(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-5569
(P2004-5569A)
(43)公開日　平成16年1月8日(2004.1.8)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 15/80	G06F 15/80		5B080
G06T 15/00	G06T 15/00	100A	

審査請求 未請求　請求項の数 10　ＯＬ　（全 31 頁）

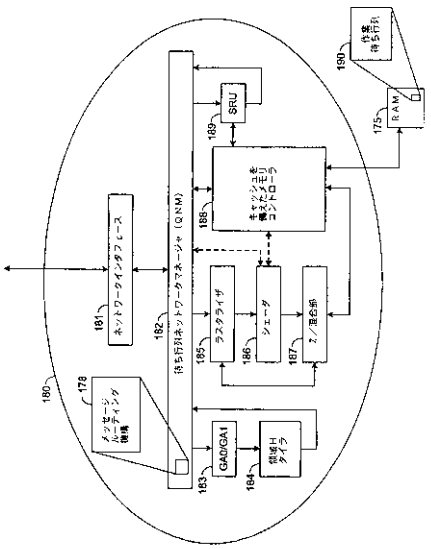
(21)出願番号　特願2003-104747(P2003-104747)
(22)出願日　平成15年4月9日(2003.4.9)
(31)優先権主張番号　10/147763
(32)優先日　平成14年5月16日(2002.5.16)
(33)優先権主張国　米国(US)

(71)出願人　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　110000039
特許業務法人アイ・ピー・エス
(72)発明者　ダニエル・エヌ・エモット
アメリカ合衆国コロラド州フォート・コリンズ・エヌ・カントリー・ロード１１・５５２４

Ｆターム(参考)　5B080 AA14 AA15 BA03 CA01 CA03 CA04 FA02 FA03 FA17 GA02 GA11 GA22

(54)【発明の名称】データおよび命令の流れを少なくとも１つの機能ユニットに配向（ｄｉｒｅｃｔ）するシステムおよび方法

(57)【要約】
【課題】データおよび命令の流れを少なくとも１つの機能ユニットに配向する。
【解決手段】複数のノードを定義するシステムにおいて、各ノードの一部を形成するキューネットワークマネージャ（ＱＮＭ）を提供する。ＱＮＭは、複数のノードの間での相互通信をサポートする、ネットワークへのインタフェースと、ノード内の機能ユニットにメッセージを渡すように構成されるインタフェースと、メッセージおよびプログラム可能命令のうちの少なくとも一方を格納するように構成されるランダムアクセスメモリ（ＲＡＭ）と、プログラム可能命令の内容に基づいて、機能ユニットの動作態様を制御するように構成されるロジックとを備える。
【選択図】　図５



【特許請求の範囲】
【請求項１】
複数のノード（４０２、４０４、４０６、４０８）を規定する構成要素のシステムにおいて、各ノード（４０２、４０４、４０６、４０８）の一部を形成する待ち行列ネットワークマネージャ（ＱＮＭ）（５８２）であって、
前記複数のノード（４０２、４０４、４０６、４０８）間の相互通信をサポートする、ネットワークへのインタフェース（５０４）と、
前記ノード内の機能ユニット（５２２、５２４）にメッセージを渡すように構成されるインタフェース（５０８）と、
メッセージ（５１２）およびプログラム可能命令（５１４）のうちの少なくとも一方を格納するように構成されるランダムアクセスメモリ（ＲＡＭ）（５１０）と、
前記プログラム可能命令（５１４）の内容に基づいて、機能ユニットの動作態様を制御するように構成されるロジック（５４０）と
を備えるＱＮＭ。
【請求項２】
前記機能ユニット（５２２、５２４）の動作態様を制御するように構成されるロジックは、前記ＲＡＭ（５１０）に格納された前記プログラム可能命令（５１４）を評価するように構成される実行ユニット（５４０）を該ＱＮＭ（５８２）内に備える
請求項１記載のＱＮＭ。
【請求項３】
少なくとも１つの実行ユニット（５４０、５４２）をさらに備え、該実行ユニット（５４０、５４２）はそれぞれ、機能動作を行うように構成される
請求項１記載のＱＮＭ。
【請求項４】
前記プログラム可能命令（５１４）は、少なくとも１つの実行ユニット（５４０、５４２）の動作を制御するように動作可能である
請求項３記載のＱＮＭ。
【請求項５】
リモートノード（４０２、４０４、４０６）からプログラム可能命令（５１４）を受け取り、該プログラム可能命令（５１４）を前記ＲＡＭに格納するロジック（５３０）をさらに備える
請求項１記載のＱＮＭ。
【請求項６】
前記機能ユニットは、メッセージ生成機能ユニット（５２４）である
請求項１記載のＱＮＭ。
【請求項７】
前記機能ユニットは、メッセージ消費側機能ユニット（５２２）である
請求項１記載のＱＮＭ。
【請求項８】
前記システムは、コンピュータグラフィックスシステムである
請求項１記載のＱＮＭ。
【請求項９】
前記少なくとも１つのメッセージ（５１２）は、前記少なくとも１つの機能ユニット（５２２、５２４）に対する命令を含む
請求項１記載のシステム。
【請求項１０】
複数のノード（４０２、４０４、４０６、４０８）を規定する構成要素のシステムにおいて、データおよび命令の流れを少なくとも１つの機能ユニット（５２２、５２４）に配向する方法であって、
第１のインタフェースを通してリモートノード（４０２、４０４、４０６）からプログラ

ム可能命令を受け取ること（６１２）と、
前記プログラム可能命令（５１４）をランダムアクセスメモリ（ＲＡＭ）（５１０）に格納すること（６１４）と、
前記プログラム可能命令（５１４）の内容を用いることであって、実行ユニット（５４０、５４２）の動作を構成すること（６１６）と
を含む方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、コンピュータグラフィックスシステムに関し、特に、データおよび命令の流れを少なくとも１つの機能ユニットに配向する新規のシステムおよび方法に関する。
【０００２】
【従来の技術】
コンピュータグラフィックスシステムは一般に、二次元映像表示画面にオブジェクトの図形表示を表示するために使用される。
現在のコンピュータグラフィックス表示システムは、高精細表現を提供し、様々な用途に使用されている。
コンピュータグラフィックス表示システムは一般に、中央演算処理装置（ＣＰＵ）、システムメモリ、グラフィックスマシン、および映像表示画面を備える。
【０００３】
通常のコンピュータグラフィックス表示システムでは、表示画面に示されるオブジェクトが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、グラフィックス表示の基本成分であって、点、線、ベクトル、およびポリゴン（たとえば、三角形および四角形）を含むことができる。
通常は、ハードウェア／ソフトウェア手法が実行されて、表示画面に表示される１つまたは複数のオブジェクトのビューを表すグラフィックスプリミティブをレンダリング、すなわち描画する。
【０００４】
通常、レンダリングされる三次元オブジェクトのプリミティブは、プリミティブデータに関してホストＣＰＵによって定義される。
たとえば、プリミティブが三角形の場合、ホストコンピュータはその頂点のＸ、Ｙ、およびＺ座標でプリミティブを定義できる。
特定の用途では、付加的なプリミティブデータを使用することができる。
レンダリングハードウェアは、各々のプリミティブを表示する表示画面ピクセルを計算するためにプリミティブデータを補間する。
【０００５】
グラフィックスマシンは、ジオメトリアクセラレータ、ラスタライザ、フレームバッファコントローラ、およびフレームバッファを一般に備える。
グラフィックスマシンは、テキスチャマッピングハードウェアを備える場合もある。
ジオメトリアクセラレータは、表示されるビューを構成するプリミティブを定義する頂点データをホストＣＰＵから受け取る。
知られているように、ジオメトリアクセラレータの動作は、非常に計算集約的である。
三次元（３Ｄ）グラフィックスディスプレイの１フレームは、数十万程度のプリミティブを含みうる。
最新技術のパフォーマンスを実現するために、ジオメトリアクセラレータは、１秒当たり数億もの浮動小数点計算を行うように求められることがある。
さらに、ホストコンピュータとグラフィックスハードウェアとの間で転送されるデータの量は、非常に多い。
ホストコンピュータからジオメトリアクセラレータに転送されるさらなるデータとしては、照度パラメータ、クリッピングパラメータ、およびグラフィックス表示の生成に必要な

