテクセル・データは、そのテクスチャに関するテクスチャＩＤ、サブテクスチャＩＤおよび基本マップのサイズによって識別されるだけでなく、それが対応するＭＩＰマップのマップ番号によっても識別される。最後に、テクセルは、そのＳおよびＴ座標（すなわちその補間されたＳ，Ｔ値）によって、ＭＩＰマップ内でユニークに識別される。

【０１１９】

サブテクスチャＩＤおよびテクスチャ・マップ基本サイズの他、テクセルをユニークに識別する上記のパラメータ類を使用して２３ビットのブロック・タグが生成される。マップ番号およびＳならびにＴ座標に関しては、本発明の１つの実施形態において、Ｓならびに Ｔ座標を生成するために使用されるハードウェアが１５ビットに限定されている。従って、この実施形態に関する限り、ハードウェアによってサポートされる最大のテクスチャ・

マップは、１５ビットＳフィールド［１４：０］および１５ビットＴフィールド［１４：０］を持ち、その結果、最大テクスチャ・マップは３２Ｋ×３２Ｋテクセルとなる。上述のとおり、テクセル・データの各ブロックは、２５６×２５６テクセルを含む。従って、低位ＳならびにＴビット（すなわちＴ［７：０］ならびにＳ［７：０］）がテクセル・データ・ブロック内の特定のテクセルを識別するために使用される。高位ＳならびにＴビット（すなわちＴ［１４：８］ならびにＳ［１４：８］）だけがテクセル・データの特定ブロックを識別するためブロック・タグの中で使用される。

【０１２０】

上述のとおり、各ＭＩＰマップは、その対応するテクスチャに関する一連のマップ内でそのマップを識別するマップ番号を割り当てられる。あるテクスチャに関する一連のマップにおけるＭＩＰマップの数にかかわりなく、その中の最小の（すなわち１×１テクセルのサイズの）ＭＩＰマップにマップ番号０が割り当てられる。３２Ｋ×３２Ｋテクスチャに関する一連のＭＩＰマップの最大のものは１６個のＭＩＰマップを含むので、サポートされる最大のＭＩＰマップ番号は１５である。

【０１２１】

図１７は、ブロック・タグが形成される様態を示す。ブロック・タグの上位８ビット［２２：１５］は、テクスチャ・データのブロックによって表されるテクスチャのテクスチャＩＤに対応する。ブロック・タグの低位ビット［１３：００］は、高位ＴならびにＳ座標［１４：０８］ならびにＳ［１４：０８］に対応する。ブロック・タグ［14］は、高位Ｔ座標フィールドの値と連係してマップ番号の識別を可能にするマップ・ビットに対応する。最大３２Ｋ×３２Ｋより小さいマップは、小さくなる程少ないビット数となって、ＳならびにＴアドレス・フィールド全体を使用しない点は理解されるべきであろう。図１７に示されるように、９以上のマップ番号を持つマップについては、未使用ビット中の最下位Ｔ座標ビットに対応するブロック・タグ・ビットは、論理「０」にセットされ、残りの上位Ｔ座標ビットに対応するブロック・タグ・ビットは、論理「１」にセットされる。例えば、Ｔ座標ビットすべてを使用するマップ番号１５については、マップ・ビットが論理「０」にセットされている。マップ・ビットに対応するブロック・タグ・ビット［１４：０７］および高位Ｔ座標ビット［１４：８］を読み取ることによって、左から右に読んで最初の論理「０」に出会う位置が、ブロック・タグによって表されるマップ番号を示す。論理「１」がブロック・タグ・ビット［１４：０８］のすべてに含まれている場合、そのマップ番号が８以下であることを表す。

【０１２２】

上述のように、８以下のマップ番号を持つ特定テクスチャのマップのすべては、それぞれが別のバンクに位置する２つのデータ・ブロック内に記憶される。８以下のマップ番号を持つマップの各々の２つの象限すなわち半分が、２つのブロックの各々の中に記憶される。ブロック・タグ・ビット［０７］は、８以下のマップ番号を持つマップの１／２部分の各々が２つのブロックのいずれに記憶されているかを示す。このように、８以下のマップ番号を持つマップの各々について、ブロック・タグ・ビット［０７］は、そのマップの１／２がバンク・ゼロに記憶されている場合は「０」の値を持ち、バンク１に記憶されている別の１／２について値「１」を持つ。特定テクスチャからの８以下のマップ番号を持つマップのすべてが２つのブロック内に記憶されるので、それらのブロックを識別するため、１ブロック・タグ・ビットだけが使用される点は理解されるべきであろう。従って、８以下の番号を持つマップの各々に関する特定のマップ番号は、ブロック・タグ・フィールドの一部として記憶されない。

【０１２３】

８以下の番号を持つマップの各々の各象限に関するブロック・タグ・ビット［０７］の値は、当該象限が記憶されるべきバンクを決定する方式に基づいて計算される。この方式は、マップ番号が偶数の場合各象限について実行されるＭＳＢ（上位）ビット値の論理的排他ＯＲ演算であり、奇数の場合は上記演算結果の反転値である。

【０１２４】

図１７に示されるように、上位Ｓアドレス・ビットに対応するブロック・タグ・ビット［６：０］は、Ｓアドレス・ビットが使われないマップ番号８以下の小さいマップについて論理「０」にセットされるため、論理「０」に等しくなければならないことを示すマップ番号に関連して、これらのビットのいずれかが論理「１」として検出されればキャッシュ・ディレクトリ・エントリには有効なデータが含まれていないことを示すように、上位Ｓアドレス・ビットを使用することができる。

【０１２５】

各ＭＩＰマップ象限について、該象限に関する最上位ＳならびにＴ座標の値に対する論理的排他ＯＲ（すなわちＸＯＲ）演算の結果が、該象限が割り当てられるべきキャッシュのＳＤＲＡＭバンクを指し示す。バンク番号は、偶数のマップ番号を持つマップについてはこのＸＯＲ演算結果に等しく、奇数のマップ番号を持つマップについてはこのＸＯＲ演算結果の反転値に等しい。これは、図１７のテーブルの右欄に、ＸＯＲ演算を示す記号"＾"および論理反転を示す記号"!"を用いて示されている。９以上のマップ番号を持つマップに関しては、各象限は、少なくとも１つのデータ・ブロックの全体を使用し、各ブロックは、（図１７の最後の欄に示される）ＸＯＲ演算によって指し示されるバンクに記憶される。

【０１２６】

８以下のマップ番号を持つマップについて、それらのマップのすべては、（各バンクに１つのブロックという形態で）２つのデータ・ブロックを占める。図１７のテーブルの最後の２つの行は、８以下のマップ番号を持つマップの別々の半分（２つの象限）に対応する。ブロック・タグ・ビット［０７］が、マップの半分がバンク０ブロックあるいはバンク１ブロックのいずれに記憶されるかを表す。ビット［０７］の値は、上述のＸＯＲ演算に基づいて計算される。例えばマップ番号８を持つマップの場合、マップの象限の各々について、ブロック・タグ・ビット［０７］は、Ｓ［7］ ＸＯＲ Ｔ[7］に等しい。マップ番号７を持つマップの象限の各々について、ブロック・タグ・ビット［０７］は、Ｓ［６］ ＸＯＲ Ｔ［６］の反転値に等しい。７より小さいマップ番号を持つマップの各象限に関するブロック・タグ・ビット［０７］は、同様に、番号の奇偶に応じて計算される。（８以下のマップ番号を持つ）マップ各々の２つの象限は同じブロックに記憶されるため、各マップのそれらの２つの象限が同じブロック・タグ・ビット［０７］を持つこととなる点は理解されるべきであろう。

【０１２７】

（アクセスされるべきテクセルをアドレスする）補間されたＳ，Ｔ座標とキャッシュ・ディレクトリの２３ビット・ブロック・タグの中の１つの間でヒット（一致）が発生すると、キャッシュ・ディレクトリは、そのテクセルを含むキャッシュ・ブロックが記憶されているキャッシュ・メモリの物理的な位置を識別するブロック・インデックスを生成する。キャッシュは、任意の１時点で６４ブロックのテクセル・データを記憶する。従って、キャッシュ・メモリにおけるブロック・アドレスを識別するため、先に述べたように、キャッシュに対する高位アドレス・ビットの役目を果たす６ビットのブロック・インデックス（2^6＝６４）が提供される。

【０１２８】

テクセル・アドレスは、２５６×２５６テクセル・ブロック内でアクセスされるべきテクセルの位置を示すビットＳ［７：０］およびＴ［７：０］を含む１６ビット・ワードである。テクセル・アドレスは、補間されたＳ，Ｔ座標、アクセスされるべきマップのマップ番号、テクスチャならびにサブテクスチャＩＤ、およびテクスチャの基本マップのサイズを使用して、図１８を参照して記述されるルーチンに従って計算される。上述のように、テクセルが記憶される該当するインターリーブを決定するため、テクセル・アドレスの下位（ＬＳＢ）Ｓビットおよび下位（ＬＳＢ）Ｔビットがデコードされる。テクセル・アドレスの残りの１４ビットは、（キャッシュ・アドレスの６つのＭＳＢすなわち上位ビットである）６つのブロック・インデックス・ビットと連係して、デコードされたキャッシュ・インターリーブ内のＳＤＲＡＭペアに送られるキャッシュ・アドレスとして使用され

る。
【０１２９】
VII．テクセル・アドレス計算
レンダリングの間、タイル作成／境界検査器エレメント７２は、パラメータ補間器６４から、アクセスされるべきテクセルの補間されたＳ，Ｔ値およびテクセルがアクセスされるべきマップのマップ番号を表す４ビット・ワードを受け取る。パラメータ補間器６４から受け取る補間されたＳ，Ｔ座標値の各々は、１６個の整数ビットおよび８個の小数ビットを含む。マップ番号を表す４ビット・ワードは、（テクセル・サイズ１の）マップ番号０から（３２ｋ×３２ｋテクセル・サイズの）マップ番号１５に至る範囲を含み、既に記述したように勾配から計算される。次に、補間されたＳ，Ｔ値とキャッシュ・ディレクトリにおけるブロック・タグ・エントリの比較が実行される。ブロック・タグの１つとのヒットが発生すれば、ブロック・インデックスが生成される。キャッシュ・ディレクトリ・サーチが実行されている時間と並列して、図１８を参照して記述されるルーチンに従ってテクセル・アドレスが計算される。
【０１３０】
テクセル・アドレスは、タイル作成／境界検査器によって、テクセルのテクスチャＩＤ、サブテクスチャＩＤ、マップ番号、基本マップ番号および補間されたＳ，Ｔ座標を用いて計算される。タイル作成／境界検査器はこれらの情報のすべてを持つ。アクセスされるべきユニークなテクセル毎に、タイル作成／境界検査器は、パラメータ補間器から、（Ｓ、Ｔの各々について１６個の整数ビットと８個の小数ビットを含む）補間されたＳ，Ｔ座標およびテクセルがアクセルされるべきマップ番号を表す４ビット・ワードを受け取り、更に、（パラメータ補間器を通過して来る）３Ｄパイプラインを経由して、８ビット・テクスチャＩＤ、４ビット・サブテクスチャＩＤおよび該テクスチャに対する基本マップのサイズを表す８ビット・ワードを含むコマンドを受け取る。基本マップのサイズを表す８ビット・ワードは、本発明のマップ番号づけ方式に対応し基本マップのＳ軸とＴ軸のサイズをそれぞれ定義する４つのＳビットならび４つのＴビットを含む。例えば、４ビットのＳおよびＴワードの各々は、（１テクセル次元に対応する）ゼロから（３２ｋのテクセルの次元に対応する）１５に至る範囲の値を持つことができる。テクスチャＩＤ、サブテクスチャＩＤおよび基本マップ番号を含む２０ビット・データが、キャッシュからアクセスされるべき次のテクセルに関する新しく異なるデータと置き換えられるまで、タイル作成／境界検査器内に配置される（図示されてない）２０ビット・レジスタに一時的に記憶される。この情報を使用して、タイル作成／境界検査器は、各テクセル毎のテクセル・アドレスを計算する。
【０１３１】
上述のように、（２５６×２５６テクセルの基本マップ以上に対応する）８以上のマップ番号の基本マップを持つテクスチャについては、そのテクスチャ内の各マップの象限は、テクスチャ・データ・ブロックおよびキャッシュ・メモリ・バンク内のあらかじめ定められた位置を持つ。そのようなテクスチャのテクセルに関するテクセル・アドレスの各ビットは、既知のあらかじめ定められた割り当て方式に従って計算される。しかし、（１２８×１２８テクセルの基本マップ以下に対応する）７以下のマップ番号の基本マップを持つテクスチャについては、そのテクスチャの複数マップの各象限について多数のメモリ位置が使用可能であるので、テクセル・アドレスの一定の上位ビットが、サブテクスチャＩＤのビットのすべてまたは一部を含む必要がある。
【０１３２】
テクセル・アドレスを計算するためにタイル作成／境界検査器によって実施されるルーチンが図１８の流れ図によって示される。ルーチンは、完了するため１サイクルを必要とする。ルーチンは、テクスチャ・マッピング・チップの境界検査器部分を形成する一組の論理ゲート（図示されてない）によって実施されることができる。図１８によって概要が示されているルーチンを実行する論理ゲートを実施する方法は当業者によって理解されるべきものであろう。例えば、このルーチンをＶｅｒｉｌｏｇのようなソフトウェア・シミュ

レーション言語で書き、メイン・プロセッサ上で動くＳｙｎｏｐｓｙｓのような合成ツールによって論理ゲート回路に変換することが可能である。その代替方法として、このルーチンをソフトウエアで書きプロセッサによって実行することもできる。

【０１３３】

ルーチンはステップ２５０で開始し、テクセル・アドレス・ビットＳ［７：０］およびＴ［７：０］が、補間されたＳ，Ｔ座標ビットＳ［７：０］およびＴ［７：０］に等しくなるように事前設定される。このステップで事前設定された値は、後にリセットされない限りそのままの値が維持される。次に、補間されたテクセルが記憶されている特定マップが８以上のマップ番号を持つか否かが判断される（ステップ２６２）。もしそうであれば、そのようなテクセルに関する限りこのルーチンは終了し、テクセル・アドレスに関するビット値は、事前設定された補間されたＳ，Ｔ座標のまま残る。

【０１３４】

マップ番号が８未満であれば、ルーチンはステップ２５４に進み、テクセルがバンク番号１またはバンク番号０のいずれに記憶されているかが決定される。上述のように、ブロック・タグ・ビット［０７］の値を検査することで、どちらのバンクにテクセルが記憶されているかはわかる。

【０１３５】

テクセルがバンク番号１に記憶されている場合、ルーチンはステップ２５６へ進み、一定のテクセル・アドレス・ビットをその事前設定値からリセットする。マップ番号１ないし４を持つマップについては、テクセル・アドレス・ビットＳ［４］＝１、マップ番号１および２を持つマップについては、テクセル・アドレス・ビットＳ［２］＝１とする。テクセルがバンク０に記憶されている場合、ルーチンはステップ２５８へ進み、マップ番号０ないし５を持つマップについては、テクセル・アドレス・ビットＳ［５］＝１とし、マップ番号０ないし３を持つマップについては、テクセル・アドレス・ビットＳ［３］＝１とし、マップ番号０および１を持つマップについては、テクセル・アドレス・ビットＳ［１］＝１とする。

【０１３６】

ステップ２５６、ステップ２５８いずれの後もステップ２６０へ進み、基本マップが８以上のマップ番号を持つか否か判断される。そうであれば、ステップ２６２へ進み、テクセルがバンク０または１のいずれに記憶されているかが判断される。テクセルがバンク１に記憶されている場合、ルーチンはステップ２６４へ進み、マップ番号７を持つマップについては、テクセル・アドレス・ビットＳ［７］＝０とし、マップ番号０ないし６を持つマップについては、テクセル・アドレス・ビットＳ［７：６］＝０：１とする。次に、ルーチンはそのようなテクセルについて終了する。バンク０に記憶されているテクセルの場合、ルーチンはステップ２６６へ進み、マップ番号７を持つマップについては、テクセル・アドレス・ビットＳ［７］＝１とし、マップ番号０ないし６を持つマップについては、テクセル・アドレス・ビットＳ［７：６］＝１：０とする。次に、ルーチンはそのようなテクセルについて終了する。

【０１３７】

基本マップが８以上のマップ番号を持っていない場合、ルーチンはステップ２６８へ進み、基本マップが７に等しいマップ番号を持っているか否か判断される。そうであれば、ステップ２７０へ進み、テクセルがバンク０または１のいずれに記憶されているかが判断される。テクセルがバンク１に記憶されている場合、ルーチンはステップ２７２へ進み、マップ番号７を持つマップについては、テクセル・アドレス・ビットＳ［７］をサブテクスチャＩＤビットＳ［１］の反転に等しく、テクセル・アドレス・ビットＴ［７］をサブテクスチャＩＤビットＴ［１］に等しくし、マップ番号０ないし６を持つマップについては、テクセル・アドレス・ビットＳ［７：６］をサブテクスチャＩＤビットＳ［１］の反転および１にそれぞれ等しくさせ、テクセル・アドレス・ビットＴ［７］をサブテクスチャＩＤビットＴ［１］に等しくする。次に、ルーチンはそのようなテクセルについて終了する。テクセルがバンク０に記憶されている場合、ルーチンはステップ２７４へ進み、マッ