任意の他のパラメータが含まれる。
【０００６】
知られているように、ラスタライザは、図またはオブジェクトを表すデータを受け取り、次いで図のピクセル様表現を提供する。
同じく知られているように、テキスチャマッピングには、テキスチャの１つまたは複数の点要素（テクセル）を、テキスチャがマッピングされているオブジェクトの表示部分の各点要素（ピクセル）に適用することが含まれる。
テキスチャマッピングハードウェアには、従来、テキスチャマップのテクセルが、オブジェクトを表す表示画面上のピクセルに対応する態様を示す情報が提供される。
テキスチャマップの各テクセルは、二次元テキスチャマップ上の位置を特定するＳおよびＴ座標によって定義することができる。
各ピクセルについて、それをマッピングする、対応するテクセル（１つまたは複数）にテキスチャマップからアクセスし、これを、表示画面上のテキスチャオブジェクトを表すためにピクセルごとに生成される最終的な赤、緑、青（ＲＧＢ）値に組み込む。
知られているように、二次元テキスチャマップにさらに、一次元、三次元、さらには他の次元のテキスチャマップもまた知られている。
この点に関し、二次元テキスチャマップは、例示目的だけのために示されるものである。
【０００７】
オブジェクトプリミティブの各ピクセルは、オブジェクトのあらゆるビューについてテキスチャマップの単一テクセルと１対１に対応してマッピングされない場合もあることを理解されたい。
たとえば、オブジェクトは、表示画面上に表されるビューポートに近いほど、大きく見える。
オブジェクトが表示画面上で大きく見えるほど、テキスチャの表現はより詳細になる。
このように、オブジェクトが表示画面のかなり大きな部分を消費する場合、多数のピクセルが表示画面上のオブジェクトを表現するために使用され、そのオブジェクトを表す各ピクセルをテキスチャマップの単一ピクセルと１対１の対応でマッピングすることができるか、または単一テクセルを複数のピクセルにマッピングすることができる。
しかし、オブジェクトが表示画面上で比較的小さな部分を占める場合、そのオブジェクトを表現するにははるかに少ない数のピクセルが使用され、テキスチャがより荒く表される結果になり、各ピクセルを複数のテクセルにマッピングすることができる。
テキスチャがオブジェクトの小さな部分にマッピングされる場合、各ピクセルを複数のテクセルにマッピングすることが可能である。
結果得られるテクセルデータが、２つ以上のテクセルにマッピングされる各ピクセルについて計算され、通常、そのピクセルにマッピングされるテクセルの平均を表す。
【０００８】
従来のグラフィックスシステムをより具体的に示すために、図１を参照する。
図１は、既知のグラフィックスパイプライン１０を示す図である。
最初に、図１に示すグラフィックスパイプライン１０を図示する様々な代替の様式があり、図１の図は、例示だけのために提示されることに留意されたい。
【０００９】
図示のように、ホストコンピュータ２０は通常、ＡＧＰ（加速グラフィックスポート）バスまたはＰＣＩ（周辺コンポーネント相互接続）バス等の高速バスを介してグラフィックスハードウェアと通信する。
ホストインタフェース２２は通常、高速バスとインタフェースするグラフィックスハードウェアのフロントエンドに設けられる。
ホストインタフェース２２の下流に、フォーマットブロック２４が設けられる。
１つまたは複数のジオメトリアクセラレータ２６および２７、ならびに１つまたは複数のラスタライザ３０および３１がフォーマットブロック２４の下流に設けられる。
各ジオメトリアクセラレータによって生み出される、または生成される情報は、すべての

下流ラスタライザに同報通信される。
ジオメトリアクセラレータおよびラスタライザの動作、ならびにこういった複数の構成要素を操作する方法／構成は既知であるため、本明細書において述べる必要はない。
【００１０】
ラスタライザ３０および３１の下流には、テキスチャマッピングハードウェア３４、フラグメントプロセッサ３６、Ｚバッファ３８、および混合ハードウェア４０がある。
これら構成要素それぞれの機能および動作は既知であるため、本明細書において述べる必要はない。
しかし、知られているように、テキスチャマッピングハードウェアシステムは通常、レンダリング中のオブジェクトに関連付けられたテキスチャを表すデータを格納するローカルメモリサブシステム５０を含む。
【００１１】
テキスチャマッピングハードウェアの下流には、表示合成部５２、表示タイミング部５４、デジタル／アナログ変換器（ＤＡＣ）５６、およびディスプレイ５８を含むサブシステムがある。
知られているように、表示合成ハードウェア５２は、異なるオブジェクト／プリミティブ層を処理して、表示する所与のピクセルの色を決定する。
【００１２】
図１のもののようなグラフィックスパイプライン１０の一般的なアーキテクチャおよびデータフローと一致して、多重／並列ラスタライザおよび多重／並列テキスチャマッピングサブシステムを備え、各ラスタライザおよびテキスチャマッピング構成要素が専用ローカルメモリと通信する従来技術によるシステムが知られている。
【００１３】
ラスタライズ処理ハードウェアおよびテキスチャマッピングハードウェアを組み合わせた他のシステムも知られている。
かかるシステムでは、多重／並列結合ラスタライザ／テキスチャマッピングサブシステムがあり得る。
かかるシステムは通常、各ラスタライザ／テキスチャマッピング構成要素ごとに専用メモリを備えていた。
知られているように、かかるシステムは、ディスプレイをセグメント化し、ディスプレイの部分部分を、処理のために異なるラスタライザ／テキスチャマッピング構成要素に専用としていた。
しかし、セグメントの境界に延出するプリミティブを処理するために、この種の並列性を実施するシステムはしばしば、テキスチャマップおよび他のデータを別個のメモリにまたがって複製するため、容量および帯域双方に問題が生じた。
換言すれば、先に述べたテキスチャマップは分離することができない。
むしろ、テキスチャデータは、分離されたディスプレイセグメントまたは領域それぞれに複製される。
帯域の問題は、データがメモリから検索される（すなわち、バースト読み取り）際の粒度によってさらに悪化した。
【００１４】
【発明が解決しようとする課題】
本発明は、データおよび命令の流れを少なくとも１つの機能ユニットに配向するシステムおよび方法を広く対象とする。
【００１５】
【課題を解決するための手段】
複数のノードを定義するシステムの構成要素の一実施形態において、各ノードの一部を形成するキューネットワークマネージャ（ＱＮＭ）を提供する。
本実施形態では、ＱＮＭは、複数のノードの間での相互通信をサポートする、ネットワークへのインタフェースと、ノード内の機能ユニットにメッセージを渡すように構成される

インタフェースと、メッセージおよびプログラム可能命令のうちの少なくとも一方を格納するように構成されるランダムアクセスメモリ（ＲＡＭ）と、プログラム可能命令の内容に基づいて、機能ユニットの動作態様を制御するように構成されるロジックとを備える。

【００１６】

複数のノードを定義するシステムの構成要素の別の実施形態では、データおよび命令の流れを少なくとも１つの機能ユニットに配向する方法を提供する。

本方法は、第１のインタフェースを通して、リモートノードからプログラム可能命令を受け取ることと、プログラム可能命令をランダムアクセスメモリ（ＲＡＭ）に格納することと、プログラム可能命令の内容を用いて、機能ユニットの動作を構成することとを含む。

【００１７】

本明細書に組み込まれ、本明細書の一部をなす添付図面は、本発明のいくつかの態様を示すと共に、説明と併せて、本発明の原理を説明する役割を果たすものである。

【００１８】

【発明の実施の形態】

本発明の特定の中心的な特徴および態様を考察する前に、まず、本発明が存在して動作する環境の説明を参照する。

この点に関して、本発明は、ノードが、通信リンクを介して相互通信する機能ユニットを含む独特なノードアーキテクチャに存在し動作する。

以下の考察から分かるように、このノードアーキテクチャの独特の設計は、帯域集約的な動作環境において特に効率的なパフォーマンスを提供する。

さらに、本明細書に述べる実施形態では、通信リンクは直列リンクである。

しかし、本発明の範囲および精神に矛盾せずに他の（すなわち、非直列）通信リンクも実施し得ることが分かろう。

さらにまた、本発明の特定の態様については、すぐ下に述べるシステム環境と併せて考察する。

しかし、これら本発明の特徴については、本発明が存在し動作する環境を考察した後に明確にする。

【００１９】

図２（Ａ）を参照して、２つのノード１０２および１０４を有するシステムを示す。

２つのノード１０２および１０４は、複数の直列リンク１０６を介して相互に通信する。

直列リンクの数は、直列リンクの速度およびシステムの帯域要件に応じて、システムごとに異なり得る。

上述したように、テキスチャマッピングは、グラフィックスシステムに高いメモリ帯域要求を課す。

したがって、コンピュータグラフィックスシステムのようなシステムでは、より低い帯域のシステムよりも一般により多くの通信リンク１０６が望まれる。

【００２０】

図２（Ｂ）を参照して、４つのノード１１０、１１２、１１４、および１１６を有するシステムを示す。

図２（Ａ）の２ノードシステムのように、４つのノード１１０、１１２、１１４、１１６は、複数の直列リンク（たとえば、１１８）を介して相互に通信する。

しかし、４ノードシステムでは、任意の２つの所与のノードを相互接続する直列リンクの数は低減する。

この点に関し、システムがより多くの機能処理ノードに分割されるほど、ノードのそれぞれの間の相互接続に必要な直列リンクは少なくなる。

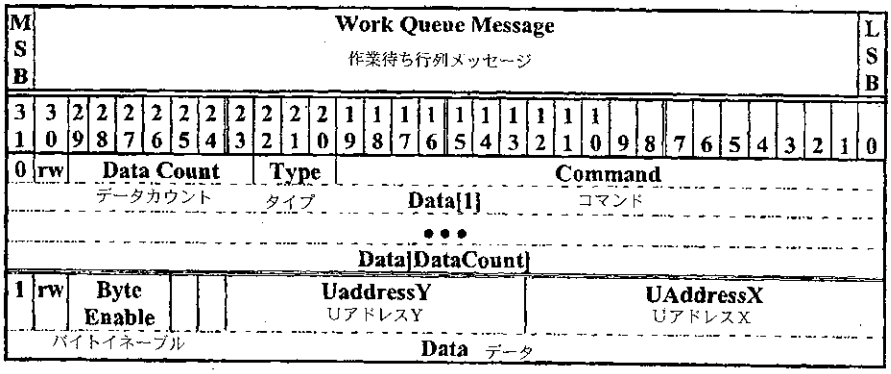
図２（Ａ）および図２（Ｂ）に提示する抽象的な例では、図２（Ａ）の２ノードシステムのシステム帯域のサポートにｎ個の直列リンクが必要であるならば、数学的にｎ／３個の直列リンクが図２（Ｂ）の４ノードシステムにおけるノードのそれぞれの間の相互接続に使用される。

【００２１】

本発明を理解するために、図２（Ａ）および図２（Ｂ）から、帯域集約的な処理システムは機能ノードに分割することが可能であるため、システム全体の機能を実行するように協働するように機能ノードが設計されることが観察される。
さらに、機能ユニットは複数の直列リンクを介して相互に通信する。
【００２２】
この基本的なアーキテクチャを要約したが、次に、グラフィックス処理システムの機能ユニットの１つの可能な構造を示す図面、および機能ユニットが相互に通信してデータを転送し、それぞれの動作機能を協働して実行する全体的な方法を参照する。
この点に関し、高性能グラフィックス処理システムに使用可能なノード１２０を示す図３を参照する。
図示のように、かかるノード１２０は、統合されたジオメトリアクセラレータ１２２、ラスタライザ１２４、およびキャッシュ１２６を備えることができる。
ジオメトリアクセラレータ１２２、ラスタライザ１２４、およびキャッシュ１２６の一般的な機能動作は、既知のシステムのものと同様である。
したがって、この機能動作を本明細書において説明する必要はない。
【００２３】
次に、所与のグラフィックス処理システム内にあることができる相互接続された複数のノードを示す図４を参照する。
図４は、ノードの作成がチップの区分化に殆ど依存しないことを示す。
代わりに、チップの統合およびチップの分割は、ピンカウント、統合の経済性、スケーラビリティ（拡張性）粒度等の要因によって導かれる。
スケーラビリティ粒度に関しては、チップ当たり１つのノードがある場合、システムは一度に１つのノードでスケーリング（拡張縮小）可能なことが分かろう。
しかし、チップ当たり４つのノードがある場合、システムは、好ましくは、一度に４つのノードでスケーリング可能である。
図には、４個のチップ１５２、１５４、１５６、および１５８を示す。
各チップは２つのノードを含む。
たとえば、チップ１５２は、ノード１６２および１６６を含む。
ノード１６２はＲＡＭ１６４と通信し、ノード１６６はＲＡＭ１６８と通信する。
したがって、かかる実施形態は、２つのノードの増分でスケーリング可能である。
【００２４】
図４はまた、チップ１５２とは別個のものとしてＲＡＭ要素１６４および１６８を示す。
一実施形態では、別個の専用ＲＡＭデバイスがノードごとに存在しうる。
かかる手法は、より細かい粒度を提供し、よりよい効率をもたらすとともに、内部機能設計の複雑さを低減する。
しかし、別の実施形態（図示せず）では、単一のＲＡＭデバイスを利用することができる。
しかし、かかる実施形態の単一ＲＡＭデバイスは、個々の区分が個々のノードに割り当てられるように分割することが可能である。
個々のノードに対して、単一のメモリデバイスを割り当てるか、それとも単にメモリデバイスの区分を割り当てるかについての設計上の判断は、帯域要件等様々な設計に固有の考慮事項に基づいて行うことができる。
さらに別の実施形態では、接続ノードと同じチップにＲＡＭを組み込んでもよい。
【００２５】
図４にさらに示すように、各ノードは複数（ｐ）の直列リンクを通して他のノードに接続される。
先に述べたように、各種ノードを相互接続する直列リンクの数は帯域要件によって導き出されるため、システムごとに異なりうる。
所与のグラフィックスシステムでは、これら機能の並列性によって得られるパフォーマンスを強化するために、いくつかのジオメトリアクセラレータ／ラスタライザ（ＧＡ／ＲＡ

）ノードが設けられる。
ノードによっては、これら機能、またはこれら機能の少なくとも一部を他の機能とグループ化することが可能なものもある。
【００２６】
次に、好ましい実施形態のノードをより詳細に示す図５を参照する。
この図では、ノード１８０は特に、階層タイラ１８４と共にジオメトリアクセラレータ１８３を備えることができ、これらは共に正確な領域選択を行うために設けられる。
パラメトリック補間を有する階層ラスタライザ１８５、シェーダ１８６、およびｚ／混合部１８７の各要素は、簡単なハードウェアが多くのパラメータを補間できるようにする。
好ましくは、シェーダ１８６は、パラメトリックテキスチャリングをサポートし、ｚ／混合部１８７はｚ検定をサポートする。
【００２７】
ノード１８０に対してローカルなＲＡＭ１７５を管理するために、オンボードキャッシュを有するメモリコントローラ１８８も設けられる。
画面リフレッシュユニット（ＳＲＵ）１８９も図示される。
ＳＲＵ１８９は、表示合成機能を含み、画面にＲＧＢ値を合成するように動作する。
上記要約した構成要素１８４、１８５、１８６、および１８７の一般的な機能および動作は一般に既知であり、当業者には理解されるはずである。
したがって、本明細書において説明する必要はない。
【００２８】
図５は、ネットワークインタフェース１８１および待ち行列ネットワークマネージャ（ＱＮＭ）１８２も示す。
ＱＮＭ１８２は、作業待ち行列（後述）を操作するとともに、ノード１８０の機能ユニット（１つまたは複数）にデータおよび命令の流れを配向するようにプログラムされたプロセッサである。
後述するように、ＱＮＭ１８２のプログラム可能性により、本発明により構築されるシステムの汎用性が増す。
【００２９】
ネットワークインタフェース１８１は単に、ノードネットワーク内の他のノードを相互接続する複数の直列リンクにノード１８０をインタフェースする回路である。
ＱＮＭ１８２は、作業待ち行列１９０（後述）の使用を通して、異なるノード／機能ユニットを共にインタフェースするように設計される。
この点に関して、ＱＮＭ１８２は、並列機能ユニットに配向するために作業待ち行列１９０を区分化し照合する。
特定の直列リンクに到来するパケットのＩＤは、そのパケットを特定の論理作業待ち行列１９０に関連付ける。
作業待ち行列１９０はローカルなＲＡＭ１７５に格納され、ＱＮＭ１８２は、好ましくは、マルチスレッドをサポートするように設計される。
スレッドへの資源の区分化は、特定の用途用にプログラマによって決定される。
【００３０】
最後に、図５は、ＱＮＭ１８２の一部であるルーティング（ｒｏｕｔｉｎｇ：経路選択）機構１７８を示す。
ルーティング機構１７８は、メッセージを各種ノードにルーティングする役割を担う。
より詳細に後述するように、各メッセージは、メッセージの宛先である宛先ノードのＩＤを含む。
確立されたルーティング技術およびアルゴリズムをルーティング機構１７８内で実施して、介在するノードに到来するメッセージを向きを変え（ｒｅｄｉｒｅｃｔｉｎｇ）転送することができる。
かかるルーティングアルゴリズムは既知のものであるため、本明細書では説明を省略する。