プ番号７を持つマップについては、テクセル・アドレス・ビットＳ［７］をサブテクスチャＩＤビットＳ［１］に等しく、テクセル・アドレス・ビットＴ［７］をサブテクスチャＩＤビットＴ［１］に等しくし、マップ番号０ないし６を持つマップについては、テクセル・アドレス・ビットＳ［７：６］をサブテクスチャＩＤビットＳ［１］および０にそれぞれ等しくさせ、テクセル・アドレス・ビットＴ［７］をサブテクスチャＩＤビットＴ［１］に等しくする。次に、ルーチンはそのようなテクセルについて終了する。

【０１３８】

（ステップ２６０において）テクスチャの基本マップが８以上のマップ番号を持ってなく、かつ、（ステップ２６８において）マップ番号が７に等しくない場合、当然テクスチャの基本マップが６以下のマップ番号を持っているので、ルーチンはステップ２７６へ進み、テクセルがバンク０または１のいずれに記憶されているかが判断される。テクセルがバンク１に記憶されている場合、ルーチンはステップ２７８へ進み、テクセル・アドレス・ビットＳ［7：６］をサブテクスチャＩＤビットＳ［１：０］の反転に等しくセットし、テクセル・アドレス・ビットＴ［7：６］をサブテクスチャＩＤビットＴ［１：０］に等しくセットする。次に、ルーチンはそのようなテクセルについて終了する。テクセルがバンク０に記憶されている場合、ルーチンはステップ２８０へ進み、テクセル・アドレス・ビットＳ［7：６］をサブテクスチャＩＤビットＳ［１：０］に等しくセットし、テクセル・アドレス・ビットＴ［7：６］をサブテクスチャＩＤビットＴ［１：０］に等しくセットする。次に、ルーチンはそのようなテクセルについて終了する。

【０１３９】

VIII．テクスチャ・データ構成の例

以下の例は、本発明 の上述の実施形態に従って、ホスト・コンピュータがテクスチャ・データを構成するプロシージャを説明するものである。特定のアプリケーションに関して、レンダリングされるプリミティブＡがテクスチャＡに対応し、プリミティブＢがテクスチャＢに対応する場合がある。１つの可能性として、ホスト・コンピュータがテクスチャＡを複数のテクスチャ・データ・ブロックに構成し、テクスチャＢをテクスチャＡと同じブロック内の異なるサブテクスチャに構成することがある。ホスト・コンピュータは、プリミティブＡをレンダリングする前にテクスチャＡおよびＢを含むテクスチャ・データ・ブロックをキャッシュ・メモリへダウンロードする。

【０１４０】

代替方法として、ホストはテクスチャＡを複数のテクスチャ・データ・ブロックに構成して、キャッシュ・メモリにテクスチャＡを含むブロックをダウンロードすることもできる。次に、ホスト・コンピュータは、テクスチャＡと同じブロックの異なるサブテクスチャ内で主メモリにテクスチャＢを構成することができる。このような形態では、ホスト・コンピュータは、（図２の）テクスチャ・マッピング・チップ４６の動作を停止させるコマンドを発し、（同じブロック内のテクスチャＡおよびＢを含む）新しく構成されたテクスチャ・データ・ブロックをテクスチャ・マッピング・システムのキャッシュ・メモリへダウンロードする。理解されることであろうが、停止状態が実行されず、新たに構成されたデータが主メモリからテクスチャ・マッピング・システムのキャッシュ・メモリにダウンロードされなかったならば、間違ったテクスチャ・マッピング・データが、プリミティブＢのレンダリングの間アクセスされる可能性がある。なぜならば、プリミティブＢをレンダリングする時、テクスチャＢを含むデータ・ブロックに関する読取りキャッシュ・タグがテクスチャＡを記憶するキャッシュのデータ・ブロックに対応するブロック・タグと一致するためキャッシュ・ディクトリのヒットが発生するからである。しかし、キャッシュのデータ・ブロックは、テクスチャＡに関するテクスチャ・データだけを記憶していてテクスチャＢに関するものは記憶していない。

【０１４１】

IX．３次元プリミティブ・パイプラインのバイパスおよびテクスチャ・マップのダウンロードに関する割り込み方式

上述のように、本発明の１つの機能によって、新しいテクスチャのためのＭＩＰマップは

、３Ｄプリミティブ・データを扱うパイプラインとは別個のデータ経路を経由してテクスチャ・マッピング・ハードウェアのローカル・メモリにダウンロードされる。（図２の）テクスチャ・マッピング基板１２および（図４の）テクスチャ・マッピング・チップ４６は各々、３Ｄプリミティブ・データおよびテクスチャ・データをそれぞれ受け取る独立したポートを有する。３Ｄプリミティブ・データはバス１８経由で集線器チップ３６から受け取られ、一方、テクスチャ・データは２Ｄ加速器チップ３４からバス２４を経由で受け取られる。従って、新しいテクスチャ・データがホスト・コンピュータ１５からテクスチャ・マッピング・チップ４６へダウンロードされる時、フロントエンド基板１０を通過する３Ｄプリミティブ・パイプラインおよびテクスチャ・マッピング・チップ４６はフラッシュされる必要はなく、このため、新しいテクスチャ・データがホスト・コンピュータからテクスチャ・マッピング・チップへダウンロードされる時は必ず３Ｄプリミティブ・パイプラインのフラッシュを必要とする従来技術のテクスチャ・マッピング・システムに比較して帯域幅が増大する。

【０１４２】

３Ｄプリミティブ・パイプラインをバイパスしてテクスチャ・データをダウンロードする独立したデータ経路は、テクスチャ・マッピング基板上のローカル・メモリがキャッシュとして実施される本発明の上述の実施形態と相まって特に有効である。上述のように、新しいテクスチャ・データがキャッシュにダウンロードされる場合、テクスチャに関する一連のＭＩＰマップ全体ではなく、必要とされるＭＩＰマップ部分だけがダウンロードされる。このようにして、３Ｄパイプラインのバイパスは、パイプラインをフラッシュすることなくキャッシュ・ミスの処理を取り扱うことを可能にする。

【０１４３】

上述のように、図３に示される本発明の１つの実施形態において、グラフィックス・システムの特定部分が反復複製して構成されることによって、システムの帯域幅が増加される。テクスチャ・マッピング基板１２は、２つのテクスチャ・マッピング・チップ４６Ａならびに４６Ｂ、および２つのキャッシュ・メモリ４８Ａならびに４８Ｂを備えている。この実施形態では、典型的には２つのテクスチャ・マッピング・チップの両方が同時に同じテクスチャ・データを使用するプリミティブに関して処理を行うので、両方のキャッシュ・メモリ４８は同じテクスチャ・データを常に保持する。従って、ある一方でキャッシュ・ミスが発生すれば必ず両方を更新するので、同じテクスチャ・データが別々の動作で両方のキャッシュへダウンロードされる必要のないことが保証されるため、この実施形態はシステムの帯域幅を節約する。

【０１４４】

図３の複式テクスチャ・マッピング・チップ実施形態において、各キャッシュ・メモリは、ホスト・コンピュータからダウンロードされるテクスチャ・データのみについて更新され、テクスチャ・マッピング・ハードウェアからローカルに書き込まれることはない。従って、キャッシュの１つにおけるキャッシュ・ミスに応答してテクスチャ・データがホスト・コンピュータからダウンロードされる時は必ず両方のキャッシュが新しいテクスチャ・データで更新されることが保証されるため、２つのキャッシュ・メモリの間の整合性が維持される。テクスチャ・マッピング・チップ４６の１つにキャッシュ・ミスが発生し割り込みが生成されると、ダウンロードされたテクスチャ・データを用いて両方のキャッシュ・メモリを更新することができるように、両方のテクスチャ・マッピング・チップ４６が停止させられる。このように、いずれかのテクスチャ・マッピング・チップから発せられるキャッシュ・ミス信号に応答して、テクスチャ・マッピング・チップの各々は動作を停止する。更に、本発明は、異なるキャッシュ・ブロックに対する２つのテクスチャ・マッピング・チップ４６の同時キャッシュ・ミスをサポートし、キャッシュ・ミスに応答して、新しいテクスチャ・データ・ブロックの両方を両方のキャッシュにダウンロードする。

【０１４５】

図２に示されるように、３Ｄプリミティブ・パイプラインのバイパスは、２Ｄ加速器チッ

プ３４を通過する２Ｄプリミティブ・パイプラインを使用してテクスチャ・データをダウンロードすることによって達成される。テクスチャ・マッピング・チップ４６へテクスチャ・データをダウンロードするデータ経路は、３Ｄプリミティブ・パイプラインをバイパスする点は同じとしても、多数の形態で実施できることは理解されるべきであろう。例えば、ホスト・コンピュータからテクスチャ・マッピング基板へテクスチャ・データをダウンロードする１つの専用データ経路を備えることも可能である。

【０１４６】

本発明のグラフィックス・システムのホスト・コンピュータが、同時に動作する複数のプロセスを持ち、プロセスが割り込まれないように一定のシステム資源をロックすることを可能にする何らかの方式を提供するＵＮＩＸのようなオペレーティング・システムを使うこともある。ロッキング方式の使用によって、特定のハードウェア資源を使用するプロセスは、それらの資源のロックをはずすまでプロセスはスワップアウトされないことを保証することができる。

【０１４７】

本発明の１つの実施形態において、急速ロック(fast lock)と遅速ロック(slowlock)という２つのタイプのロックがプロセスによる使用のため用意される。急速ロックが使われると、スワップインされるプロセスは、適切なハードウエア資源を検査して、そのプロセスがそれらの資源を使用する最後のプロセスであったか否か判断する。そうであれば、プロセスはハードウェア資源状態を復元することなく継続する。しかし、最後のものでない場合、遅速ロックが要求され、そのプロセスが最後にスワップアウトされた時の状態にハードウエア資源が復元される。同様の結果を達成することができる多数の代替方法があることは理解されるべきであろう。

【０１４８】

３Ｄプリミティブをレンダリングする間２Ｄプリミティブ・パイプラインを使用してテクスチャ・データをダウンロードする本発明の実施形態において、２Ｄおよび３Ｄプロセスは同時に動かされない。ホスト・コンピュータのオペレーティング・システムによって提供されるロッキング方式の使用によって、３Ｄパイプラインが空でない限り２Ｄプロセスは開始しないこと、および、２Ｄパイプラインが空でない限り３Ｄプロセスは開始しないことを保証することによって、上記制約が守られる。３Ｄプロセスが開始する時それはロックをかけ、先行プロセスが２Ｄであった場合２Ｄパイプラインが空になるまで開始せず待機する。同様に、２Ｄプロセスが開始する時それはロックをかけ、先行プロセスが３Ｄであった場合３Ｄパイプラインが空になるまで開始せず待機する。

【０１４９】

プロセスによって、３Ｄおよび２Ｄ動作の両方を実行し、かつ、遅速ロックを放棄することなく３Ｄプリミティブと２Ｄプリミティブの間の切り換えを行うこともできる。そのようなプロセスは、また、３Ｄパイプラインがハードウェアへ２Ｄプリミティブ・データをダウンロードする前に空であることを確認し、同様に２Ｄパイプラインがハードウェアへ３Ｄプリミティブ・データをダウンロードする前に空であることを確認する方式を実行する。この方式を達成するため、２Ｄおよび３Ｄプリミティブ・パイプラインの各々が空であるか否かを標示するレジスタ状態ビットが用意されることもある。２Ｄおよび３Ｄプリミティブ・データを使うプロセスは、２Ｄおよび３Ｄプリミティブ・データの間の切り換えを行う前にパイプラインがが空であることを確認するため、この状態レジスタを読み取る。

【０１５０】

上記の本発明の実施形態はキャッシュとして実施されるテクスチャ・マッピング基板上のローカル・メモリを含むが、本発明はそれに限定されない点は理解されるべきであろう。テクスチャ・マッピング基板上のローカル・メモリがキャッシュでなく、プリミティブがレンダリングされる時テクスチャ・マッピング・データがローカル・メモリで使用可能であるようにするため、３Ｄプリミティブ・パイプラインとは別の経路を経由して、プリミティブがレンダリングされる前に、プリミティブをレンダリングするために必要とされる

テクスチャ・マッピング・データの各ブロックがダウンロードされることを保証するようなその他の技術を使用するテクスチャ・マッピング・システムを実施することもできる。

【０１５１】

更に、ホスト・コンピュータによるローカル・メモリのデータ・ブロックの更新のための割り込みを生成する本発明の方式は、多くの他のアプリケーションについて実施することができ、テクスチャ・マッピング・ハードウェア・システムにおける使用に限定されない点は理解されなければならない。この方式は、処理されるべきデータ・ブロックを記憶する主メモリを備えるホスト・コンピュータ、および処理されるべきデータ・ブロックを記憶するローカル・メモリを備えるデータ処理ハードウェアを含むデータ処理システムにとって利点がある。

【０１５２】

X. キャッシュ・ブロック置き換え方式

上述のように、キャッシュにないテクスチャ・データ・ブロックについてキャッシュ・ミスが発生すると、ホスト・コンピュータは、要求されたテクスチャ・データのブロックを（図２の）キャッシュ４８へダウンロードする。キャッシュがいっぱいの時キャッシュ・ミスが発生すると、キャッシュ・ブロックの１つが、新しくダウンロードされるテクスチャ・データ・ブロックと置き換えられる。本発明の１つの実施形態において、最も以前に使用されたキャッシュ・ブロックが判定され、キャッシュの活動的ブロックを維持するための置き換え用としてそのブロックが選択される。ホスト・コンピュータ１５のメモリ１７に記憶されホスト・コンピュータのプロセッサ１９上で動くソフトウェア・ルーチンによって、置き換えられるべきキャッシュ・ブロックが決定される。テクスチャ・マッピング・チップ４６は、置き換えられるべきキャッシュ・ブロックを決定するソフトウェア・ルーチンをサポートする２組のレジスタを含む。キャッシュ・ミスが発生すると、これらのレジスタが、３Ｄバイパス・データ経路を経由してホスト・コンピュータによって読み取られ、置き換えられるべきキャッシュ・ブロックを決定する際に使用される。

【０１５３】

レジスタの第１の組は、バンク０およびキャッシュ４８の１つにそれぞれ対応するように配置され、最も最近使用された２つの３２ビット・レジスタＭＲＵ０およびＭＲＵ１（集合的にＭＲＵ、Most Recently Usedと呼ぶ）を含む。これらのレジスタの各ビットは、その対応するキャッシュ・バンク内に含まれる３２個のキャッシュ・ブロックの１つに対応する。あるブロックについてキャッシュ・ヒットが発生するたびごとに、最も最近使用されたレジスタがキャッシュ・ヒットを蓄積するように、ＭＲＵ０またはＭＲＵ１における対応するビットがセットされる。

【０１５４】

レジスタの第２の組は、バンク０およびキャッシュの１つにそれぞれ対応するように配置され、現在使用中の３２ビット・レジスタＣＵ０およびＣＵ１（集合的にＣＵ、Currently Usedと呼ぶ）を含む。ＣＵ０またはＣＵ１の１つのビットがセットされている場合、それは、対応するキャッシュ・ブロックがキャッシュのミニディレクトリに現在存在し、従って置き換えられるべきものではないことを標示する。キャッシュのミニディレクトリの詳細は後述する。

【０１５５】

キャッシュ・ミスが発生し、ホスト・コンピュータへの割り込みが起きると、図１９の流れ図によって示されるソフトウェア・ルーチンが、ホスト・コンピュータのプロセッサ１９によって実行され、ダウンロードされるように要求されたテクスチャ・データを含むブロックとどのキャッシュ・ブロックを置き換えるべきかが決定される。ソフトウェア・ルーチンは、置き換えルーチンを実行する際に使用される２つの６４ビット状態ワード（すなわちＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥおよびＢＬＯＣＫＳ＿ＢＵＳＹ）を保持する。これらの状態ワードの６４状態ビットの各々は、６４個のキャッシュ・ブロックの１つに対応する。

【０１５６】

図１９のステップ３００において、それぞれ初期的に置き換え可能状態にあることを各ビットが標示するようにＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥが初期化される、ステップ３０２において、ルーチンはキャッシュ・ミス割り込みが受け取られたか否かを判断するため継続的に検査し、割り込みが検出されると、ステップ３０４へ進み、３Ｄバイパス・データ経路を経由してレジスタＭＲＵおよびＣＵを読み取る。上述のように、２つのテクスチャ・マッピング・チップが使われている本発明の実施形態においては、２つのチップのキャッシュ・メモリは、同じテクスチャ・データを常時保持する。従って、システムが２つのテクスチャ・マッピング・チップ４６を含むならば、両方のチップのレジスタＭＲＵおよびＣＵが読み取られ、ルーチンは、置き換え用として、いずれかのテクスチャ・マッピング・チップにおいて最も以前に使用されたキャッシュを選択することができる。ステップ３０６において、ＭＲＵまたはＣＵでオンにされているビットに対応するＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのビットをオフにする。２つ以上のテクスチャ・マッピング・チップが使用される実施形態では、ＭＲＵとＣＵの論理和を使用して、オフにすべきＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのビットを決定する。

【０１５７】

ステップ３０８で、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのいずれかのビットがオンにされているかという判断が行われ、少くとも１つがオンであれば、ルーチンはステップ３１０へ進み、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオンにされているビットの数が所定のしきい値より少ないか否かの判断が行われる。このステップは、複数のキャッシュ・ミスに関するキャッシュ・ブロック使用実績の維持を援助し、（後述される）将来のキャッシュ・ミス割り込みの適切な処理を確実にするため実行される。ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオンにされているビットの数が所定のしきい値より少ない場合、ルーチンはステップ３１２へ進み、ＭＲＵのビットのすべてがオフにされる。この結果、ＭＲＵは、現在処理されているキャッシュ・ミスの後に発生するキャッシュ・ヒットについてのみキャッシュ・ヒットを累積し始める。本発明の１つの実施形態では、上記しきい値は、ＢＬＯＣＫＳ_ＴＯ＿ＵＳＥ中オンにされた１１ビットと設定され、これは１１個のキャッシュ・ブロックが置き換えに使用できることを標示する。