【００３１】
図５の要素についての上記説明は、グラフィックス処理システムにおいて可能な１つのノード設計の高レベル例示の提供のみを意図する。
しかし、本発明は、特定のノード設計に制限されない、すなわち依存しない。
さらに、グラフィックス処理分野における当業者は、図５に関連付けて大まかに述べた要素の動作を認識し、理解するであろう。
したがって、本明細書では、さらなる説明は省略する。
【００３２】
次に、図６を参照する。
図６は、各種ノードの機能動作を制御するために使用される作業待ち行列１９０を示すブロック図である。
本質的に、作業待ち行列は論理的なＦＩＦＯである。
第１のノードが第２のノードの動作を引き起こすために、第１のノードは、第２のノードに連絡され作業待ち行列１９０に配置されるメッセージまたはメッセージシーケンスを生成する。
第２のノードは後に、作業待ち行列内のそのメッセージを処理するときに、第１のノードによって要求、すなわち特定される動作を実行する。
この点に関し、作業待ち行列１９０は複数のメッセージ１９２を含む。
一般に、メッセージは、ノード上にある機能ユニットへのコマンドまたは命令である。
データパケット１９４も示される。
データパケットとは、メッセージを連絡するために使用される機構であり、通信リンクを介してやりとりされる最小量のデータである。
【００３３】
待ち行列１９０およびメッセージがとりうる様々な形態があるが、好ましい実施形態の範囲および精神と一致して、好ましい実施形態のこれら待ち行列およびメッセージの形態を以下の表１ないし表６により完全に示す。
【００３４】
以下の表１は、本発明の一実施形態によるメッセージのための作業待ち行列メッセージフォーマットを示す。
以下の表１および表２に示すように、メッセージは、２つの基本フォーマットのうちの１つで一般に提供することができる。
第１のフォーマット（ビット３１を０に設定することによって定義される）では、最初にデータワードカウントおよびメッセージ内容の識別子を含む、１～１２８のコヒーレントワードを可能にする。
第２のフォーマット（ビット３１を１に設定することによって定義される）は２ワード長であり、一度に１ピクセルずつの二次元（２Ｄ）サーフェスのＤｉｒｅｃｔ　Ｄｒａｗ（直接描画）アクセスをサポートする。
【００３５】
【表１】



【００３６】
以下の表２は、表１のメッセージフィールドの全体的な動作または内容を定義する。
【００３７】
【表２】

フィールド	ビット	機能
Ｐｃ	３１	メッセージを２つの種類に分ける １－Ｐｉｘｅｌ　１ピクセルちょうどのデータワードがメッセージにつく ０－Ｃｏｍｍａｎｄ　データカウントワードがメッセージにつくのを可能にする
ｒｗ	３０	読み取り書き込み選択：０－書き込み、１－読み取り
Data Count	２９：２３	バッファリングされたメッセージにつく［０．．１２７］を含むデータワードを指定する
Type (タイプ)	２２：２０	８つのメッセージタイプから１つを選択、８つのメッセージタイプのうちの４つは独立して説明される ０ー状態　　５－ジオメトリ頂点　　７－ＱＮＭ ２～４－予約　　６－ラスタライザ頂点
Command	１９：０	タイプによって指定される空間内の演算子および物理レジスタを選択する
Uaddress Ｘ／Ｙ	２３：０	拡張未バッファアドレス空間により８Ｋ×２ＫＤｉｒｅｃｔ　Ｄｒａｗサーフェスへのアクセスが許容される
Byte Enable	２９：２６	バイトイネーブル：１は変更可能であり、［３１：２４］、［２３：１６］、［１５：８］、［７：０］に対応する
Data	３１：０	コンテキスト依存データ

Data Count: データカウント
Command: コマンド
UaddressX/Y: UアドレスX/Y
Data: データ

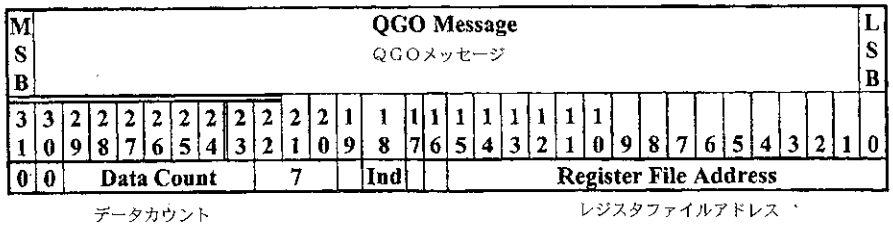
【００３８】
表１に示すように、３ビットがメッセージタイプを定義する。
表２に示すように、これら３ビットの値がゼロの場合、メッセージは「状態」タイプメッセージである。
状態タイプメッセージは、機能ユニットタイプが機能ユニットの内部レジスタにアクセスするために使用することができる。
「状態」タイプメッセージの構造およびフォーマットを以下の表３に提示する。
【００３９】
【表３】

MSB	State Message 状態メッセージ																													LSB
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	rw	Data Count データカウント							0			Category Select				Register Select レジスタ選択															
カテゴリ選択　Data　データ																														

【００４０】
表３に示すように、状態タイプメッセージ内にて定義される様々なフィールドがある。これらフィールドの動作または内容を表４において以下に定義する。
【００４１】
【表４】

フィールド	ビット	機能
ｒｗ	３０	読み取り書き込み選択：０－書き込み、１－読み取り
データカウント	２９：２３	バッファリングされたメッセージにつく［0.. １２７］を含むデータワードを指定する
カテゴリ選択	１９：１５	アクセスするレジスタのカテゴリを選択する ０×００－映像表示　０×０８－ラスタライザ　０×１２－ホストインタフェース ０×０４－ジオメトリ　０×０９－フレームバッファ　０×１３－ＢＩＮＣ ０×０７－シェーダ　０×１０－共有
レジスタ選択	１４：０	カテゴリ内でアクセスするレジスタを選択する
データ	３１：０	コンテキスト依存データ

【００４２】
ＱＮＭ１８２に配向される作業待ち行列メッセージは、ＱＮＭ１８２ファイルアドレス空間内で実行される作業待ち行列ルーチンによってオペランド解析され解釈される。
こういったルーチンは、ＱＮＭ１８２の振る舞いに影響を及ぼしうる内容の有無についてすべてのメッセージを自由に調べることができる。
好ましくは、１つのメッセージタイプがＱＮＭ１８２の単独使用のために確保される。
このメッセージタイプは、ＱＧＯメッセージ（表５がＱＧＯメッセージのメッセージ構造を示す）と呼ばれ、作業待ち行列インタプリタ（図１０参照）が、メッセージ内で示されたレジスタファイルロケーションに実行点を変更するために使用する。
これは、ＱＮＭ内で複雑な演算を行うコンパクトかつ効率的な方法を提供する。
そのロケーションにおけるメッセージリストがおそらく、作業待ち行列インタプリタへの復帰（ｒｅｔｕｒｎ：リターン）を行う（Ｔｈｅ　ｍｅｓｓａｇｅ　ｌｉｓｔ．．．ｗｏｒｋ　ｑｕｅｕｅ　ｉｎｔｅｒｐｒｅｔｅｒ．）。
【００４３】
ＱＧＯに含められる付加的なデータワードの利用は、呼び出されたルーチンの機能である。
さらに、作業待ち行列レスポンダは、データカウントをさらにパディングして、直列リンクネットワーク上のパケットフレーミングに適合することができる。
【００４４】
【表５】



【００４５】
表５に示すように、ＱＧＯタイプメッセージ内で定義される様々なフィールドがある。
これらフィールドのうちの特定のものの動作または内容を表６において以下に定義する。
【００４６】
【表６】

フィールド	ビット	機能
レジスタファイルアドレス	１５：０	ＱＮＭにとってローカルの記憶装置の６４Ｋアドレス空間内のワードロケーションを指定する。 レゾリューション後のアドレスは、ＱＮＭメッセージリストの最初である
Ｉｎｄ	１８	指定レジスタファイルアドレスに含まれる値を介しての間接レゾリューション

【００４７】
間接アドレス指定は、ポインタを介して生成側（ｐｒｏｄｕｃｅｒ）と利用側（ｃｏｎｓｕｍｅｒ）との分離をもたらすことができる。
ＱＧＯメッセージは、メッセージを受け取っている機能ユニットに新しいメッセージタイプを合成する能力を有する。
ＱＮＭ１８２は、ＱＧＯメッセージを使用して、新しいメッセージを作業待ち行列インタプリタに挿入することができる。
ＱＧＯメッセージはなんらかのサブルーチンへのポインタであるため、プログラマは、所与の機能ユニットに予め定義されていない新しいコマンドを作成するためにこれを使用することができる。
【００４８】
ＱＮＭのレジスタファイル内において、別のメッセージセットを使用して、ＱＮＭ１８２の振る舞いを実施することができる。
バッファリングされた作業待ち行列ＱＧＯメッセージも有効なＱＮＭメッセージであることができる。
ＱＮＭ１８２は、直列リンクのネットワークを介してすべてのノードのローカルな記憶装置に読み取りアクセスすることができ、またそれ自体のノード１８０のローカルな記憶装置１７５に読み取り／書き込みアクセスすることができる。
自由に使用可能なように記憶装置の一部を割り当ててもよく、またサブルーチンメッセージにより、指定情報をローカルな記憶装置１７５からロードし、またそこから格納するように明示的に指示することもできる。
【００４９】
ＱＮＭ１８２は、高度に動的な多数の待ち行列およびスタック構造を管理することも可能である。
構造の数により、レジスタファイル空間の固定部分をそれぞれに確保することは望ましくない。
また、機能ユニット間の相互作用のダイナミクスにより、任意所与の瞬間に最も多くのデータを含む構造がシフトするものと予想される。
したがって、ストレージの動的割り当てをサポートする機構を含めることができる。
【００５０】
本発明の教示により構築されるグラフィックスシステムの基本要素の高レベルの構造および動作について述べたが、システムは、テキスチャマッパおよびジオメトリアクセラレータ等、ノード内に含まれる機能ユニットが、ポイントツーポイント直列リンクを備えるノードからなるルータベースのネットワークを介して通信する新規のアーキテクチャを有することを理解されたい。
直列リンクは、発信ノードがネットワーク上の応答ノードのローカル記憶空間からデータパケットを戻すように要求するための媒体を確立する。
ローカル記憶空間は、物理的なＲＡＭであっても、またはキャッシュおよび待ち行列等、そのＲＡＭの仮想表現であってもよい。
直接リンクを持たないノードは、中間ノードを介してＱＮＭ１８２によってルーティングされる。
好ましい実施形態では、リモートノードのローカルメモリ空間への書き込みは許されない。

代わりに、資源および作業は、ノードのローカル空間への書き込みだけが必要なように分散される。
【００５１】
作業またはデータをどのようにシステムのノードに分散可能かを説明するために、ＳＲＵ１８９を考える。
ＳＲＵは、様相（画面解像度（ｄｉｓｐｌａｙ　ｒｅｓｏｌｕｔｉｏｎ）、フロントバッファ選択等）の変更を除き、作業待ち行列によって導かれない連続プロセスである。
合成プロセスは、ディスプレイを表す画像サーフェスの空間的局所性に基づいて区分化される。
このプロセスは、それぞれの空間的局所性から独立して、複数の画像サーフェスにアクセスすることができる。
ディスプレイに位置合わせされたサーフェスは、同じ空間的局所性を有することができる。
ディスプレイをアップデートするプロセスは、ディスプレイドライバのロケーションに対してローカルである。
この点において、ディスプレイに単一のドライバのみがあることが好ましい。
これにより、実際のディスプレイ上でのピクセルの位置合わせに伴う問題が回避される。
ディスプレイアップデートプロセスでは、適正なノードから合成画像にアクセスする。
【００５２】
ジオメトリアクセラレータに関しては、作業は、各種ジオメトリアクセラレータの間で均衡のとれた作業負荷が維持されるように、システム内の各種ジオメトリアクセラレータに分散することができる。
既知のように、ジオメトリアクセラレータによって演算されるデータは、画面座標ではなくユーザ座標にある。
したがって、空間的局所性に基づいてジオメトリアクセラレータ間に作業を分割しないほうが好ましい。
しかし、ロードバランスに基づいてデータを分割することが可能である。
【００５３】
この点において、各種ジオメトリエンジンは一般に、アイドルになるときかまたは作業が必要なとき、それぞれの作業待ち行列をホストインタフェースＱＮＭからポーリングする。
ホストインタフェースＱＮＭは、このポーリングアクティビティを監視して、どのジオメトリアクセラレータがより暇であるかを判断し、付加的なプリミティブをそういったジオメトリアクセラレータに割り当てることができる。
このようにして、ホストインタフェースＱＮＭは、最も効率的に作業負荷を平衡化させるように、各種ジオメトリアクセラレータに作業を分散させることができる。
本明細書においてさらに考察するように、作業の順序が維持される限り、「ラウンドロビン」様式で、または他の方法でこの作業を分散させることができる。
【００５４】
ラスタライザに関しては、作業を、ジオメトリアクセラレータにより実行された後、空間的局所性に基づいて分散させることができる。
すなわち、ジオメトリアクセラレータがデータに対するそれぞれの作業を完了した後に、そのデータを画面空間に投影することができる。
作業を画面空間に投影すると、階層タイラ１８４が、どのラスタライザがディスプレイ（画面空間）の特定のエリアに関連しているかを判定することができ、画面空間投影に基づいて、作業をラスタライザに適宜配向することができる。
【００５５】
テキスチャマッピングに関しては、ピクセル、またはさらにサブピクセルレベルに対するテキスチャパラメータを計算した後、個々のテクセルの空間的局所性を決定することができる。