【０１５８】

ステップ３１２でＭＲＵがクリアされた後、または、ステップ３１０でＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオンのビット数が所定のしきい値未満であると判断された後、ルーチンはステップ３１４へ進み、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥでオンにセットされたビットの１つが、ダウンロードされるべきテクスチャ・データの新しいブロックとの置き換え用として選択される。ステップ３１４において置き換えのため選択されたブロックが、図２１を参照して後に説明する方法で、テクスチャ・データの新しいブロックによって置き換えられる。ステップ３１４において、置き換えられるブロックが選択された後、ルーチンはステップ３０２へ戻り、別のキャッシュ・ミス割り込みを待つ。

【０１５９】

ステップ３０８でＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのビットにオンにされているビットがないと判断されると、ルーチンはステップ３１６へ進み、ＢＬＯＣＫＳ＿ＢＵＳＹが、ＭＲＵとＣＵの論理和と等しく設定される。従って、ＢＬＯＣＫＳ＿ＢＵＳＹでセットされているビットだけが、ＭＲＵまたはＣＵレジスタのどちらかにセットされているビットに対応する。その後、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥは、ＢＬＯＣＫＳ＿ＢＵＳＹの補数に等しく設定される。このような形態では、ＭＲＵおよびＣＵでオンにされ、置き換え用に選択されるべきでないことを標示するビットに対応するビットを除いて、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥの各ビットはオンにされる。

【０１６０】

ステップ３１６でＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥがＢＬＯＣＫＳ＿ＢＵＳＹの補数と等しく設定された後、ルーチンはステップ３１８へ進み、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのいずれかのビットがオンになっているか判断される。ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥの少なくとも１つのビットがオンとなっていれば、ルーチンはステップ３１０ないしステップ３１４へ

進み、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオンにされたビットの数がしきい値を下まわっていればＭＲＵのすべてのビットがオフにされ、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオンにされたビットの１つが置き換えのため上述の方法で選択される。

【０１６１】

オンとなっているビットがＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥに１つもない場合、ルーチンはステップ３２０へ進み、３つのアクションが取られる。第１に、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオンにされたビットの数が所定のしきい値より少ないのでＭＲＵのすべてのビットをオフにする。第２に、ＢＬＯＣＫＳ＿ＢＵＳＹは、ＣＵレジスタと等しい値に設定される。上述のように、各ＣＵレジスタは、対応するキャッシュ・ミニディレクトリに現在維持されているキャッシュ・ブロックを示しているので、置き換えられてはならない。複数のテクスチャ・マッピング・チップが使われている場合、ＢＬＯＣＫＳ＿ＢＵＳＹは、ＣＵレジスタの論理和と等しく設定される。最後に、ＢＬＯＣＫ＿ＴＯ＿ＵＳＥが、ＢＬＯＣＫＳ＿ＢＵＳＹの補数に等しく設定される。その結果、テクスチャ・マッピング・チップの１つのキャッシュ・ミニディレクトリに現在維持されているデータ・ブロックのビットに対応するビットを除いて、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥの各ビットはオンにされる。ルーチンはステップ３１４へ進み、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥの中のオンにされたビットの１つが置き換え用として選択される。このように、ミニディレクトリにあるもの以外のキャッシュのブロックのいずれかを置き換え用として選択することができる。

【０１６２】

図１９に示される本発明の実施形態は、キャッシュ・ミスが発生する時ＬＲＵ法（すなわち最も以前に使用されたものを対象とする方式）を用いてキャッシュ・ブロックを置き換える置き換え方式を使う。本発明の有効範囲を逸脱することなくこの方式に種々の変更を加えることができることは理解されなければならない。例えば、図１９に示される実施形態において、ＭＲＵハードウェア・レジスタが、複数のキャッシュ・ミスが潜在的に含まれる可能性のある一定時間にわたってキャッシュ・ヒットを収集するために使用され、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオンにされたビットの数が所定のしきい値を下回った時にのみＭＲＵレジスタをクリアする。更にまた、ソフトウエア状態ワードのＢＬＯＣＫＳ＿ＢＵＳＹは、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのすべてのビットがオフであることが判明した場合にのみステップ３１６またはステップ３２０で更新される。代わりの方法として、キャッシュ・ミス割り込みが受け取られる度毎にＭＲＵを使用してＢＬＯＣＫＳ＿ＢＵＳＹを更新することによって置き換えを実行することができる。この形態で、複数のキャッシュ・ミスが潜在的に含まれる可能性のある一定時間にわたってキャッシュ・ヒットの実績を累積するためソフトウェア状態ワードＢＬＯＣＫＳ＿ＢＵＳＹを使用することができ、そして、ハードウェア・レジスタＭＲＵを、ミスとミスの間のヒットを累積するため使用することができる。

【０１６３】

更に、ＭＲＵをクリアする効果を持つＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオン・ビットのしきい値が上述の実施形態において１１ブロックについてオンにセットされているというものであったが、この数値は明らかに変更することができることは理解されるべきであろう。このしきい値は、ルーチンが、ステップ３０８におて、ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのビットがどれもオンになってない状態に遭遇する回数に影響を及ぼす。この状態は最も最近使用されたキャッシュ・ブロックについて（ステップ３１６またはステップ３２０で）ＢＬＯＣＫ＿ＴＯ＿ＵＳＥの更新につながるのでこの状態を回避することが望ましい。ＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥにおいてオンにセットされるビットが、複数のキャッシュ・ミスの処理を通して使用されたことのないブロックを反映するように、高分解能を備えることが望ましい。従って、ＭＲＵクリアにつながるＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオン・ビットのしきい値を制御することによって、ステップ３０８においてＢＬＯＣＫＳ＿ＴＯ＿ＵＳＥのオン・ビット数を判断する本ルーチンの回数が最小限に抑えられ、最も以前に使用されたキャッシュ・ブロックを決定する場合の望ましい分解能が与えられる。

【０１６４】

ホスト・コンピュータ上で実行されるソフトウエア・ルーチンによって実施される上述の置き換え方式がキャッシュ・メモリに関する使用に限定されるものではないことは理解されるべきであろう。ローカル・メモリが、処理されるデータ・ブロックを含み、追加データ・ブロックがホスト・コンピュータからローカル・メモリへダウンロードされる時にローカル・メモリ内のデータ・ブロックが置き換えられるようなデータ処理システムのどのようなシステムにおいても上記置き換えルーチンを使用することが可能である。

【０１６５】

XI．キャッシュ動作の実行禁止

本発明の１つの実施形態において、プリミティブのレンダリングの間テクスチャ・データが必要とされる前に任意の３Ｄプリミティブに関するテクスチャ・データがメモリ４８にダウンロードされるようにするため、キャッシュ・ミスを実行禁止(disable、ディスエーブル)にすることによってテクスチャ・マッピング基板上のローカル・メモリ４８のキャッシュ動作を実行禁止にする機能が備えられる。テクスチャ・マッピング・チップ４６の各々は、キャッシュとしてのローカル・メモリの動作が実行可能状態であることを標示する状態ビットを含む。この状態ビットがオンになっていれば、キャッシュ・ミスの発生が、ホスト・コンピュータの割り込みおよびテクスチャ・マッピング・チップの停止を引き起こす。しかし、この状態ビットがオフとなっていれば、テクスチャ・マッピング基板のローカル・メモリ４８はキャッシュとして機能せず、いかなるプリミティブに対するテクスチャ・データも、キャッシュ・ミスが発生しないように、該プリミティブによって必要とされる前にメモリ４８にダウンロードされる。本発明の１つの実施形態において、キャッシュとしてのローカル・メモリの動作が実行禁止にされ、テクスチャ・データが、テクスチャ・マッピング基板上のローカル・メモリに３Ｄプリミティブ・パイプラインを経由してダウンロードされ、テクスチャ・データと対応する３Ｄプリミティブ・データの同期が取られる。

【０１６６】

XII．キャッシュ・ミスに応答してテクスチャ・データをダウンロードする方式を支援するテクセル・ポート・レジスタ

上述のように、（図２の）テクスチャ・マッピング・チップ４６は、ホスト・コンピュータ１５からダウンロードされるテクスチャ・データを受け取るため使用される（図４の）テクセル・ポート９２を含む。テクセル・ポートは、テクセル・データのダウンロードを支援する多数のレジスタを含む。それらのレジスタのいくつかは、上述のレジスタＭＲＵとＣＵを含む。その他のテクセル・ポート・レジスタには、コマンド・レジスタ、状態レジスタ、テクセル・データ・レジスタ、ディレクトリ・タグ・レジスタ、キャッシュ・アドレス・レジスタおよびパイプ・タグ・レジスタが含まれる。それぞれの機能を以下に説明する。

【０１６７】

３Ｄプリミティブ・パイプライン経由でこれらレジスタへの書き込みを行えるように、テクセル・ポート・レジスタへのアクセスが提供される。３Ｄパイプラインが使用中の時でも、単にパイプラインに置かれているレジスタ書き込み用データを用いて、テクセル・ポート・レジスタに書き込むことができる。さらに、テクセル・ポート・レジスタは、２４ビット・バス２４（図２）上に備わる３Ｄパイプライン・バイパスを経由してアクセスすることも可能である。テクセル・ポート・レジスタをアクセスする場合、バス２４の８ビットが、どのテクセル・ポート・レジスタの読み取りまたは書き込みを行うべきかを指定するレジスタ・アドレスとして使用され、データがテクセル・ポート・レジスタに書き込まれる時、上記バス２４のその他の１６ビットがデータを提供する。

【０１６８】

テクセル・ポート・レジスタの構成は図２０に示されている通りである。本発明の１つの実施形態において、テクセル・ポート・レジスタの各々のビット数は３２ビットである（ただし一部のレジスタの多数のビットは未使用）。

【０１６９】

A．テクセル・コマンド・レジスタ

テクセル・コマンド・レジスタは、詳細は後述するが、キャッシュ・ミスを取り扱うホスト・コンピュータのソフトウェア・ルーチンによって使用される多数のビットを含む。停止ビット３５０は、ソフトウェア割込み処理ルーチンによってセットされ、テクスチャ・マッピング・チップに対してその動作を停止するように指示する。上述のように、２つのテクスチャ・マッピング・チップが備わる本発明の実施形態においては、両方のキャッシュが整合性を保つようにどちらのキャッシュ・ミスが発生しても両方のテクスチャ・マッピング・チップは同じテクスチャ・データで更新される。どちらかのテクスチャ・マッピング・チップのキャッシュ・ミスが受け取られると、それぞれのテクセル・コマンド・レジスタの停止ビット３５０をセットすることによって両方のテクスチャ・マッピング・チップが停止させられる。キャッシュ・ミスに応答して新しいテクスチャ・データがホスト・コンピュータからダウンロードされた後、キャッシュ・ミスを扱うソフトウェア・ルーチンがコマンド・レジスタに書き込みを行うことによって停止ビットがクリアされる。

【０１７０】

割り込み可能ビット３５２は、それがオンの時、キャッシュ・ミス発生の際テクセル・ポートからの割り込みを可能にする。このビットは、テクスチャ・マッピング基板１２（図２）上のローカル・メモリ４８をキャッシュとして動作させない上述の機能を提供する場合にはオフにセットされる。

【０１７１】

書込みＬｏｋｉ０３５４ならびに書込みＬｏｋｉ１ビット３５６は、テクセル・ポート・レジスタに関する書込み可能／禁止を制御する。Ｌｏｋｉは、テクスチャ・マッピング・チップ４６を識別するために使用される短縮形名称である。２つのチップが使用される本発明の実施形態においては、それら２つのチップはＬｏｋｉ０ならびにＬｏｋｉ１と呼ばれる。単一のテクスチャ・マッピング・チップが使われ時は、そのチップはＬｏｋｉ０として識別される。テクセル・ポート・レジスタのいずれかに対する書込みを行うコマンドがバス２４経由で受け取られると、各テクスチャ・マッピング・チップ（すなわちＬｏｋｉ０ならびにＬｏｋｉ１）はその書込みビットが使用可能とされているかを判断するためそのコマンド・レジスタを検査し、使用可能であれば、受け取られた書込みコマンドに従ってそのテクセル・ポート・レジスタを更新する。従って、書込みＬｏｋｉ０３５４ならびに書込みＬｏｋｉ１ビット３５６の値を制御することによって、ホスト・コンピュータ上で動くソフトウェア・ルーチンが、２つのテクスチャ・マッピング・チップのテクセル・ポート・レジスタに、別々に、あるいはまた同時に書き込むことができる。

【０１７２】

Ｌｏｋｉ読取りビット３５８は、テクスチャ・マッピング・チップの１つのテクセル・ポート・レジスタの読取りを可能にする。テクセル・ポート・レジスタを読み取るコマンドがテクセル・バス２４経由で受け取られると、一時点において、テクスチャ・マッピング・チップ（複数）の中の１つだけが応答して、バス上へそのテクセル・ポート・レジスタの内容を送出する。２つのテクスチャ・マッピング・チップが備わる実施形態において、各々は、該チップがＬｏｋｉ０かＬｏｋｉ１のどちらかであることを示すハードウエア・ピンを備える場合がある。Ｌｏｋｉ読取りビットがソフトウェアによってセットされる場合、それはＬｏｋｉ１からの読取りが可能とされていることを示し、Ｌｏｋｉ読取りビットがセットされていない場合、それはＬｏｋｉ０からの読取りが可能とされていることを示す。テクセル・コマンド・レジスタの形式が、両方のテクスチャ・マッピング・チップ（Ｌｏｋｉ０とＬｏｋｉ１）に同じデータで同時に書き込むことを可能にするため、レジスタへの書き込みのため１回の書込みサイクルだけでよい点は前述の記載から理解されることであろう。

【０１７３】

B．テクセル状態レジスタ

テクセル・ポート状態レジスタは、オンにセットされる時、システムが２つのテクスチャ・マッピング・チップを含むことを示す複数Ｌｏｋｉビット３６０を含む。コマンド・レ

ジスタのビット３５２がオンの時は必ず割込み可能ビット３６２はオンにされ、テクスチャ・マッピング・チップのローカル・メモリはキャッシュとして機能し、キャッシュにないテクスチャ・データが必要とされる時、ホスト・コンピュータに割り込むためのキャッシュ・ミスを生成することを標示する。このビットは、コマンド・レジスタと共に状態レジスタにも含められるので、状態レジスタを単に読むだけでテクセル・ポートの状態を読み取ることができる。

【０１７４】

割り込みがテクスチャ・マッピング・チップから発生し、該チップが新たなテクスチャ・データがダウンロードされるのを待つ時、割り込み有効ビット３６４がオンにされる。パイプ・タグ・レジスタ（後述）に記憶されているキャッシュ・ミスを発生したキャッシュ読取りタグと一致するキャッシュ・タグを用いてキャッシュ・ディレクトリ・タグ・レジスタ（後述）に書き込みが行われると、このビットはクリアされる。

【０１７５】

状態レジスタは、キャッシュ・ミスが発生する時テクスチャ・マッピングの停止を支援する２つのビットを含む。停止可能ビット３６８は、コマンド・レジスタの停止ビット３５０がセットまたはクリアされる毎に、ホスト・コンピュータ上のソフトウェア・ルーチンによってそれぞれセットまたはクリアされ、当ビットがオンの時動作を停止するようにテクスチャ・マッピング・チップに指示する。このビットがコマンド・レジスタと共に状態レジスタに備えられるので、テクスチャ・マッピング・チップの状態が単一のレジスタに記憶される。キャッシュ・ミスが発生し、キャッシュ・ディクトリが新たなデータがダウンロードされるのを待つ時、割込み有効ビット３６４がテクスチャ・マッピング・チップのハードウエアによってオンにされる。キャッシュ・ミスが発生したブロック・タグと一致するキャッシュ・タグを用いてキャッシュ・ディレクトリ・タグ・レジスタ（後述）に書き込みが行われると、このビットはクリアされる。

【０１７６】

C．パイプ・タグ・レジスタ

パイプ・タグ・レジスタは、テクスチャ・マッピング・チップのパイプラインによってインデックス付けされた最後のブロック・タグを記憶する。キャッシュ・ミスが発生すると、パイプ・タグ・レジスタは、キャッシュ・ミスが発生したブロック・タグ３７０を記憶する。テクセル・ポート・バス２４を経由してパイプ・タグ・レジスタを読み取ることによって、キャッシュ・ミス割り込みに応答するソフトウェアが、キャッシュ・ミスに応答してダウンロードされなければならないキャッシュ・ブロックのタグを決定することができる。

【０１７７】

D．テクセル・データ・レジスタ

テクセル・データレジスタは、キャッシュ・ミスが発生した時テクスチャ・データをキャッシュ４８へダウンロードするために使用される。上述のように、各テクセルは、αを示す１バイト３７２、赤の値を表す１バイト３７４、緑の値を表す１バイト３７６および青の値を表す１バイト３７８を含む３２ビットのデータによって表される。