一般に、テキスチャの空間的局所性と画像の空間的局所性との間に相関はない。
【００５６】
作業は、好ましくは、画像の空間的局所性、より古典的なＦＩＦＯ格納手段を通してのパイプラインパラメータによって示されるノードに分配されて、空間的局所性から独立したテクセルサンプルを読み出す機能を有するテキスチャマッピングを提供する。
【００５７】
機能ユニットが別のユニットにおいて一連のアクションを行うために、メッセージシーケンスが生成され、実施ノードのローカル記憶空間に書き込まれる。
好ましい実施形態では、また上述のように、メッセージは、指示されたアクションの実行に使用されるデータワードが続くコマンドワードからなる一連の３２ビットワードである。
影響を受ける機能ユニットは、同意されたロケーションからシーケンスを消費し、メッセージを意図されるように解釈する。
メッセージシーケンスは、作業待ち行列と呼ばれ、これについても上述した。
機能ユニットは、それぞれの入力において作業待ち行列を消費し、それぞれの出力において作業待ち行列を生成する。
この点において、生成側ノードは、それ自体のローカルメモリ空間に作業待ち行列を生成する。
対照的に、消費側ノードは、データ、命令、または他の情報（必要に応じて）を生成側のメモリから引き出す。
このようにして、コヒーレンシが簡略化された態様で維持される。
消費側が、まだ準備できていないデータまたは情報を要求する場合、そのデータまたは情報は消費側に戻されず（データまたは情報が生成側によって生成されるまで）、よって消費側がストールすることを理解されたい。
【００５８】
実行機能ユニットおよび影響を受ける機能ユニットは、同じ物理ノード内にある必要がない。
プログラム可能なポインタは、明示ノードのローカル記憶空間内の待ち行列に空間を指定し、生成側と消費側とを分離できるようにする。
独立して生成側ポインタおよび消費側ポインタを指定することで、新しい機能ユニットを古いユニットについての知識なしで挿入できるようにすることによってスケーラビリティが向上する。
機能ユニットは、ＦＩＦＯ様インタフェースを介して待ち行列にアクセスし、ノード格納アドレスの管理はＱＮＭによって行われる。
【００５９】
例として、機能ユニットによるメッセージの実行は、作業待ち行列に明示的に提供されない情報を必要とする場合がある。
たとえば、あるアクションは、テキスチャが貼り付けられた三角形プリミティブのレンダリング、またはＤＡＣを通しての画像を表す情報のシーケンシングであることができる。
機能ユニットは、そのノードのメモリサブシステムインタフェースへの専用インタフェースを介して、またはＱＮＭ１８２への専用ポートを介して、必要な情報を直接要求する。
たとえば、図５のシェーダ１８６は、メモリコントローラ１８８と直接通信することができるか、あるいは、ＱＮＭ１８２への専用ポートを介して通信することができ、ＱＮＭ１８２は、シェーダ１８６に／からのデータがローカルであるか、それともリモートであるかを判断することができる。
メモリサブシステムは、どのノードのローカルストレージが情報を含むかを判断し、リモートノードにある場合はネットワークへのインタフェース上に要求を発信し、リモートノードにない場合はそれ自体のローカル記憶装置インタフェースに対するアクセスを始める。
応答が戻る待ち時間を補償するために、複数の要求を発信することも可能である。

【００６０】
ＱＮＭに待ち行列ポインタを変更する能力を与えることで、作業待ち行列パラダイムにおけるフロー制御が行われる。
たとえば、一連の生成側から消費側作業待ち行列を得ることが望ましい場合がある。
ＱＮＭ内に含まれるレジスタファイルに格納されているルーチンにアクセスするメッセージクラスでＱＮＭの能力を増強することで、あるクラスのユニットの「ラウンドロビン」シーケンシング等（以下に提示する例では、複数のジオメトリアクセラレータの出力のシーケンシング等）多種多様な有用な振る舞いを作成することが可能であり、またはコンテキストをローカル記憶空間に保存かつ／または再格納することが可能である。
さらに、ＱＮＭのすべての振る舞いは、結果としてより高レベルのルーチンにアクセス可能な、メッセージ待ち行列の内容を転送し解釈するルーチンによって決められる。
予想される正確な振る舞いは、好ましくは、ＱＮＭにプログラムされる実施によって決定される。
【００６１】
このパラダイムの使用についての単純な例示を提供するために、作業待ち行列を介して通信する機能ユニットの概念を示す図７を手短に参照する。
具体的に、図７は、ホストインタフェース２０２、４つのジオメトリアクセラレータ（ＧＡ）２０４、２０６、２０８、および２１０、ならびに２つのシェーダ２１２および２１４を示す。
ホスト上のデバイスドライバ２２０は、コマンド／データバッファの形態のメッセージ２２２をＩ／Ｏバスに書き込み、ホストインタフェース２０２がこれを、ホストインタフェースＱＮＭ２３０に書き込まれる作業待ち行列メッセージに翻訳する。
図示の実施形態では、全ての４つの描画プリミティブが完成した後に、常に、ホストインタフェースＱＮＭ２３０は、ＥＯＣ（Ｅｎｄ－Ｏｆ－Ｃｈｕｎｋ）メッセージを合成してロードバランシングを行う。
ＥＯＣメッセージは、エンドオブチャンク（本明細書においてさらに考察する）を行うＱＧＯメッセージ（上述）である。
【００６２】
破線の矢印は、デバイス間の論理的な通信を表すことを理解されたい。
かかるデバイス間の実際の通信は、好ましくは、直接ではなく、むしろ直列リンクのネットワークを介して、また各ノードのＱＮＭを通して行われる。
【００６３】
ＱＮＭ２３０は、作業待ち行列（ＱＮＭにとってローカルなメモリにおいて）をオペランド解析するパワーアップ後に開始されるメッセージリストを実行し、状態メッセージをすべてのジオメトリアクセラレータ作業待ち行列に分配し、次のＥＯＣまで、１つのジオメトリアクセラレータ作業待ち行列に作業メッセージを分配する。
このＥＯＣは、「ラウンドロビン」シーケンスから次のジオメトリアクセラレータを選択することができるサブルーチンメッセージリストを呼び出し、アクティブなジオメトリアクセラレータ作業待ち行列にシェーダＥＯＣを合成する。
各ジオメトリアクセラレータ２０４、２０６、２０８、および２１０は、それぞれの作業待ち行列を消費し、シェーダ２１２および２１４に作業待ち行列を生成する。
ジオメトリアクセラレータによって直接実行可能ではないメッセージは、その出力に伝播されるが、他は、新しいシェーダメッセージの合成につながる描画プリミティブを表す。
ＱＮＭは、シェーダ作業待ち行列を解析するパワーアップの後、開始される他のメッセージリストをも実行し、すべてのシェーダ作業待ち行列に状態メッセージを分配し、メッセージに含まれる空間情報によって指示されるように各シェーダ作業待ち行列に作業メッセージを分配する。
【００６４】
図示のように、ネットワーク２４０との通信は各種ＱＮＭによって行われる。
たとえば、ホストインタフェースＱＮＭ２３０は、ジオメトリアクセラレータ２０４、２

０６、２０８、および２１０の作業待ち行列メッセージをネットワーク２４０に連絡する。
ジオメトリアクセラレータ２０４、２０６、２０８、および２１０の各ＱＮＭ２４２、２４４、２４６、および２４８にて、これら作業待ち行列メッセージをネットワーク２４０から受け取る。
【００６５】
概念上、任意の数の機能ユニットがノード内に存在しうるが、各機能ユニットは独立して、それぞれの作業待ち行列を生成し消費する。
所与のノードに存在する機能ユニットの実際の数は、ＱＮＭ内で必要な資源を決め、ネットワーク上の帯域消費に貢献する。
入出力の帯域は、好ましくは、ネットワークのノード部分上の利用可能な帯域を越えず、さらなる生成側および消費側にサービスを提供可能な場合であっても、ＱＮＭの複雑性に事実上上限を設ける。
異種混在ノードクラスセットでは、機能ユニットタイプの異方性分布を許し、新しいタイプの機能ユニットのネットワークへの導入を容易にするため、スケーラビリティが向上する。
【００６６】
複数の機能ユニットは、各機能ユニットの通信シーケンス順が、他の機能ユニットから独立して維持されるため、デッドロックなしで同時に動作することができる。
好ましくは、ネットワーク２４０上の通信粒度は、十分に細かいため、どの機能ユニットも他の機能ユニットのネットワークへのアクセスを大幅に遅らせることがない。
代わりに、利用不可能なパケットへの所与の要求を発した特定の機能ユニットのみがストールする。
他のシーケンスは、ネットワーク２４０に沿って独立して継続する。
作業待ち行列は、応答なしで発信を無期限に取り残すことのないように終わる構造を有する。
上記ＥＯＣ等の機構を利用する待ち行列は、先を読み、到来するＥＯＣについて警告する役割を果たすＱＮＭメッセージを挿入することによって、待ち行列の論理的終わりまたは待ち行列間の遷移を越えて要求を発信しないようにするルーチンを生成側ＱＮＭおよび消費側ＱＮＭに含む。
【００６７】
機能ユニット間で動作を同期させるという浸透したニーズもまた、待ち行列の概念で実現可能である。
同期させる各ユニットは、単純にリターンである短い待ち行列を生成する。
この文脈において、リターンは単純に、指定されたアドレスから検索される内容のないパケットであり、戻り値について解釈または実行はない。
単一の指定ユニットは、各待ち行列を順番に消費するためにジャンプする。
最後の待ち行列の検索に成功した場合、各ユニットに待ち行列を生成する。
生成は本質的に、同時に完了し、各機能ユニットが消費を完了し、それぞれの作業待ち行列を略同期して継続することが可能である。
【００６８】
作業待ち行列生成に関して、１つの機能ユニットにて生成されたメッセージシーケンスは、複数のユニットによって消費されるシーケンスに区分化することができる。
シーケンスは、すべてのターゲットユニットに複製すべき部分（状態）、およびターゲットユニットのサブセットが処理すべき部分（作業）を含むことができる。
状態部分は、最適化可能な部分（同じメッセージの初期書き込みを除去可能）および最適化不可能な部分にさらに分けることができる。
作業部分のターゲットは、その内容によって予め決めることができるか、または空の作業部分を受け取る「ラウンドロビン」シーケンスにおいて最初に利用可能な中間ユニットによって動的に決めることができる。