【０１７８】

E．テクセル・キャッシュ・アドレス・レジスタ

テクセル・キャッシュ・アドレス・レジスタは、キャッシュへテクセル・データを書き込み、ブロック・タグをキャッシュ・ディレクトリに書き込むために使用される。上述のように、キャッシュは、各ブロックが２５６×２５６アレイのテクセルを含む６４ブロックのテクスチャ・データを記憶する。テクセル・キャッシュ・アドレス・レジスタは、キャッシュにおける６４ブロックのうちで読み込みまたは書き込みの対象となる特定の１つのブロックを識別する６ビットのブロック・インデックス・フィールド３８０を含む。加えて、このレジスタは、上記ブロック・インデックス・フィールドで識別されるブロック内で読み込みまたは書き込みの対象となる特定のテクセル・アドレスを識別する１６ビットのブロック・アドレス・フィールド３８２を含む。キャッシュ・ミスに応答してテクスチ

ャ・メモリにデータがダウンロードされる時、ブロック・インデックスが、上述のＬＲＵ（すなわち最も以前に使用されたものを選択する）置き換え法を使用してソフトウェア・ルーチンによってセットされ、ブロック・アドレス・フィールド３８２が最初のテクセルをブロックに書き込むためゼロに初期化される。キャッシュ・アドレス・レジスタは、テクセル・データ・レジスタがアクセスされると必ずブロック・アドレス・フィールド３８２を自動的に増分する。このように、キャッシュ・ブロック内のブロック・アドレスのすべてにわたってブロック・アドレス・フィールドが増分され、テクセル・データの新しいブロックがキャッシュに書き込まれる。

【０１７９】

F．テクセル・ディレクトリ・タグ・レジスタ

テクセル・ディレクトリ・タグ・レジスタは、キャッシュ・ブロック・タグを表す２３ビットのブロック・タグ・フィールド３８４を含み、キャッシュ・アドレス・レジスタのブロック・インデックス・フィールド３８０によって定義されるキャッシュ・ディレクトリ・エントリを書き込むために使用される。上述のように、キャッシュ・ブロック・タグの２３ビットは、８ビットのテクスチャＩＤ、７ビットのＳ座標、７ビットのＴ座標、および、ブロック・タグに対応するテクスチャ・データのブロックによって表されるマップのマップ番号を識別する付加ビット（１ビット）を表す。キャッシュ・ミスに応答してテクスチャ・データの新しいブロックがホスト・コンピュータからダウンロードされる時、そのブロック・タグは、テクセル・バス２４経由でディレクトリ・タグ・レジスタにロードされる。ディレクトリ・タグ・レジスタから、ブロック・タグは、キャッシュ・アドレス・レジスタのブロック・インデックス・フィールド３８０によって識別されるキャッシュ・ディレクトリ・エントリに書き込まれる。上述のように、キャッシュ・ミスを発生させたパイプ・タグ・レジスタのタグと一致するディレクトリ・タグ・レジスタにブロック・タグが書き込まれると、キャッシュ・ミス割り込みはクリアされる。

【０１８０】

XIII．キャッシュ・ミス割り込みを取り扱うソフトウエア・ルーチン

上述の記載から理解されるように、テクセル・ポート・レジスタは、キャッシュ・ミス割り込みに応じて必要なテクスチャ・データをダウンロードする機能を遂行するホスト・コンピュータ１５上のソフトウェア・ルーチンによって使用される。図２１は、このソフトウェア・ルーチンの流れ図を示す。ステップ４００において、Ｌｏｋｉ０およびＬｏｋｉ１両方のテクセル・コマンド・レジスタにおける停止ビット３５０がセットされる。次にルーチンはステップ４０２へ進み、テクセル状態レジスタの停止ビット３６８を読み、両方のＬｏｋｉが停止したか否か判断する。Ｌｏｋｉ０およびＬｏｋｉ１が停止したことが確認できるまでルーチンは両方の状態レジスタの読み取りを継続する。両方が停止したと確認すると、ルーチンはステップ４０４へ進む。システムがただ１つのテクスチャ・マッピング・チップ４６（すなわち、Ｌｏｋｉ０）を含む場合、Ｌｏｋｉ０は、テクセル・バス２４上にＬｏｋｉ０のテクセル・ポート・レジスタの内容を送出することによってＬｏｋｉ１のテクセル・ポート・レジスタを読み取る要求に応答する。このように、ソフトウェア・ルーチンがステップ４０２で両方のＬｏｋｉが停止したか否か判断する時、Ｌｏｋｉ０が、Ｌｏｋｉ０が停止した場合のようにＬｏｋｉ１の読み取り要求に応答するので、処理はステップ４０４へ進む。

【０１８１】

ステップ４０４で、Ｌｏｋｉ０がキャッシュ・ミスによって割り込みを起こした否かを調べるためＬｏｋｉ０のテクセル状態レジスタ中の割り込み有効ビット３６４が読み込まれる。もし割り込みが発生していれば、ルーチンはステップ４０６へ進み、Ｌｏｋｉ０のパイプ・タグ・レジスタが読み取られ、キャッシュ・ミスを発生させたテクスチャ・データのブロックのブロック・タグを識別する。ソフトウェア・ルーチンはこのブロック・タグを使用して、ホスト・コンピュータのメモリ１７（図２）に記憶されているテクスチャ・データの対応するブロックにアクセスし、ステップ４０８で、キャッシュのどのブロックを、ダウンロードされるテクスチャ・データの新しいブロックと置き換えるべきかを決定

する。この決定は、図19に関連して上述したＬＲＵ法を使用して実行される。

【０１８２】

上述のように、システムが２つのテクスチャ・マッピング・チップを含む場合、各々におけるキャッシュは全く同じエントリを持つように維持される。従って、テクスチャ・マッピング・チップの１つに発生したキャッシュ・ミスに応答してホスト・コンピュータからダウンロードされるテクスチャ・データは、両方のチップのキャッシュに書き込まれる。かくして、置き換えられるキャッシュ・ブロックが識別されたならば、ルーチンはステップ４１０へ進み、Ｌｏｋｉ０および（Ｌｏｋｉ１が存在する場合）Ｌｏｋｉ１におけるキャッシュ・アドレス・レジスタが、ステップ４０８で決定されたブロック・インデックスを用いて書き込まれる。ステップ４１２で、キャッシュ・ミスに応答してテクスチャ・キャッシュにダウンロードされるべきテクスチャ・データのブロックのブロック・タグを用いてディレクトリ・タグ・レジスタに書き込みが行われ、ステップ４１４において、テクスチャ・データがテクセル・データ・レジスタに書き込まれる。このように、本ルーチンは、キャッシュ・ミスのあったテクスチャ・データのブロックをダウンロードしてキャッシュにこのデータ・ブロックを書くことによってキャッシュ・ミスに応答する。

【０１８３】

ステップ４０６ないしステップ４１４においてテクスチャ・データのブロックがＬｏｋｉ０およびＬｏｋｉ１にダウンロードされたあと、あるいは、ステップ４０４でＬｏｋｉ０が割り込まれなかったと判断されたなら、ルーチンはステップ４１６へ進み、キャッシュ・ミスが発生したのがＬｏｋｉ１であることを示すＬｏｋｉ１状態レジスタの割り込み有効ビット３６４を調べる。上述のように、システムが１つのテクスチャ・マッピング・チップだけを含む場合、Ｌｏｋｉ０がＬｏｋｉ１テクセル・ポート・レジスタの読取りに応答する。Ｌｏｋｉ０がＬｏｋｉ１の状態レジスタの読取りに応答する場合、その割り込み有効ビット３６４はマスクされているので、ソウフトウエア・ルーチンはステップ４１６においてＬｏｋｉ１が割り込みを起こしていないと判断する。このマスキングによって、Ｌｏｋｉ０からの割り込みを再プロセスしてテクスチャ・データをダウンローすることが防止される。従って、ただ１つのテクスチャ・マッピング・チップだけが備わるシステムにおいては、ルーチンはステップ４１６でＬｏｋｉ１に割り込みが発生していないと判断し、ステップ４１８へ進み、Ｌｏｋｉ０のコマンド・レジスタの停止ビット３５０がオフにされ、これによって、テクスチャ・マッピング・チップがそのパイプラインにおけるプリミティブ処理を続行することが可能とされる。

【０１８４】

システムが２つのテクスチャ・マッピング・チップを含む場合、ルーチンはステップ４１６でＬｏｋｉ１が割り込みを起こしたか否かを判断し、起こしていなければ、ステップ４１８へ直接進み、両方のテクスチャ・マッピング・チップの停止ビットをオフにして両方のチップがプリミティブ処理を続行することを可能にする。しかし、ステップ４１６で、Ｌｏｋｉ１がキャッシュ・ミスに応答して割り込みを起こしたと判断すれば、ルーチンは、ステップ４２０ないしステップ４２４へ進み、Ｌｏｋｉ０の割り込みを処理するステップ４０６ないしステップ４１４と同様の方法でＬｏｋｉ１の割り込みを処理する。次にルーチンはステップ４１８へ進み両方のテクスチャ・マッピング・チップの停止ビットをオフにする。

【０１８５】

２つのテクスチャ・マッピング・チップを備えるシステムにおいては、両方のチップが同じブロック・タグについてまたは異なるブロック・タグについて同時にキャッシュ・ミス割り込みを生成することができる点は理解されるべきであろう。両方のテクスチャ・マッピング・チップが、同じブロック・タグについてキャッシュ・ミス割り込みを生成する場合、割り込みは、ステップ４００ないしステップ４１４で処理される。従って、ステップ４１２においてキャッシュ・ミスのあったブロック・タグを両方のＬｏｋｉのディレクトリ・タグ・レジスタに書き込むことによってＬｏｋｉ１からの割り込みがクリアされるので、ステップ４１６においてルーチンはＬｏｋｉ１の割り込みを検出しない。このように

して、図２１に示される方法が、いずれのテクスチャ・マッピング・チップからの割り込みに対してもそれぞれ個別的に、あるいは、両方の割り込みに同時に、応答することができる。

【０１８６】

XIV．キャッシュ・ミニディレクトリおよび主ディレクトリ

上述のように、本発明の１つの実施形態において、キャッシュは、１ブロックが２５６×２５６テクセル・データからなる６４ブロックのテクセル・データ、および２３ビットのブロック・タグを持つ６４のエントリを含む完全連想型キャッシュ・ディレクトリを含む。本発明が３線形補間モードで動作する場合、８個のテクセル読取りが、ある１つのピクセルについて所望のテクセル・データを決定するために実行される。それら８個のテクセルのうち、ある１つのマップにおける４個のテクセルが１回の読み取り動作で同時に読み出され、別のマップの４個のテクセルが第２の１回の読み取り動作で同時に読み出される。当該ピクセルがキャッシュ・ブロック境界を隣接するマップの位置に対応する場合、１つのマップ内で所望のテクセル・データを生成するためキャッシュから読み取られる４つのテクセルが各々異なるキャッシュ・ブロックに記憶されていることがある。このように、各ピクセルに関するキャッシュからの４つのテクセルの同時の読み取りは、キャッシュ・ディレクトリにおける６４ブロック・タグ・エントリと４回の別々の比較を行うことを必要とする場合がある。

【０１８７】

従来技術の完全連想型キャッシュは、次の２つの形態の１つで動作する。第１の形態は、１つの読取りタグを１回のサイクルであらゆるキャッシュ・タグ・エントリと比較することができるように、キャッシュ・タグ・エントリ毎に別々のハードウェア比較器を備えるものである。そのような技術は、４回の読み取りを同時に行う本発明において高価なハードウェア・コストの原因となる２５４（すなわち４×６４）個の２３ビット比較器を必要とするであろう。従来技術の完全連想型キャッシュによって使われる第２の技術は、１つのキャッシュ・タグ比較器を使用し、各キャッシュ・エントリは、読取りタグと順次比較される。そのような技術は、１回の読取り動作の間に読み取られる４つのテクセルの各々がキャッシュにあるか否かを判断するためキャッシュ・ディレクトリに対する２５６回の読取りサイクルが潜在的に必要とされるため、本発明のシステム帯域幅にマイナスの影響を与える。

【０１８８】

これらの問題を克服するため、本発明のキャッシュ・システムは、ミニディレクトリ（図２２）および主ディレクトリ（図２３）を含む。ミニディレクトリは、完全連想型ディレクトリであって、対応するブロック・インデックスと共に、５つの最も最近読まれたキャッシュ・ブロック・タグを含む。図２１に示されるように、ミニディレクトリ５００は、ミニディレクトリから出力５０１－５０５上にそれぞれ出力される５つのエントリを含む。それらエントリの各々は、４グループのタグ比較器５０７－５１０に接続される。タグ比較器５０７－５１０の各グループは、５つの２３ビット比較器（図示されてない）を含み、双線形または３線形補間が実行される時１回の読取り動作で実行される４つのキャッシュ読取りタグの１つに対応する。このように、完全連想型ミニディレクトリの特性が、同時に読み取られるタグの数にミニディレクトリのエントリの数を乗じた数に等しい２０個の２３ビット比較器を用いて実現される。

【０１８９】

同時に読み込まれるある１つのピクセルに関する４つのキャッシュ読取りタグが、該ピクセルが対応するマップ位置に最も近い４つのテクセルを含むキャッシュ・ブロックを識別する。それらの４つのキャッシュ読取りタグは、左上（ＵＬ、Upper Left）タグ、右上（ＵＲ、Upper Right）タグ、左下（ＬＬ、Lower Left）タグおよび右下（ＬＲ、Lower Right）タグと呼ばれる。左上、右上、左下および右下のテクセルのためのキャッシュ読取りタグは、それぞれ左上、右上、左下および右下のタグ比較器５０７－５１０グループに接続される。タグ比較器５０７－５１０の各グループは、その対応するキャッシュ読取りタ

グをミニディレクトリに記憶されている５つのブロック・タグと比較して、タグがミニディレクトリ・エントリの中の１つと一致することを示すヒット出力を生成し、同時に対応するテクセル・データ・ブロックが記憶されているキャッシュの位置を標示するブロック・インデックスを出力する。

【０１９０】

上述から認められるように、４つのキャッシュ読み取りタグ（ＵＬ、ＵＲ、ＬＬ、ＬＲ）の各々がミニディレクトリにあれば、対応する４つのテクセル・データ・ブロックが記憶されているキャッシュの位置を標示するブロック・インデックスを決定するため必要とされるディレクトリ・アクセスはただ１回でよい。１つまたは複数の読取りタグがミニディレクトリにない場合にだけ主キャッシュ・ディレクトリに対するアクセスが行われる。ミニディレクトリ５００は、ミニディレクトリにおいてキャッシュ読み取りタグ・ミスが発生するたびごとに更新されるので、ミニディレクトリ５００は常にテクスチャ・データの最も最近アクセスされた５つのブロックのブロック・タグを保持する。

【０１９１】

４つのキャッシュ読み取りタグの１つまたは複数がミニディレクトリでヒットしない場合、主キャッシュ・ディレクトリ５２０（図２２）に対するアクセスが行われる。上述のように、主ディレクトリは、各々が１つのブロック・タグを持つ６４のエントリを含む。主ディレクトリは、６４個の２３ビット比較器５２２を備えるため、１つのキャッシュ読み取りタグは１回のサイクルで主ディレクトリのすべてと比較することができる。比較器５２２は、キャッシュ読取りタグが主ディレクトリのエントリの１つと一致したことを標示する信号を生成し、読み取りタグと一致した比較器の位置を使用して、テクセル・データの対応するブロックがキャッシュに存在する位置を標示するブロック・インデックスを生成する。読取りタグが主キャッシュ・ディレクトリのどのエントリとも一致しない場合、キャッシュ・ミスが生成され、その結果、ホスト・コンピュータの割り込みが発生し、テクスチャ・データの要求されたブロックが上述の方法でダウンロードされる。

【０１９２】

上述のように、４つのキャッシュ読取りタグ（ＵＬ、ＵＲ、ＬＬ、ＬＲ）の１つまたは複数がミニディレクトリをヒットしない場合のみ主キャッシュ・ディレクトリ５２０がアクセスされる。キャッシュ読取りタグの複数がミニディレクトリと一致しない場合、各キャッシュ読取りタグ毎に別々のサイクルで主ディレクトリをアクセスしなければならないとすれば、処理能力低下の負荷を減らすことが望ましい。そのような成果を達成するため、本発明の１つの実施形態において、図２４に示されるように、６つの比較器５２６－５３０が追加される。これら６つの比較器は、同時にアクセスされる４つのキャッシュ読取りタグの各々を一致するものがあるか調べるため他のものと比較する。これらの比較器は、ＵＬタグをＵＲタグと比較する比較器５２６、ＵＬタグをＬＬタグと比較する比較器５２７、ＵＬタグをＬＲタグと比較する比較器５２８、ＵＲタグをＬＬタグと比較する比較器５２９、ＵＲタグをＬＲタグと比較する比較器５３０、およびＬＬタグをＬＲタグと比較する比較器５３１を含む。

【０１９３】

比較器５２６－５３２によって実行される比較は、処理性能の低下が起きないように他の比較と並列的に実行される。例えば、キャッシュ読み取りタグがミニディレクトリと比較されるサイクルの間に、あるいは、ミニディレクトリにおけるミスを起こした最初のキャッシュ読取りタグが主ディレクトリと比較される時のサイクルの間に、上記の比較を実行することができる。少くとも２つのキャッシュ読取りタグが主ディレクトリにおいてヒットせず、かつそれらが等しいものであると判断されれば、比較器５２６－５３２の出力は、これら少くとも２つのキャッシュ読取りタグについては主ディレクトリを１回だけアクセスすればよいことを示す。このようにして同一のタグについて主ディレクトリをアクセスする場合複数サイクルを伴う必要がなくなるので、複数のキャッシュ読み取りタグがミニディクトリで一致ミスを起こす場合のシステム帯域幅に対する影響を最小限にとどめることができる。