【００６９】
区分化するシーケンスで最適化可能な状態が見つかる場合は常に、アクティブ状態として保存され、すべての機能ユニットにマークが付けられる。
ビットアレイとしてレジスタファイルワードを操作するメッセージにより、これらアクションが容易になる。
ターゲットとする作業待ち行列に作業を配置すべき場合は常に、マークの付いたすべてのアクティブ状態は、作業待ち行列にまず配置される。
このようにして、作業を有する機能ユニットは、作業メッセージ間ですべての状態遷移を受け取ることなく、アクティブ状態を有するようになる。
【００７０】
さらに、適当な機能ユニット入力または出力における作業待ち行列を処理するＱＮＭメッセージを導入することによって、生成側、消費側、および／またはネットワーク２４０の帯域に対してより効率的なシーケンスの特定が可能である。
たとえば、頂点データ（たとえば、Ｘ、Ｙ、Ｚ、Ｒ、Ｇ、Ｂ、アルファ、Ｓ、Ｔ等の座標および値）の異なる部分をノードに通信するために複数の連続メッセージを使用した場合、単一メッセージパケットを、単一パケットでこのすべてのデータを送信するように構築することができる。
これにより、複数のメッセージでやりとりされる「ヘッダ」情報の量が削減される。
【００７１】
次に、図５のノード１８０と同様であるが、異なる機能ユニットを有するノード２８０を示す図８を手短に参照する。
この点に関して、図５は、コンピュータグラフィックスシステムにおいて特定のグラフィックス処理機能を実行するノード１８０を示した。
図５のノードのように、図８に示すノード２８０は、ネットワークインタフェース２８１、ＱＮＭ２８２、メモリコントローラ２８８、メッセージルーティング機構２７８、および専用ＲＡＭ２７５を備える。
ノード２８０は、ホストインタフェースユニット２９０およびＶＧＡ／ＲＯＭ２９２も示す。
【００７２】
このノード２８０またはこのようなノードは、ホストコンピュータとの他のノードの初期化および通信を調整するように動作する。
機能ユニット２９０は、ホストインタフェースとしての役割を果たし、ホストと通信するように動作する。
スタートアップすると、システム内の他のノードは、ノード２９０と通信して命令を受け取ることができる。
すなわち、スタートアップすると、他のノード（消費側）は、ホストインタフェース２９０からそれぞれのセットアップおよび構成情報を読み取って得ることができる。
この初期通信中、その他のノードには、それぞれの作業待ち行列を読み出す場所について命令することができる。
【００７３】
ＶＧＡ／ＲＯＭ２９２とラベルの付いたブロックは、ＱＮＭ２８２およびホストインタフェースユニット２９０の双方と通信することが可能である。
ＶＧＡ／ＲＯＭ２９２は、本発明のグラフィックスシステムおよびドライバが表示の制御を得る前に、グラフィックスＢＩＯＳ（基本入出力サービス）と通信することによってグラフィックス初期化を行うことができる。
ホストと初期化および通信を、ノード２９０を通して行う特定の方法は、本発明に対する制限ではなく、当業者により実施可能である。
【００７４】
好ましい実施形態の一態様は、分散共有メモリとして見ることができる。
共有または統合メモリシステムは既知であり、メモリシステムが複数のサブシステムまた

は機能／処理ユニットによって共有される。
分散メモリシステムもまた既知であり、メモリが各種機能／処理ユニットによる使用のために分散される。
上述した分散共有メモリ態様は、共有手法および分散手法の双方からの利点を生かすものである。
まず、分散帯域（メモリ帯域ボトルネックの回避）等、分散メモリシステムからの特定の利点を提供する。
同時に、メモリ資源の共有を可能にすることによって、複数のメモリ装置間でのデータの複製を回避する。
好ましい実施形態では、メモリ装置への書き込みアクセスが、関連するノードの機能ユニットによってしか行うことができないように、メモリ装置へのアクセスは制限される。
しかし、読み取りアクセスはすべてのノードに許可される。
この制約により、複雑なコヒーレンシ方式を実施する必要がなくなる。
【００７５】
次に、図５に示すもののようなグラフィックスノードが使用しうる潜在的なメモリセグメント化方式の一例を示す図９を参照する。
しかし、本発明の範囲および精神に一致して他の多数のセグメント化方式を使用可能なことを理解されたい。
図９に示す実施形態では、グラフィックス処理機能を実行するノード１８０にとってローカルなメモリ１７５は、いくつかの専用セグメントを含むことができる。
１つのセグメント３０２は、ローカルノード１８０のＱＮＭ１８２（図５）専用である。
空間に格納可能な他のデータおよび情報の中でもとりわけ、この専用メモリ空間３０２は、複数の作業待ち行列３０３を含むことができる。
この点に関して、ＱＮＭ１８２は、消費ノードに作業待ち行列を生成する生産ノードとして機能する。
これと併せて、消費ノードは、これら作業待ち行列３０３から読み出すことが可能である。
【００７６】
メモリ１７５の他のメモリセグメントは、第１のジオメトリアクセラレータのためのセグメント３０４、第２のジオメトリアクセラレータのためのセグメント３０６、ラスタライザのためのセグメント３０８、画像サーフェスの一部を含むセグメント３１０、テキスチャサーフェスの一部を含むセグメント３１２を含むことができる。
ジオメトリアクセラレータセグメントおよびラスタライザセグメントに関して、これら構成要素が「消費側」として機能する場合、メモリセグメントは作業待ち行列を格納する必要はない（作業待ち行列を格納する必要があるのは生成側構成要素のみであるため）。
本発明の統合メモリ構造により、テキスチャサーフェスを区分化し、複数のノードにまたがって格納することが可能である。
テキスチャサーフェスを表示中の画像に空間的にマッピングすることは容易ではない範囲において、これは特に独特である。
【００７７】
したがって、システムは、各種ノードのメモリのセグメント化に基づいて作業待ち行列を変更する機構を含む。
この機構は、メモリセグメント化に関連する機能ユニットに応じて異なる。
先に述べたように、ホストインタフェースは、各種ノードのジオメトリアクセラレータの間でロードバランシングが行われるように作業をジオメトリアクセラレータに分散させる機構（特に図示せず）を含むことができる。
空間的局所性（たとえば、メモリセグメント化）に基づいて各種ラスタライザに作業に分散させる別の機構（たとえば、階層タイラ１８４）を使用することも可能である。
【００７８】
図示の実施形態（たとえば、図５）では、メモリ装置はノードが割り当てられ、ノードに