【０１９４】
上述のことから認められるように、キャッシュ・ミニディレクトリを利用する本発明の実施形態は、高いシステム帯域幅を達成する一方、キャッシュ・ディレクトリを実施するハードウェアの数量を比較的少なくするという対立する目標を効率的に均衡させる。複数のキャッシュ読取りタグがミニディレクトリで一致しない場合の処理性能の低下はアプリケーションに依存する。それぞれが４つのキャッシュ読取りタグからなるユニークな２セットをミニディレクトリに対して２サイクル毎に処理することもできるが、典型的には４つのキャッシュ読取りタグの各セットにおいてただ１つまたは２つのユニークなブロック・タグが現れると考えられる。上述のように、１つのオブジェクトの複数ピクセルがレンダリングされ、そして、３線形補間法が使われる場合、隣接するピクセルが、ＭＩＰマップの同じ２つのマップに頻繁に対応するので、キャッシュに対する読取りが２つのマップを記憶するキャッシュ・ブロックの間で連続的に切り換わることを必要とする。図２２に示される実施形態において、現在処理されている読取りタグのセットに対する４つのユニークなキャッシュ・タグがミニディレクトリにたとえ存在するとしても、前の読取りタグ・セットでアクセスされた少くとも１つのタグがミニディレクトリに残っていることを保証するため、ミニディレクトリは５つのブロック・タグを記憶する。このようにして、３線形補間の間４つのユニークなキャッシュ・タグの２つのセットの間で切り替えが行われても、各セットについて少なくとも１つの読取りキャッシュ・タグが、ミニディレクトリに残るので、４つのキャッシュ・タグを主ディレクトリと順次比較する必要がない。

【０１９５】
３線形補間を使用するテクセルのレンダリングの間、キャッシュへの連続的な読取りは、１つのマップにおける第１のセットの４つのテクセルと別のマップにおける第２のセットの４つのテクセルを読み取る。プリミティブがレンダリングされる際、２つのマップの各々の範囲内の隣接するテクセルが１つおきのサイクル毎にそれぞれアクセスされ、複数のテクセルが、１つのキャッシュ・ブロック内に一般的に位置づけられる。従って、複数のユニークなタグが４つのキャッシュ読取りタグの各セットに出現するとすれば、各ピクセルのキャッシュ読取りタグがミニディレクトリ５００においてヒットする状態を保ちながら多数のピクセルをレンダリングすることができる。もしも４つのセットの各々においてただ１つのキャッシュ読み取りタグがミニディレクトリと一致しない場合は、次の４つの読み取りタグセットがミニディレクトリと比較されている間にそのタグを主ディレクトリと比較することができるので、処理性能の低下は派生しない。

【０１９６】
主ディレクトリおよび相対的に小規模のミニディレクトリを含む本発明のキャッシュ・ディレクトリは、テクスチャ・マッピング・ハードウエアでの使用に限定されず、その他の多数のアプリケーションについて使用することができることは理解されるべきであろう。複数のキャッシュ読取りタグが同時に処理される場合、およびキャッシュ読み取りタグが連続的にアクセスされる以前に使用されたタグと相関している場合、完全連想型キャッシュを実施し、ディレクトリ・タグ比較のコストを低減する上で、本発明のミニキャッシュ・ディレクトリ方式は特に有効である。例えば、１時点でＸ個のタグを記憶するキャッシュ・メモリについて、Ｎ個のキャッシュ読み取りタグが同時にディレクトリ・ブロック・タグと比較されると仮定し、Ｍ個のタグを含むミニディレクトリを維持すれば十分である（ただしＭはＮ以上である）。Ｍ個のミニディレクトリ・タグの各々は、Ｎ個のキャッシュ読取りタグに対して、１回の読取り動作において比較される。ミニディレクトリにおいてヒットしないキャッシュ読取りタグについて、主ディレクトリが順次アクセスされる。そのような読取りタグは、主ディレクトリ・タグと１回のサイクルで比較される。主ディレクトリのＸ個のタグの各々がＮ個の読み取りタグと１回の読取り動作で比較される場合のシステムにおいて、比較器の観点からのハードウエア節約は、(Ｘ＋Ｍ＊Ｎ)／(Ｘ＊Ｎ)という比率に従う。

【０１９７】
このハードウェア節約を達成するために必要とされる処理速度の減少はアプリケーション

に依存し、連続的読み取り動作でアクセスされる一連のタグの様態に基づく。各読取りセットにおいて１つ以上のタグがミニディレクトリとの不一致を起こさなければ、ミスのあったタグと主ディレクトリとの比較は、次のセットの読み取りタグがミニディレクトリと比較されている間に並列的に行うことができるため、処理速度の低下は起きない。

【０１９８】

複数のキャッシュ読み取りタグがミニディレクトリでミスをを起こす場合の処理性能の低下を防ぐため使用される上述の比較器５２６－５３０に関しては、４つの読取りタグが同時にアクセスされるので、６つの比較器が使用される。各キャッシュ読取りタグを他のものと比較するために使われる比較器の数は、同時にアクセスされる読取りタグの数Ｎに依存し、１から(Ｎ－１)までの整数の和に等しい。

【０１９９】

図２２ないし図２４のミニディレクトリおよび主ディレクトリを含むキャッシュ・ディレクトリの１つの実施形態が図２５に示されている。図２５に示されている実施形態は例示の目的のためにすぎず、その他の形態を実施することができる点は理解されるべきであろう。

【０２００】

図２２のミニディレクトリ・エントリ５０１－５０５は、タグ・レジスタ５０１Ｔ－５０５Ｔに記憶されるタグ・コンポーネント、および、インデックス・レジスタ５０１Ｉ－５０５Ｉに記憶されるインデックス・コンポーネントに分割される。上述のように、キャッシュ・ディレクトリは、処理中のピクセルが対応するＭＩＰマップ位置に最も近い４つのテクセル（すなわちＵＬ、ＵＲ、ＬＬおよびＬＲ）に対応する一組の４つの読取りキャッシュ・タグを受け取る。４つの読取りタグの各々は、６つのタグ比較器５４１－５４６へ送られる。比較器のうちの５つ（すなわち、５４２－５４６）は、また、５つのミニディレクトリ・タグ・レジスタ５０１Ｔ－５０５Ｔの１つにそれぞれ接続している。例えば、比較器５４２は、ミニディレクトリ・エントリ１に関するタグ・レジスタ５０１Ｔに接続して、ミニディレクトリのそのエントリのタグが読取りキャッシュ・タグＵＬ、ＵＲ、ＬＬまたはＬＲのいずれかのタグと一致するか否かを標示する出力を生成する。比較器５４３－５４６は同様に動作して、読取りキャッシュ・タグＵＬ、ＵＲ、ＬＬまたはＬＲをミニディレクトリ・エントリ２－エントリ５に対するタグを記憶するタグ・レジスタ５０２Ｔ－５０５Ｔとそれぞれ比較する。４つの読取りキャッシュ・タグの新しいセットの各々が１回のサイクルでミニディレクトリと比較される。そのサイクルの終了時点で、４つのタグＵＬ、ＵＲ、ＬＬおよびＬＲは、レジスタ５５０－５５３にそれぞれ記憶される。図２５に示されるように、レジスタ５５０－５５３の各々は、また、ミニディレクトリ・タグ比較器５４２－５４６の出力を受け取る制御回路５５９に接続する。４つの読取りキャッシュ・タグの新しい１セットがミニディレクトリ・タグと比較されるサイクルの終了時点で、レジスタ５５０－５５３の各々は、その対応するタグ（すなわちＵＬ、ＵＲ、ＬＬ、ＬＲ）がミニディレクトリ・エントリの１つと一致したか否か、一致した場合はどのエントリと一致したかを標示するデータをロードされる。

【０２０１】

上述のように、ただ１つのキャッシュ読取りタグのミスがミニディレクトリで発生すれば、次のセットの４つのテクセル読取りタグがミニディレクトリと比較されている間に、そのタグは主ディレクトリと比較される。１つのキャッシュ・ミスがミニディレクトリで発生する場合、ミスを起こしたタグを含むようにミニディレクトリは更新されるので、ミニディレクトリは常に最も最近アクセスされた５つのキャッシュ・タグを反映する。次のセットの４つのテクセル読取りタグがミニディレクトリと比較されている間にミニディレクトリでミスを起こした読み取りキャッシュ・タグが主ディレクトリと比較されるサイクルの間、ミニディレクトリ・タグ・レジスタ５０１Ｔ‐５０５Ｔは、前のサイクルでミニディレクトリにおけるミスを発生したキャッシュ・タグを含むようにまだ更新されていない。従って、次のセットの読み込みキャッシュ・タグがミニディレクトリと比較される時、６番目の比較器５４１を使用して、前のサイクルでミニディレクトリとミスを起こし現在

主ディレクトリと比較されているタグと、４つの読取りタグ（ＵＬ、ＵＲ、ＬＬおよびＬＲ）を比較する。４つのキャッシュ読取りタグ（ＵＬ、ＵＲ、ＬＬおよびＬＲ）のセットにおける複数のユニークなタグがミニディレクトリのキャッシュ・ミスを発生する場合、主ディレクトリとの複数の比較が発生するため、キャッシュ・ディレクトリを通過するパイプラインが停止させられる。しかし、ただ１つのユニークなタグがミニディレクトリにおけるキャッシュ・ミスを起こすならば、キャッシュ・ディレクトリが各サイクル毎に４つのキャッシュ読取りの新しいセットを受取る後述の方法でパイプラインは続行する。

【０２０２】

上述のように、前のサイクルでミニディレクトリと比較された読取りタグがレジスタ５５０－５５３に記憶される。これらのレジスタの出力は、４対１マルチプレクサ５５５に接続する。このマルチプレクサは、主ディレクトリと比較され、ミニディレクトリが最も最近受け取った読取りキャッシュ・タグで更新されるようにするためサイクルの終了時にミニディレクトリにロードされるべき上記レジスタの１つを選択する。マルチプレクサ５５５の出力が、また、６番目の比較器５４１に接続されるので、前のサイクルでミニディレクトリにおけるキャッシュ・ミスを起こしたキャッシュ読み取りタグを新しいセットの読取りタグ（ＵＬ、ＵＲ、ＬＬおよびＬＲ）の各々と比較することができる。比較器５４２－５４６との連係で、比較器５４１は、ミニディレクトリがキャッシュ・ディレクトリによって受け取られた４つのキャッシュ読取りタグの各セットを最も最近受け取った５つの読取りタグと比較することを保証する。

【０２０３】

上述のように、マルチプレクサ５５５から出力されるキャッシュ読取りタグが、主ディレクトリと比較されるサイクルの終了時点で、ミニディレクトリ・タグ・レジスタ５０１Ｔないし５０５Ｔの１つにロードされる。このようにして、ミニディレクトリは、最も最近アクセスされたキャッシュ・タグを含むように更新される。マルチプレクサ５５５から出力される新しいキャッシュ・タグでどのエントリを更新するかの決定は、以下に記述の置き換え方式に基づいて行われる。

【０２０４】

図２４に関連して記述された６つの比較器セット５２６－５３２が、図２５においては便宜上単一の比較器ブロックとして示されている。これらの比較器の出力は、各々が制御回路５５９に送られる比較器５４１－５４６の出力と共に、いくつかの機能を実行する。ミニディレクトリに対するキャッシュ・ミスが発生すると、制御回路５５９は、ミニディレクトリの中のどのエントリが新しい読取りキャッシュ・タグと置き換えられるべきかを決定する。制御回路５５９は、ミニディレクトリに対して比較されている新しく受け取った４つの読取りキャッシュ・タグの１つ、または主ディクトリと比較された最後の読み取りキャッシュ・タグによってヒットされたいかなるエントリも置き換えることはせず、ミニディレクトリに維持される最高優先度をこれらのエントリに割り当てる。加えて、制御回路５５９は、４つの読み取りタグの先行セットによるヒットがあったミニディレクトリ・エントリに関する状態情報を記憶し、ミニディレクトリに維持される次に高い優先度をそれらのエントリに割り当てる。残りのエントリには低い優先度が割り当てられる。

【０２０５】

制御回路５５９は、少くとも１つのエントリを含む最下位優先度グループにあるエントリを置き換えのため選択する。このように、ミニディレクトリに対して比較されている新しく受け取った４つの読取りキャッシュ・タグの１つによってヒットされず、主ディレクトリに対して比較された最後の読み取りキャッシュ・タグでなく、４つの読取りタグの先行セットに含まれていなかった低位優先度グループに少くとも１つのエントリがあれば、その低位優先度グループの中のエントリの１つが置き換えのため選択される。しかし、低位優先度グループにエントリがなければ、最優先グループのエントリ（すなわち新しく受け取った４つの読取りキャッシュ・タグの１つおよび主ディクトリと比較された最後の読み取りキャッシュ・タグによってヒットされたエントリ）を除く高位優先度グループのエントリから置き換えのためのエントリが選択される。

【０２０６】
使用可能な最下位優先度ミニディレクトリ・エントリのグループが識別されると、５つのミニディレクトリ・エントリの各々が１度に１つずつ置き換えられるサイクルをとる置き換え方式に従って、そのグループ内のどのエントリを置き換えるべきか決定される。この決定を行う方法は多数ある。本発明の１つの実施形態においては、５つのミニディレクトリ・エントリに１ないし５というラベルが付けられる。置き換えられるべきエントリを最下位優先度グループから選択する場合、先ず当該グループにない最も高い番号のエントリを識別し、次に、当該グループ内で次に最も高い番号を持つエントリを置き換え用に選択する。エントリ５が最下位優先度グループにない場合、置き換え方式は１にもどって、エントリ１が次の最高の番号を持つエントリとみなされる。この置き換え方式によって、制御回路５５９は、ミニディレクトリ・エントリが１サイクルに１つずつ置き換えられるようにサイクルを進め、選択されたミニディレクトリ・タグ・レジスタ５０１Ｔ－５０５Ｔへの書き込みを制御する。

【０２０７】
制御回路５５９は、また、比較器５４１－５４６の出力をデコードして、読取りタグがミニディレクトリの１つのエントリと一致したか否か、一致したとすればどのエントリと一致したかを標示するデータを４つの読取りタグ（ＵＬ、ＵＲ、ＬＬおよびＬＲ）の各々毎に生成する。このデータは、４つの読取りタグ（ＵＬ、ＵＲ、ＬＬおよびＬＲ）の各々に対応するレジスタ５５０－５５３に記憶される。例えば、読取りタグＵＬがミニディレクトリ・エントリ３と一致すれば、制御回路５５９によってデコードされたデータは、ＵＬレジスタ５５０に記憶され、読取りタグがミニディレクトリ・エントリ３と一致したことを標示する。後述するように、このデータは、キャッシュ・ディレクトリ・パイプラインを通過し、ＵＬテクセルに関するブロック・インデックスが、ミニディレクトリ・エントリ３に関するブロック・インデックスを保持するレジスタ５０３Ｉに記憶されていることを標示する。

【０２０８】
読み取りタグセットＵＬ、ＵＲ、ＬＬおよびＬＲのただ１つのユニークなタグがミニディレクトリと一致しない場合、対応するテクスチャ・データに関するブロック・インデックスがミニディレクトリにないことを標示するデータが、その読取りタグを記憶するレジスタ５５０－５５３の各々にロードされる。次のサイクルの間、不一致のタグを記憶するレジスタ５５０－５５３の１つの出力が、主ディレクトリ５２０と比較され、読取りタグに対するブロック・インデックスが、主ディレクトリから、主ディレクトリ・ブロック・インデックスを記憶するレジスタ５６１にロードされる。また、ブロック・インデックスがミニディレクトリのエントリに対応しないことを標示するデータが、マルチプレクサ５５５の出力から、入力５６２経由で、レジスタ５６１に記憶される。

【０２０９】
上述のように、４つのテクセルが同時にアクセスされることができるようにキャッシュ・メモリは４つのインターリーブＡ－Ｄを含む。４つのテクセル読取りタグＵＬ、ＵＲ、ＬＬおよびＬＲのセットは、いかなる形態でもインターリーブＡ－Ｄに対応することができる。レジスタ５５０－５５３に記憶され、どのミニディレクトリ・エントリがテクセルＵＬ、ＵＲ、ＬＬおよびＬＲの各々に対応するかを標示するデータは、テクセルＵＬ、ＵＲ、ＬＬおよびＬＲの各々をその対応するインターリーブＡ－Ｄに関連づけるように制御される軸シフタ(barrel shifter)５６３を通過する。軸シフタの出力は、それぞれインターリーブＡ－Ｄに対応するインターリーブ・インデックス制御レジスタ５６５－５６８にロードされる。インターリーブ・インデックス制御レジスタの各々は、インターリーブに関するブロック・インデックスを記憶するミニディレクトリ・エントリを識別する。ただ１つのユニークな読み取りキャッシュ・タグだけがミニディレクトリと一致しない場合、レジスタ５５０－５５３からの出力のシフトおよびレジスタ５６５－５６８の書き込みが、主ディレクトリ５２０へのアクセスと並列して行われる。

【０２１０】

上述のように、レジスタ５６５－５６８にロードされるデータは、対応するインターリーブに関するブロック・インデックスを記憶するミニディレクトリ・エントリを識別する。このデータは、ミニディレクトリ・インデックス・レジスタ５０１Ｉ－５０５Ｉの１つおよび主ディレクトリ・ブロック・インデックス・レジスタ５６１から、各インターリーブに対する対応するブロック・インデックスを選択する複数のインターリーブ・インデックス・マルチプレクサ５７１を制御するために使用される。複数のインターリーブ・インデックス・マルチプレクサ５７１は、４つの独立した６対１マルチプレクサを表す。１つのマルチプレクサは、各インターリーブに対応して、５つのミニディレクトリ・インデック・スレジスタ５０１Ｉ－５０５Ｉおよび主ディレクトリ・ブロック・インデックス・レジスタ５６１から１つを選択する。各インターリーブ・インデックス・マルチプレクサは、同じインターリーブに対応し、どのミニディレクトリ・エントリが該インターリーブに対するブロック・インデックスを記憶しているかを識別するレジスタ５６５－５６８の１つによって制御される。インターリーブに対するブロック・インデックスがミニディレクトリ・エントリにないことをこのデータが標示する場合、対応するマルチプレクサが、ミニディレクトリ不一致に引き続いて主ディレクトリから読み取られたブロック・インデックスを記憶する主ディレクトリ・ブロック・インデックス・レジスタ５６１から提供されるインデックスを選択する。インターリーブＡ－Ｄの各々に対するブロック・インデックスは、ライン５８０－５８３上に送出され、上述の方法でキャッシュＳＤＲＡＭをアドレス指定するために使用される。

【０２１１】

上述のように、読み取りキャッシュ・タグＵＬ、ＵＲ、ＬＬおよびＬＲのセットの複数タグがミニディレクトリと一致せずしかしただ１つのユニークなキャッシュ・タグだけを含む場合、その読み取りタグに関するブロック・インデックスを提供するため、主ディレクトリ５２０は１回だけアクセスされる。このプロセスも、４つの読取りタグのどの２つが一致するかを識別する比較器５２６－５３２の出力を使用する制御回路５５９によって制御される。４つの読取りタグのセットの２つ以上が、同じキャッシュ・タグを含むミニディレクトリと一致しない場合、対応するレジスタ５５０－５５３の各々は、ブロック・インデックスがミニディレクトリ・エントリに含まれないことを標示するように制御回路５５９によってセットされる。このようにして、読み取りタグに対応するデータが、インターリーブ・インデックス・レジスタ５６５－５６８を通過する時、各々がその対応するインターリーブ・インデックス・マルチプレクサ５７１に送られるように主ディレクトリ・ブロック・インデックス制御レジスタ５６１を選択する。

【０２１２】

制御回路５５９は、また、読取りタグ・レジスタ５５０－５５３のどれが主ディレクトリに対して比較されるべきかを制御するディレクトリ制御レジスタ５７３をセットする。レジスタ５７３は、主ディレクトリに対して１時点に比較されるべきレジスタ５５０－５５３の１つを選択するようにマルチプレクサ５５５を制御する。読取りタグＵＬ、ＵＲ、ＬＬおよびＬＲの１つがミニディレクトリと一致せずしかし共通タグを共有しない場合、制御レジスタ５７３は、レジスタの中の１つだけが主ディレクトリに対して比較されるべきであることを標示するようにセットされる。このようにして、４つの読取りキャッシュ・タグのセットがミニディレクトリとの不一致を起こすただ１つのユニークなタグだけを含む場合、主ディレクトリは一度だけアクセスされる。

【０２１３】

４つの読取りキャッシュ・タグＵＬ、ＵＲ、ＬＬおよびＬＲのセットがミニディレクトリとの不一致を起こす２つ以上のユニークなタグを含む場合、キャッシュ・ディレクトリ・パイプラインを通過する上記の流れが変更され、キャッシュ・ディレクトリは、ビジー(busy)となって読取りタグの新しいセットを次のサイクルで受け取らない。ディレクトリがビジー(busy)であることを標示することによって、ミニディレクトリと一致しなかった読取りタグを含むレジスタ５５０－５５３の各々が主ディレクトリに対して比較されることができ、新しい読取りタグで上書きされない。更に、ディレクトリ・パイプラインを通過

する流れが変更されるため、ミニディレクトリと一致しなかった読取りタグの各々について主ディレクトリがアクセスされ、それらに対応するブロック・インデックスが、主ディレクトリからレジスタ５０１Ｉ－５０５Ｉまたは５６１の１つにロードされることができる。読取りキャッシュ・タグＵＬ、ＵＲ、ＬＬおよびＬＲのセットに対するブロック・インデックスのすべてが主ディレクトリから読み取られるか、あるいは既にミニディレクトリに存在するようになるまで、パイプラインは、レジスタ５５０－５５３のいずれのデータも軸シフタ５６３を通過させないように構成される。このように、テクセルＵＬ、ＵＲ、ＬＬおよびＬＲのセットは、グループとして対応するインターリーブに関連づけられる。

【０２１４】

読取りタグの１セット中の複数のユニークなタグがミニディレクトリと一致しない場合、不一致のタグは順次処理される。（タグのセットがミニディレクトリと比較される）第１のサイクルの間、制御回路５５９は、ミニディレクトリの中のどのエントリが第１の不一致読取りタグによって置き換えられるべきかを判断し、対応するレジスタ５５０－５５３には、そのブロック・インデックスがそのミニディレクトリ・エントリに記憶されることを標示するデータがロードされる。最初に処理された不一致タグを記憶するレジスタ５５０－５５３の出力が、第２のサイクルの間に、主ディレクトリ５２０と比較される時、主ディレクトリ・ブロック・インデックス・レジスタ５６１は、ミニディレクトリ・インデックス・レジスタ５０１Ｉ－５０５Ｉのどれが置き換えられるべきかを標示するデータを用いて更新される。第３のサイクルの間、対応するブロック・インデックスは、レジスタ５６１から、置き換えのため選択されたミニディレクトリ・エントリに対応するレジスタ５０１Ｉ－５０５Ｉにロードされる。

【０２１５】

ミニディレクトリと一致しなかった後続のユニークなタグの各々は、それが処理されるべき最後のタグとなるまで同じ方法で処理される。キャッシュ・ディレクトリを通過して処理される最後の不一致タグは、あたかも読取りタグのセット中でミニディレクトリと一致しない唯一のユニークなタグであるかのように取り扱われる。最後の不一致タグの処理を開始する時、ディレクトリは、読取りタグの新しいセットを受け取ることができるようにするため、ビジー(busy)であることを標示する信号をオフにセットする。

【０２１６】

最後に処理される不一致タグに関して、制御回路５５９は、その対応するレジスタ５５０－５５３に、該タグに対するブロック・インデックスがミニディレクトリ・エントリに記憶されていないことを標示するデータをロードする。これは、すべての読取りタグがミニディレクトリと比較される第１のサイクルの間に、またはその他の不一致タグの処理と並列して、実行される。最後の不一致タグが主ディレクトリと比較されるサイクルの間、レジスタ５５０－５５３のデータが、軸シフタ５６３を通過して、インターリーブ制御レジスタ５６５－５６８にロードされ、不一致タグに関するブロック・インデックスは、主ディレクトリから、主ディレクトリ・ブロック・インデックス・レジスタ５６１にロードされる。最後に、ディレクトリの最終パイプライン段階において、インターリーブ・インデックス制御レジスタ５６５－５６８の出力を使用して、それらの対応するインターリーブ・インデックス・マルチプレクサ５７１を制御することによって、最後に処理された不一致タグに関するインデックスが主ディレクトリ・ブロック・インデックス・レジスタ５６１から送られ、該セット中の他の読取りタグの各々に関するブロック・インデックスが、その対応するミニディレクトリ・インデックスレジスタ５０１Ｉ－５０５Ｉから渡される。最後に処理された不一致タグに関するブロック・インデックスを主ディレクトリ・ブロック・インデックス・レジスタ５６１からアクセスすることによって、サイクルは、このタグに関するブロック・インデックスがそのミニディレクトリ・インデックス・レジスタにロードされるのを待たないので、１サイクルが節約される点理解されるべきであろう。

【０２１７】

以上、本発明の少なくとも１つ実施形態を記述したが、当業者にとって種々の変更、修正

および改良を行うことは容易であろう。そのような変更、修正および改良は本発明の精神および対象範囲内にあるものと意図されている。従って、上記記述は、例示のためのものにすぎず、本発明をそれに限定するものとして意図されていない。

【０２１８】

本発明には、例として次のような実施様態が含まれる。

（１）キャッシュ・メモリと連動し、Ｘ個のキャッシュ・タグを記憶し、１回のキャッシュ・ディレクトリ・アクセスにおいてＮ個の読取りキャッシュ・タグに応答して該Ｎ個の読取りキャッシュ・タグの各々について対応するキャッシュ・ブロック・インデックスを提供するキャッシュ・ディレクトリであって、上記キャッシュ・タグの各々がキャッシュ・データの１ブロックに対応し、上記キャッシュ・メモリが上記キャッシュ・データのＸ個のブロックを記憶し、Ｎ個の読取りキャッシュ・タグの各々の１つについて提供される上記キャッシュ・ブロック・インデックスがＮ個の読取りキャッシュ・タグの１つに対応するキャッシュ・データ・ブロックが記憶されているキャッシュ・メモリ位置を識別するように構成され、上記Ｘ個のキャッシュ・タグのサブセットでありかつ少くともＮ個に等しいＭ個のキャッシュ・タグを記憶し、その記憶されているＭ個のキャッシュ・タグを上記Ｎ個の読取りキャッシュ・タグの各々と同時に比較し、上記Ｎ個の読取りキャッシュ・タグの少なくとも１つと一致する上記記憶されているＭ個のキャッシュ・タグの各々１つに関して対応するブロック・インデックスを提供する完全連想型のミニディレクトリと、上記Ｘ個のキャッシュ・タグを記憶し、Ｎ個未満であるＰ個の読取りキャッシュ・タグを上記Ｘ個の読取りキャッシュ・タグの各々と同時に比較し、上記Ｐ個の読取りキャッシュ・タグの１つと一致する上記記憶されているＸ個のキャッシュ・タグの各々１つに関して対応するブロック・インデックスを提供する完全連想型の主ディレクトリと、上記Ｎ個の読取りキャッシュ・タグをミニディレクトリと初期的に比較して、ミニディレクトリと一致したＮ個の読取りキャッシュ・タグの各々に関してブロック・インデックスを提供し、また、ミニディレクトリと一致しなかった上記Ｎ個の読取りキャッシュ・タグの各々を主ディレクトリと比較して各々に関するブロック・インデックスを提供する制御回路と、を備えるキャッシュ・ディレクトリ。

（２）１時点にＮ個の読取りキャッシュ・タグのただ１つが主ディレクトリと比較されるように上記Ｐが１に等しく、主ディレクトリが、Ｘ個の比較器、およびＸ個のエントリを含む記憶素子を含み、Ｘ個のエントリの各々が、記憶されているＸ個のキャッシュ・タグの１つを記憶し、Ｘ個の比較器の各々が、Ｘ個のエントリの対応する１つに記憶されているキャッシュ・タグを、主ディレクトリと比較されたＮ個の読取りキャッシュ・タグの上記ただ１つと比較し、Ｎ個の読取りキャッシュ・タグの上記ただ１つがＸ個のエントリの１つに記憶されているキャッシュ・タグと一致する場合、該Ｘ個のエントリの各々１つが主ディレクトリから提供される対応するブロック・インデックスを有する、上記（１）に記載のキャッシュ・ディレクトリ。

（３）１からＮ－１までの整数の和に等しい数の読取りキャッシュ・タグ比較器を更に備え、上記読取りキャッシュ・タグ比較器の各々が、Ｎ個の読取りキャッシュ・タグの任意の２つを比較して互いに一致するか否か判断し、制御回路が読取りキャッシュ・タグ比較器に応答して、Ｎ個の読取りキャッシュ・タグの少なくとも２つがミニディレクトリと不一致であるがそれらが互いに一致する場合、Ｎ個の読取りキャッシュ・タグの上記少なくとも２つのうちのただ１つを主ディレクトリと比較して、Ｎ個の読取りキャッシュ・タグの少なくとも２つの各々についてブロック・インデックスを提供する、上記（１）に記載のキャッシュ・ディレクトリ。

（４）キャッシュ・メモリおよびコンピュータ・グラフィックス・システムと連係して動作する上記（１）に記載のキャッシュ・ディレクトリであって、上記コンピュータ・グラフィックス・システムが、該システムによってレンダリングされるべきプリミティブを提供するため、該システムによってレンダリングされるべきプリミティブに対応するテクスチャ・マッピング・データを記憶する主メモリを含むホスト・コンピュータと、上記ホスト・コンピュータから提供されるプリミティブをレンダリングするため、レンダリングさ

れるべき少くとも１つのプリミティブに対応するテクスチャ・マッピング・データを記憶するキャッシュ・メモリおよびキャッシュ・ディレクトリを含むプリミティブ・レンダリング・ハードウエアと、上記ホスト・コンピュータを上記プリミティブ・レンダリング・ハードウエアに接続し、レンダリングされるべきプリミティブを上記ホスト・コンピュータから上記プリミティブ・レンダリング・ハードウエアへ転送する経路となるプリミティブ・データ経路とを備える構成をなすキャッシュ・ディレクトリ。

【０２１９】

（５）Ｎ個の読取りキャッシュ・タグをＸ個の記憶されたキャッシュ・タグと比較して、Ｎ個の読取りキャッシュ・タグの各々がＸ個の記憶されたキャッシュ・タグの１つと一致するか否かを判断する機能を持つキャッシュ・ディクトリを操作する方法であって、(a)主ディレクトリにＸ個のキャッシュ・タグを記憶するステップと、(b)Ｎに少なくとも等しくかつＸ未満であるＭ個のキャッシュ・タグを含む上記Ｘ個のキャッシュ・タグのサブセットを、ミニディレクトリに記憶するステップと、(c)Ｎ個の読取りキャッシュ・タグを、ミニディレクトリに記憶されたＭ個のキャッシュ・タグと比較するステップと、(d)記憶されたＭ個のキャッシュ・タグの１つと一致するＮ個の読取りキャッシュ・タグの各々がＸ個の記憶されたキャッシュ・タグの１つと一致することを標示するステップと、(e)Ｎ個の読取りキャッシュ・タグの少なくとも１つが記憶されたＭ個のキャッシュ・タグのいずれとも一致しない場合、Ｎ個の読取りキャッシュ・タグの少なくとも１つを、主ディレクトリに記憶されたＸ個のキャッシュ・タグと比較するステップと、を含む方法。

（６）上記ステップ(c)が、Ｎ個の読取りキャッシュ・タグの各々をミニディレクトリに記憶されたＭ個のキャッシュ・タグと同時に比較するステップを含む、上記（５）に記載の方法。

（７）上記ステップ(e)が、Ｎ個の読取りキャッシュ・タグの少なくとも１つを主ディレクトリに記憶されたＸ個のキャッシュ・タグと順次に比較するステップを含み、さらに、(f)Ｎ個の読取りキャッシュ・タグの少くとも２つからなるグループがミニディレクトリに記憶されたＭ個のキャッシュ・タグのいずれとも一致しない場合、Ｎ個の読取りキャッシュ・タグの上記グループ内の各々をそのグループ内の他のキャッシュ・タグと比較してそのグループ内の少なくとも２つの読取りキャッシュ・タグが一致するか否かを判断するステップを含み、上記ステップ(f)が一致するタグがあると判断する場合、上記ステップ(e)が上記同一グループ内で一致するＮ個の読取りキャッシュ・タグのただ１つを主ディレクトリに記憶されたＸ個のキャッシュ・タグと比較するステップを含む、上記（５）に記載の方法。

（８）Ｎ個の読取りキャッシュ・タグの複数セットが、キャッシュ・ディレクトリに対して連続的ディレクトリ・アクセスの形態で比較され、第１回のディレクトリ・アクセスの間にキャッシュ・ディレクトリに対して比較されるＮ個の読取りキャッシュ・タグがミニディレクトリに記憶されたＭ個のキャッシュ・タグのいずれとも一致しないただ１つのユニークなキャッシュ・タグを含む場合、上記ただ１つのユニークなキャッシュ・タグが、第１回のディレクトリ・アクセスの際の上記ステップ（ｅ）の間に、主ディレクトリに記憶されているＸ個のキャッシュ・タグと比較され、一方、上記Ｎ個の読取りキャッシュ・タグの第２のセットが、第２回のディレクトリ・アクセスの際の上記ステップ（ｃ）の間に、ミニディレクトリに記憶されたＭ個のキャッシュ・タグと同時に比較される、上記（５）に記載の方法。

【０２２０】

（９）キャッシュ・メモリと連動し、Ｘ個のキャッシュ・タグを記憶し、１回のキャッシュ・ディレクトリ・アクセスにおいてＮ個の読取りキャッシュ・タグに応答して該Ｎ個の読取りキャッシュ・タグの各々について対応するキャッシュ・ブロック・インデックスを提供するキャッシュ・ディレクトリであって、上記キャッシュ・タグの各々がキャッシュ・データの１ブロックに対応し、上記キャッシュ・メモリが上記キャッシュ・データのＸ個のブロックを記憶し、Ｎ個の読取りキャッシュ・タグの各々の１つについて提供される上記キャッシュ・ブロック・インデックスがＮ個の読取りキャッシュ・タグの１つに対応

するキャッシュ・データ・ブロックが記憶されているキャッシュ・メモリ位置を識別するように構成され、上記Ｘ個のキャッシュ・タグのサブセットでありかつ少くともＮ個に等しいＭ個のキャッシュ・タグを記憶し、その記憶されているＭ個のキャッシュ・タグを上記Ｎ個の読取りキャッシュ・タグの各々と同時に比較し、上記Ｎ個の読取りキャッシュ・タグの少なくとも１つと一致する上記記憶されているＭ個のキャッシュ・タグの各々１つに関して対応するブロック・インデックスを提供する完全連想型のミニディレクトリと、上記Ｘ個のキャッシュ・タグを記憶し、Ｎ個未満であるＰ個の読取りキャッシュ・タグを上記Ｘ個の読取りキャッシュ・タグの各々と同時に比較し、上記Ｐ個の読取りキャッシュ・タグの１つと一致する上記記憶されているＸ個のキャッシュ・タグの各々１つに関して対応するブロック・インデックスを提供する完全連想型の主ディレクトリと、上記Ｎ個の読取りキャッシュ・タグをミニディレクトリと初期的に比較して、ミニディレクトリと一致したＮ個の読取りキャッシュ・タグの各々に関してブロック・インデックスを提供し、また、ミニディレクトリと一致しなかった上記Ｎ個の読取りキャッシュ・タグの各々を主ディレクトリと比較して各々に関するブロック・インデックスを提供する制御手段と、を備えるキャッシュ・ディレクトリ。

（１０）Ｐが１に等しく、上記制御手段が、Ｎ個の読取りキャッシュ・タグの少なくとも２つがミニディレクトリと一致せずかつ互いに同一でない場合ディレクトリ・ビジー信号をオンにセットし、各々を個別に主ディレクトリと比較する手段を含む、上記（９）に記載のキャッシュ・ディレクトリ。

【０２２１】

【発明の効果】

本発明に従うテクスチャ・マッピング・ハードウエアへのテクスチャ・データのダウンロードが、プリミティブ伝送経路と異なる経路を使用して実施され、従って３次元プリミティブ・パイプラインのフラッシングを必要としないので、システムの帯域幅および処理能力が向上する。また、本発明の１つの実施形態において、特定のプリミティブ・レンダリング・タスクが複数のプリミティブに関して並列的に実行されるように、テクスチャ・マッピング基板およびフレーム・バッファ基板に関する諸ハードウェアが反復配置されるため、システムの帯域幅が拡大される。

【０２２２】

更にまた、本発明に従うキャッシュ・メモリ内のＳＤＲＡＭチップの各々は、同時に別々の活動ページ（すなわち、共通の行アドレスを持つメモリ位置グループ）を維持することができる２つの等しいサイズのバンクに内部的に分割されるため、従来技術のＤＲＡＭの場合に起きるような２つの異なるページ（すなわち２つの異なる行アドレス）からデータを取り出すことに付随する再ページングの負荷を伴うことなく、ＳＤＲＡＭチップの２つのバンク内の異なるページにあるデータを連続的読取りサイクルでアクセスすることができるという効果を持つ。

【０２２３】

また、本発明の１つの実施形態において、多数の近接した読み取りサイクル内で共通のテクセル・データが多数回アクセスされる場合、キャッシュは、最初の読取りについてのみアクセスされるだけで、後続の読取りの各々についてはキャッシュ読取りサイクルが節約され、これによって、必要とされるキャッシュ・アクセス数が減り、システムの帯域幅が増大する。

【０２２４】

加えて、本発明に従って、キャッシュ・ディレクトリを主ディレクトリおよびミニディレクトリに分割構成することによって、同一のタグについて主ディレクトリをアクセスする場合複数サイクルを伴う必要がなくなるので、複数のキャッシュ読み取りタグがミニディクトリで一致ミスを起こす場合のシステム帯域幅に対する影響を最小限にとどめることができ、高いシステム帯域幅を達成する一方キャッシュ・ディレクトリを実施するハードウェアの数量を比較的少なくすることができる。

【０２２５】

このように、本発明は、種々の局面において、コンピュータ・グラフィックス・システムなどのデータ処理システムの処理能力を向上させる効果を奏する。

【図面の簡単な説明】

【図１】一組のテクスチャＭＩＰマップの例を示す図である。

【図２】本発明のコンピュータ・グラフィックス・システム全体の１つの実施形態のブロック図である。

【図３】本発明のコンピュータ・グラフィックス・システム全体の別の１つの実施形態のブロック図である。

【図４】本発明のテクスチャ・マッピング・ハードウェアのブロック図である。

【図５】本発明のテクスチャ・マッピング・ハードウェアのパラメータ補間回路エレメントのブロック図である。

【図６】本発明のキャッシュ・メモリおよびテクスチャ・マッピング・ハードウェアの一部を示すブロック図である。

【図７】テクスチャ・データのブロックが、本発明のキャッシュ・メモリの４つのインターリーブ形態の利点を生かすように構成される様態を示す図である。

【図８】本発明のキャッシュ・メモリを形成するメモリ・チップの構成を示すブロック図である。

【図９】本発明のテクスチャ・マッピング・ハードウェアの一部を示すブロック図である。

【図１０】本発明のテクスチャ・マッピング方式に従って、ピクセルの各々について隣接するＭＩＰマップからアクセスされるテクセルの例を示す図である。

【図１１】テクスチャ・マッピング・ハードウェア・バッファおよび図１０の例に従う関連データのエントリを示す図である。

【図１２】本発明のテクスチャ・マッピング・ハードウェアによって使われる回路のブロック図である。

【図１３】一組のテクスチャＭＩＰマップの例を示す図である。

【図１４】本発明のメモリ記憶方式に従って図１３の例のＭＩＰマップがメモリに記憶される形態を示す図である。

【図１５】本発明のメモリ記憶方式に従ってＭＩＰマップが細分化される形態を示すＭＩＰマップのブロック図である。

【図１６】本発明のメモリ記憶方式に従ってＭＩＰマップが更に細分化される形態を示す図１５のＭＩＰマップ部分のブロック図である。

【図１７】キャッシュ・ブロック・タグが生成される形態を示す図である。

【図１８】補間されたテクセルを基に対応するテクスチャ・データ・ブロックを持つテクセル・アドレスを決定するプロセスを示す流れ図である。

【図１９】キャッシュ・ミスが発生する時置き換えられるべきキャッシュ・ブロックを決定するプロセスを表す流れ図である。

【図２０】テクスチャ・マッピング・チップにおいて提供されるテクセル・ポート・レジスタを示す図である。

【図２１】ホスト・コンピュータにおいてキャッシュ・ミス割り込みを処理するプロセスを示す流れ図である。

【図２２】キャッシュのミニディレクトリのブロック図である。

【図２３】キャッシュの主ディレクトリのブロック図である。

【図２４】キャッシュ読取りタグがミニディレクトリと一致しない場合の処理能力低下を防ぐために用意される一連の比較器のブロック図である。

【図２５】本発明のキャッシュ・ディレクトリの１つの実施形態を表すブロック図である。

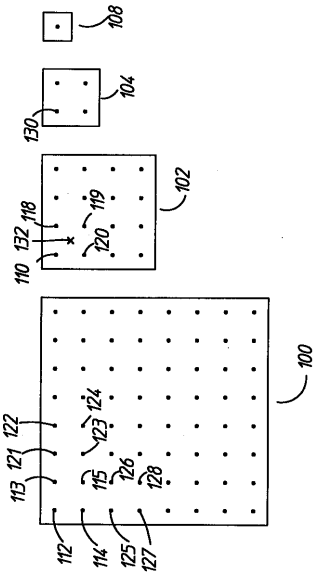
【符号の説明】

１０　フロントエンド基板

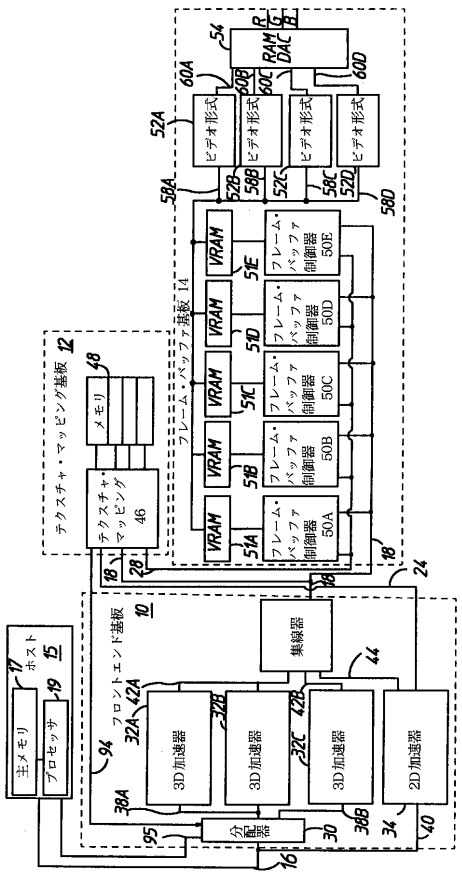
１２　テクスチャ・マッピング基板

１４　フレーム・バッファ基板
１５　ホスト・コンピュータ
４６　テクスチャ・マッピング・チップ
４８　ローカル・メモリまたはキャッシュ・メモリ
１００　基本マップ
１０２、１０４、１０８　ＭＩＰマップ
１１０、１１２、１３０　テクセル
１３２　加重平均テクセル
２０４Ａ、２０４Ｂ、２０４Ｃ、２０４Ｄ　インターリーブ
２０６Ａ、２０６Ｂ、２０６Ｃ、２０６Ｄ　キャッシュ・アクセス・コマンドＦＩＦＯ
２１４Ａ０、２１４Ａ１、２１４Ｂ０、２１４Ｂ１、２１４Ｃ０、２１４Ｃ１、２１４Ｄ０、２１４Ｄ１　テクセル・データＦＩＦＯ
２１６　テクセル補間回路コマンドＦＩＦＯ
５５５　４対１マルチプレクサ

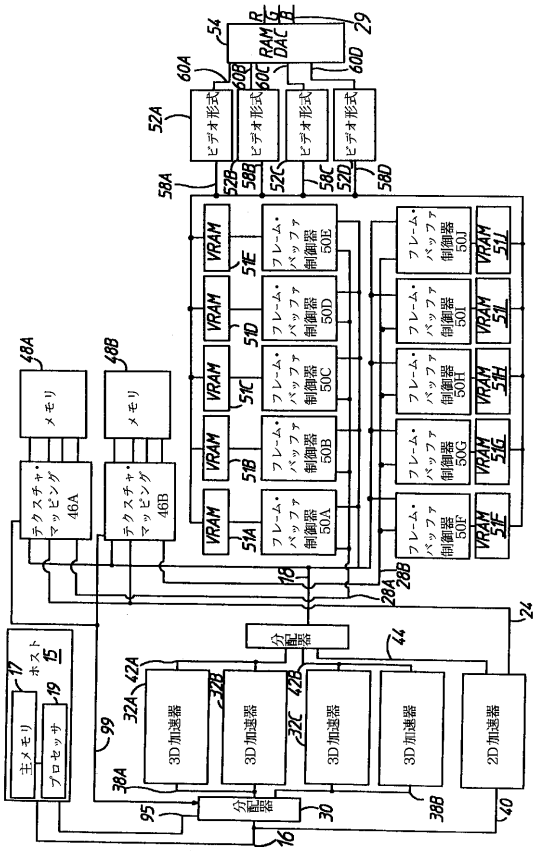
【図１】



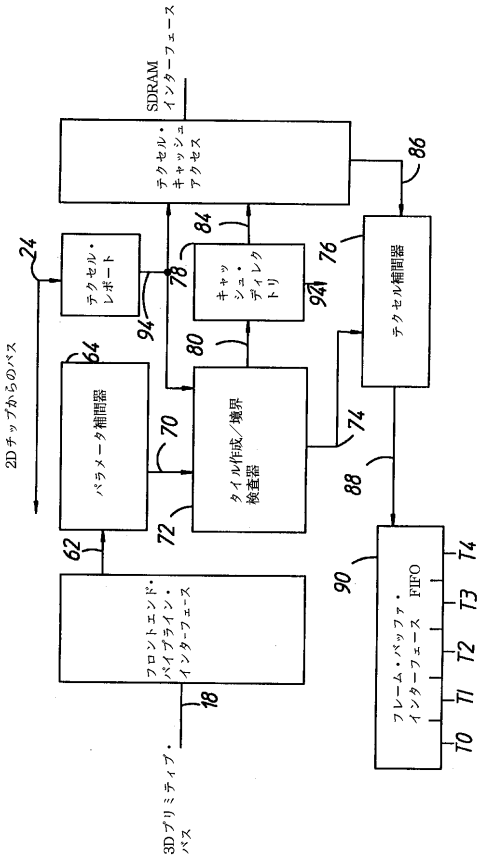
【図２】



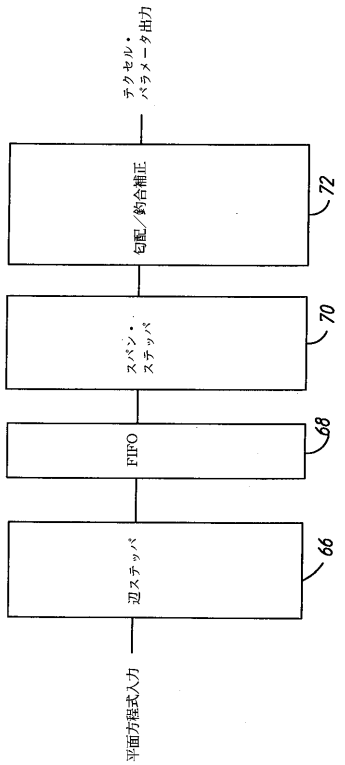
【図３】


メモリ
テクスチャ・マッピング
主メモリ
プロセッサ
ホスト
3D加速器
2D加速器
分配器
VRAM
フレーム・バッファ制御器
ビデオ形式
RAM DAC

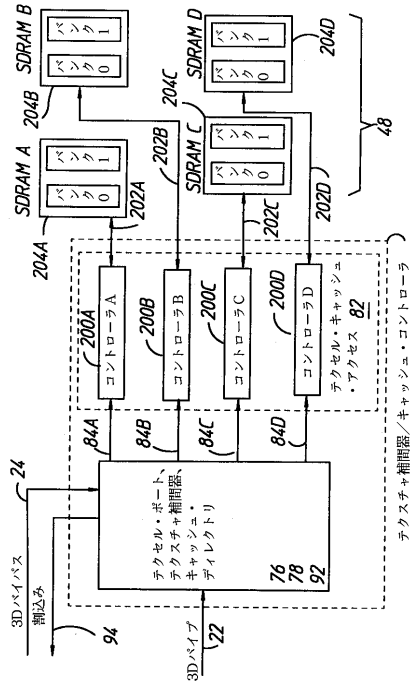
【図４】


SDRAMインターフェース
テクセル・キャッシュアクセス
テクセル・レポート
キャッシュ・ディレクトリ
テクセル補間器
パラメータ補間器
タイル作成／境界検査器
フロントエンド・パイプライン・インターフェース
フレーム・バッファ・インターフェース FIFO
2Dチップからのバス
3Dプリミティブ・バス

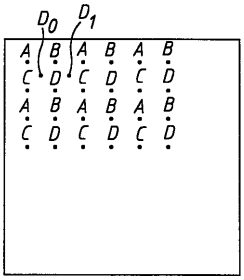
【図５】


テクセル・パラメータ出力
切配／約合補正
スパン・ステッパ
FIFO
辺ステッパ
平面方程式入力

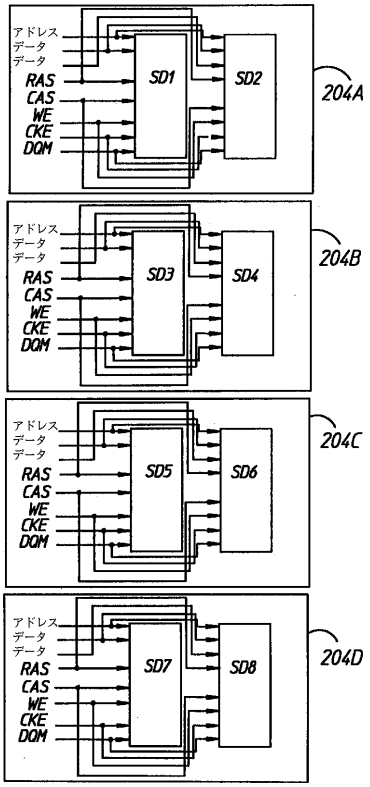
【図６】


SDRAM A
SDRAM B
SDRAM C
SDRAM D
バンク0
バンク1
コントローラA
コントローラB
コントローラC
コントローラD
テクセル・キャッシュ・アクセス
テクセル・ポート、テクスチャ補間器、キャッシュ・ディレクトリ
テクスチャ補間器／キャッシュ・コントローラ
3Dバイパス
割込み
3Dバイプ

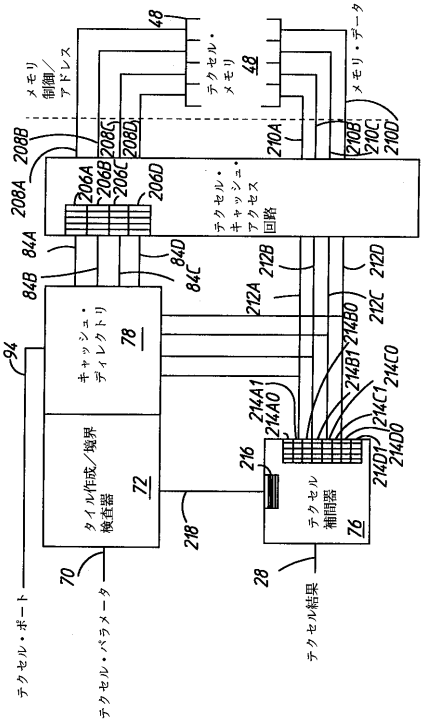
【図７】


D0
D1
A
B
C
D

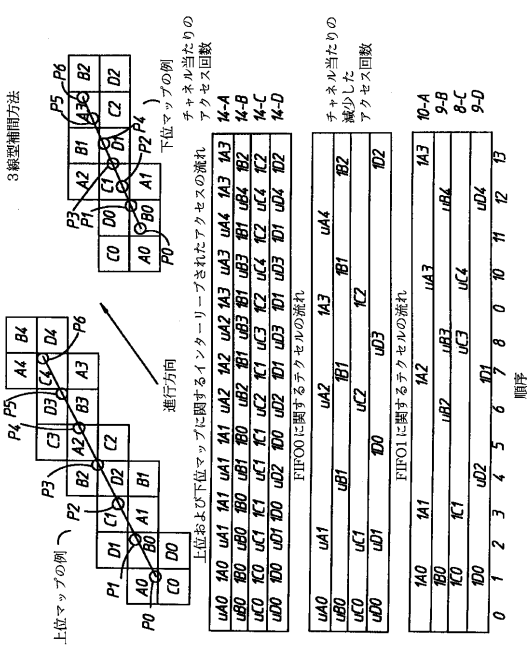
【図８】


アドレス
データ
データ
RAS
CAS
WE
CKE
DQM
SD1
SD2
204A
SD3
SD4
204B
SD5
SD6
204C
SD7
SD8
204D

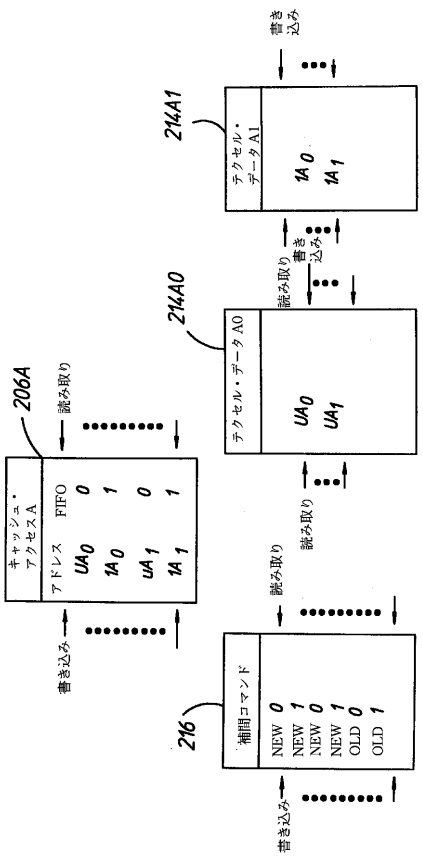
【図９】


メモリ制御／アドレス
テクセル・メモリ
メモリ・データ
テクセル・キャッシュ・アクセス回路
キャッシュ・ディレクトリ
タイル作成／境界検査器
テクセル補間器
テクセル・ポート
テクセル・パラメータ
テクセル結果

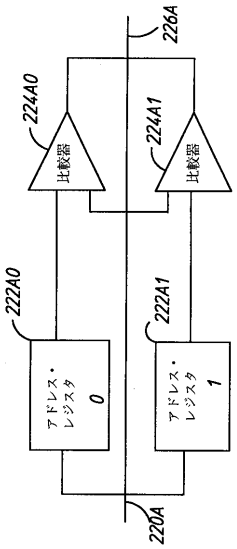
【図１０】


3線型補間方法
上位マップの例
下位マップの例
進行方向
上位および下位マップに関するインターリーブされたアクセスの流れ
チャネル当たりのアクセス回数
FIFO0に関するテクセルの流れ
FIFO1に関するテクセルの流れ
チャネル当たりの減少したアクセス回数
順序

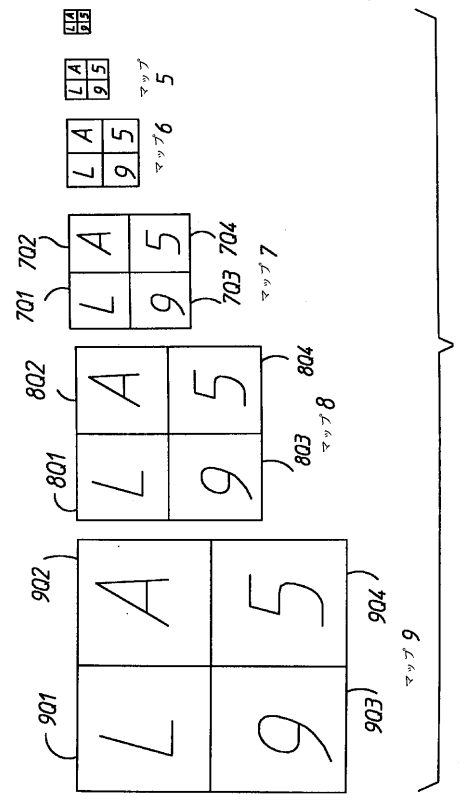
【図１１】



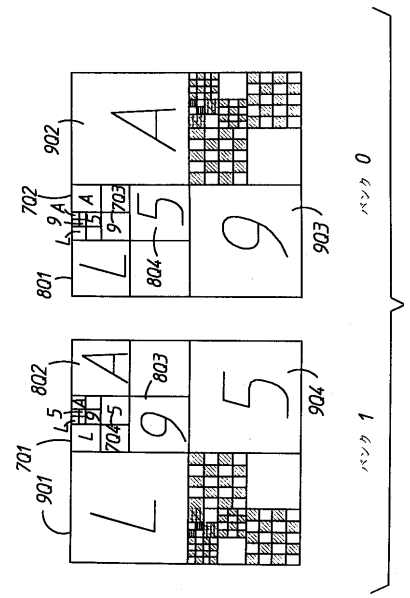
【図１２】



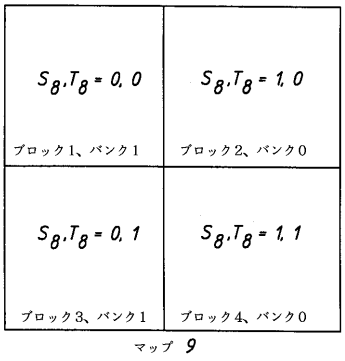
【図１３】



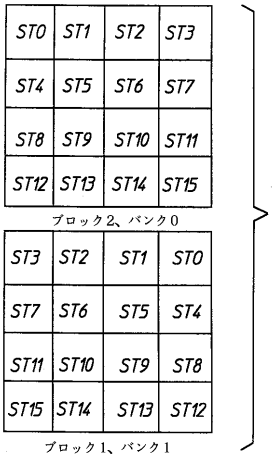
【図１４】



【図１５】



【図１６】

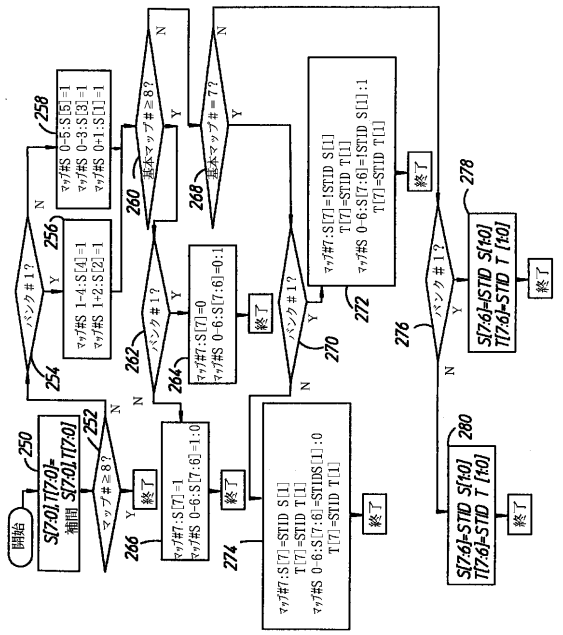


【図１７】

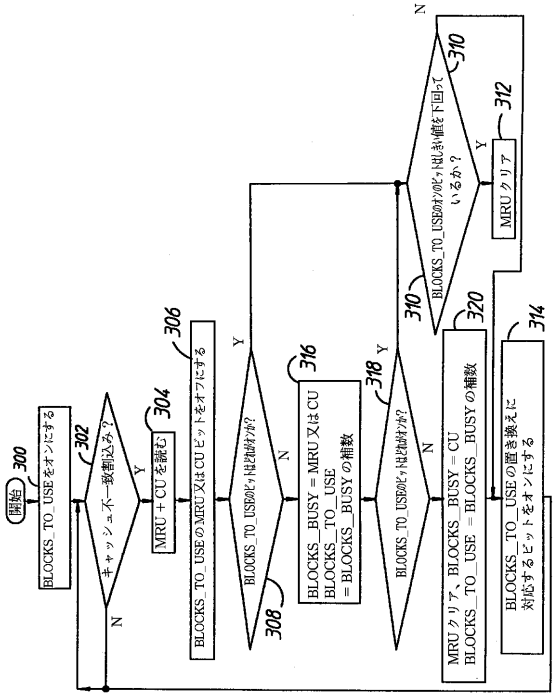
ブロック・タグ [22:0] = {テクスチャID [1:0],マップ・ビット,T[14:8],S[14:8]}

マップ番号	マップビット	Tパラメータ軸上位ビット							Sパラメータ軸上位ビット							バンク
15	0	t14	t13	t12	t11	t10	t09	t08	s14	s13	s12	s11	s10	s09	s08	t14^s14
14	1	0	t13	t12	t11	t10	t09	t08	0	s13	s12	s11	s10	s09	s08	t13^s13
13	1	1	0	t12	t11	t10	t09	t08	0	0	s12	s11	s10	s09	s08	t12^s12
12	1	1	1	0	t11	t10	t09	t08	0	0	0	s11	s10	s09	s08	t11^s11
11	1	1	1	1	0	t10	t09	t08	0	0	0	0	s10	s09	s08	t10^s10
10	1	1	1	1	1	0	t09	t08	0	0	0	0	0	s09	s08	t9^s9
9	1	1	1	1	1	1	0	t08	0	0	0	0	0	0	s08	t8^s8
≦8	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	タグ [7]
≦8	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	タグ [7]

【図１８】



【図１９】



【図２０】

テクセル・コマンド・レジスタ

31:9	8	7	6:4	3	2	1	0
未使用	停止	割込可能	未使用	書き込みLoki1	書込みLoki0	未使用	Loki読取り
	350	352		356	359		358

テクセル状態レジスタ

31:7	6	5	4	3	2	1	0
未使用	複数Loki	未使用	割込有効	割込可能	停止	停止可能	未使用
	360		364	362	368	366	

パイプ・タグ・レジスタ

31:23	22:0
未使用	ブロック・タグ
	370

テクセル・データ・レジスタ

31:24	23:16	15:8	7:0
アルファ	赤	緑	青
372	374	376	378

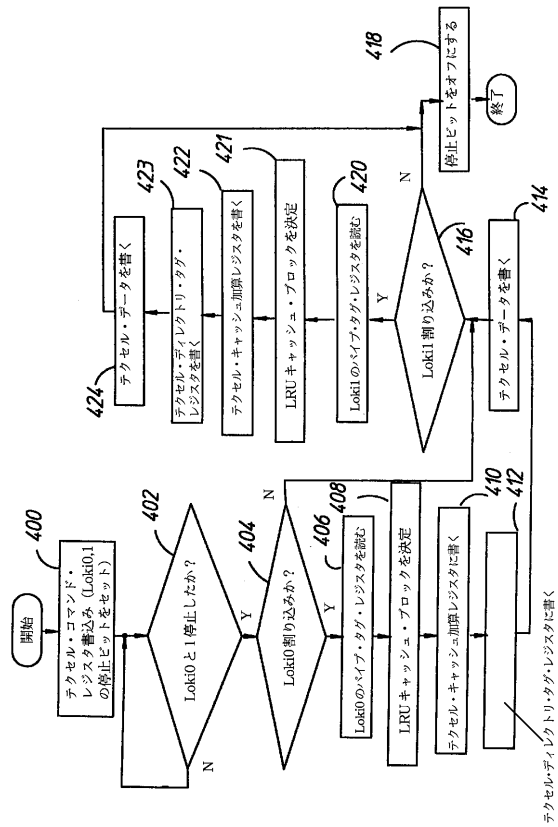
テクセル・キャッシュ・アドレス・レジスタ

32:22	21:16	15:0
未使用	ブロック・インデックス	ブロック・アドレス
	380	382

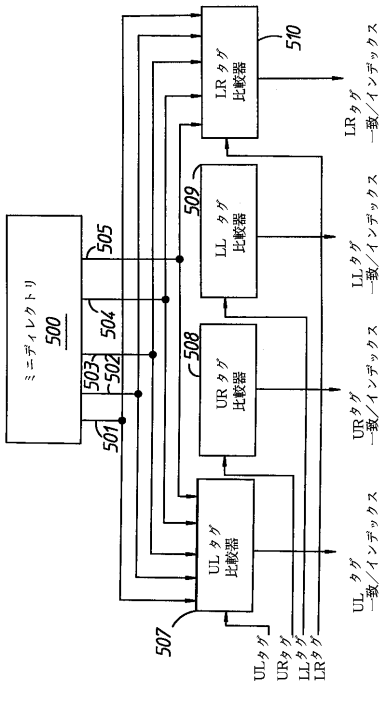
テクセル・ディレクトリ・タグ・レジスタ

31:23	22:0
未使用	ブロック・タグ
	384

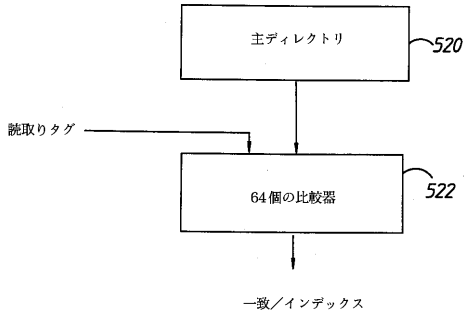
【図２１】



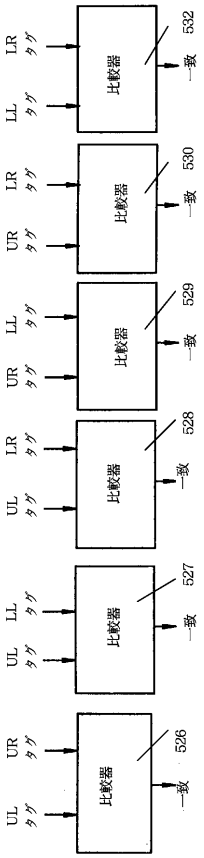
【図２２】



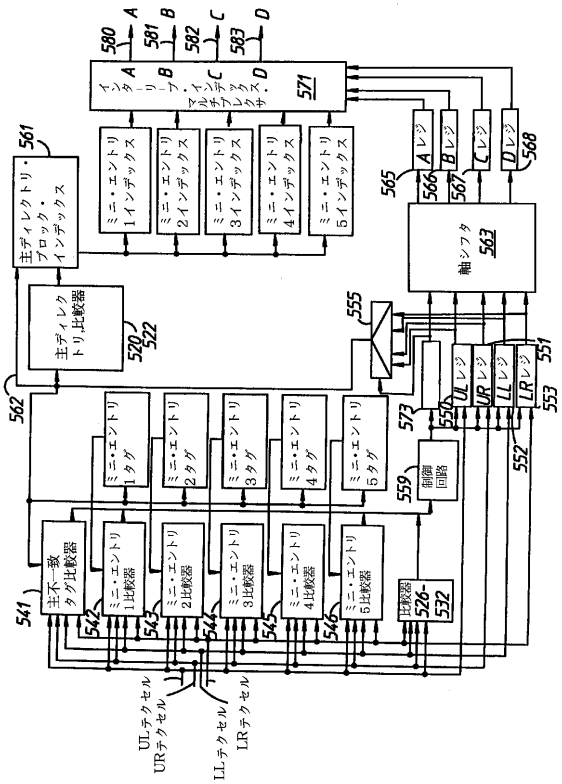
【図２３】


主ディレクトリ
520
読取りタグ
64個の比較器
522
一致／インデックス

【図２４】


比較器
LR タグ
LL タグ
UR タグ
UL タグ
一致
532
530
529
528
527
526

【図２５】


主ディレクトリ・ブロック・インデックス
561
主ディレクトリ比較器
520
522
562
ミニ・エントリ 1インデックス
ミニ・エントリ 2インデックス
ミニ・エントリ 3インデックス
ミニ・エントリ 4インデックス
ミニ・エントリ 5インデックス
インターリーブ・マルチブロック・インデックス
571
580
581
582
583
A
B
C
D
Aレジ
Bレジ
Cレジ
Dレジ
565
566
567
568
軸シフタ
563
555
ULレジ
URレジ
LLレジ
LRレジ
551
553
550
573
552
559
制御回路
ミニ・エントリ 1タグ
ミニ・エントリ 2タグ
ミニ・エントリ 3タグ
ミニ・エントリ 4タグ
ミニ・エントリ 5タグ
主不一致タグ比較器
541
ミニ・エントリ 1比較器
ミニ・エントリ 2比較器
ミニ・エントリ 3比較器
ミニ・エントリ 4比較器
ミニ・エントリ 5比較器
542
543
544
545
546
比較器
526
532
ULテクセル
URテクセル
LLテクセル
LRテクセル

フロントページの続き

(56)参考文献 特開平０６－３１８１７８（ＪＰ，Ａ）
特開平０５－３０３８９１（ＪＰ，Ａ）
特開平０２－２２４０４２（ＪＰ，Ａ）
特表平１１－５０１７４４（ＪＰ，Ａ）
特表平１０－５０４９２２（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F12/08-12/12
G06T 1/00-17/50