関連付けられる。
各ノードは、データを処理する少なくとも１つの機能ユニットを含む。
かかるシステムアーキテクチャには、特にコンピュータグラフィックス環境において多くの利点がある。
これら利点のいくつかとしては、統合されたサーフェス定義を提供すること、複数のメモリにテキスチャを複製することなくテキスチャの共有が可能なこと、画像サーフェスがテキスチャを定義可能なこと、コンピュータグラフィックスカード内にディスプレイリストの分散格納が可能なこと、すべてのジオメトリアクセラレータがディスプレイリストにアクセス可能なこと、グラフィックスシステムがスケーリングされる際のテキスチャ帯域のスケーラビリティ、および大きなメモリ構成のサポートが挙げられる。
大きなメモリ構成のサポートに関して、かかる構成は、テキスチャマッピング、アンチエイリアシング、およびディスプレイリスト格納に強化されたサポートを提供する。
【００７９】
既知のように、ディスプレイリストは、グラフィックスハードウェアに対してローカルに格納されるグラフィックスプリミティブのリストである。
ディスプレイリストの実施および使用は既知であり、グラフィックスハードウェアの速度がホストＣＰＵよりも高速であったときに開発された。
進化してＣＰＵの速度が向上し、ディスプレイリストの使用および実施が減り始めた。
しかし、グラフィックスハードウェアの速度および能力は、再び大半のホストＣＰＵよりも優れたものとなりつつあり、ディスプレイリストの使用および実施がより望ましくなってきている。
本発明の分散共有メモリ構造は、ディスプレイリストの実施を単純なことにすることを理解されたい。
【００８０】
好ましい実施形態の一態様によれば、共通アクセス可能なメモリを介して各種機能ノードの間でメッセージを受け渡すシステムおよび方法が提供される。
この点に関して、「メッセージ」なる用語は広義に解釈される。
ある文脈では、メッセージは単純に、渡されるかまたは共有される情報を含むことができる。
別の文脈では、メッセージは、消費側機能ユニットが消費可能な作業待ち行列に関連する。
たとえば、作業は、コンピュータグラフィックスのセグメント上でそれぞれ動作する、異なるノードの複数のジオメトリアクセラレータに割り当てることができる。
生成側であるジオメトリアクセラレータは、出力をそれぞれの自身のローカルメモリに「生成」する。
しかし、ラスタライザのＱＮＭは、複数のジオメトリアクセラレータの各出力を検索し、ラスタライザに関連するノードのメモリ空間に作業待ち行列を形成し、この作業待ち行列はラスタライザによって消費される。
【００８１】
別の文脈では、好ましい実施形態の構造は、システムの素早い拡張を容易にする、または実現する。
一例として、ジオメトリアクセラレータとラスタライザ（二次補間回路等）との動作の間に機能的に挿入される機能ユニットを追加したいものと仮定する。
この機能ユニットを含む別個のノードを追加することができる。
この機能ユニットのＱＮＭは、複数のジオメトリアクセラレータの各出力を検索し、新しい機能ユニットが消費する作業待ち行列を形成することができる。
そして、新しい機能ユニットはそれ自体の出力を生成することができ、ラスタライザ（１つまたは複数）がこの出力を検索し、演算を行う。
かかるハードウェアの拡張は、他のハードウェアを変更することなく、単純なソフトウェアアップデートにより、システムによって独自に実現可能であることが理解されよう。

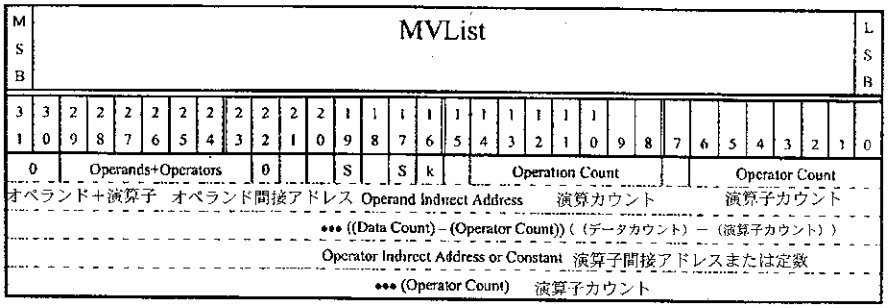
【００８２】
図６に併せて考察したように、第１のノードが第２のノードの動作を引き起こすために、第１のノードは、第２のノードに連絡され、第２のノードの作業待ち行列１９０に配置されるメッセージまたはメッセージシーケンスを生成する。
第２のノードは後に、作業待ち行列内のそのメッセージを処理するときに、第１のノードによって要求される、すなわち特定される動作を実行する。
この点に関し、作業待ち行列１９０は複数のメッセージ１９２を含む。
一般に、メッセージは、ノード上にある機能ユニットへのコマンドまたは命令である。
データパケット１９４も示される。
データパケットとは、メッセージを通信するために使用される機構であり、通信リンクを介してやりとりされる最小量のデータである。
【００８３】
ＱＮＭ１８２に配向される作業待ち行列メッセージは、ＱＮＭ１８２ファイルアドレス空間内で実行される作業待ち行列ルーチンによってオペランド解析され解釈される。
こういったルーチンは、ＱＮＭ１８２の振る舞いに影響を及ぼし得る内容がないかすべてのメッセージを自由に調べることができる。
上述したように、ＱＧＯメッセージタイプは、ＱＮＭ１８２の単独使用のために確保される。
作業待ち行列インタプリタ４４０（図１０参照）が、メッセージ内で示されたレジスタファイルロケーションに実行点を変更するためにＱＧＯメッセージを使用する。
これは、ＱＮＭ内で複雑な演算を行うコンパクトかつ効率的な方法を提供する。
その位置におけるメッセージリストがおそらく、作業待ち行列インタプリタへの復帰（ｒｅｔｕｒｎ：リターン）を行う（Ｔｈｅ ｍｅｓｓａｇｅ ｌｉｓｔ ａｔ ｔｈａｔ ｌｏｃａｔｉｏｎ ｍａｙ ｗｅｌｌ ｅｆｆｅｃｔ ａ ｒｅｔｕｒｎ ｔｏ ｔｈｅ ｗｏｒｋ ｑｕｅｕｅ ｉｎｔｅｒｐｒｅｔｅｒ．）。
【００８４】
ＱＧＯに含められる付加的なデータワードの利用は、呼び出されたルーチンの機能である。
さらに、作業待ち行列レスポンダは、データカウントをさらにパディングして、直列リンクネットワーク上のパケットフレーミングに適合することができる。
上述したように、間接アドレス指定により、ポインタ４３０（図１０参照）を介して生成側と消費側とを分離することができる。
【００８５】
メッセージ受け渡しの特定の態様を上述したが、これら態様について、図１０を参照してさらに説明する。
図１０は、複数のノード４０２、４０４、４０６、および４０８を有するシステム４００を示す図である。
もちろん、付加的なノード（図示せず）が存在してもよい。
簡単および例示のために、ノード４０２、４０４、４０６、および４０８の各々が、図５に示すノード１８０と併せて示した各種機能ユニットを備えるものと仮定する。
これら機能ユニットの一部のみがノード４０８に示されており、ノード４０２、４０４、および４０６には示されていない。
ジオメトリアクセラレータおよびラスタライザの動作の上流において、ホストから送信されたグラフィックスデータストリームが解析され、ジオメトリアクセラレータによる処理のために、各種ノード４０２、４０４、４０６、および４０８にセグメント化される。
各ノードはそれぞれのローカルメモリにのみ書き込むため、ジオメトリアクセラレータの出力結果は、各ノード内のＲＡＭに保存される。
次いで、ノード４０８は、ラスタライザ４８５に作業待ち行列を形成するにあたり、各種ジオメトリアクセラレータの結果を検索する。
もちろん、ラスタ化プロセスを各種ノードに分割してもよい。

しかし、簡単のために、１つのかかるラスタライザノードのみを示す。
【００８６】
４つのラスタライザがある、図示の場合のような状況では、各ジオメトリアクセラレータからの結果をラスタライザのうちの１つまたは複数に分散させなければならない。
図示のように、メッセージ４９２は、ジオメトリアクセラレータ４８３から生成し、作業待ち行列インタプリタ４４０によって解釈されることができ、作業待ち行列インタプリタ４４０が、どのラスタライザがメッセージを処理する必要があるかを判断する。
ローカルメモリ４７５（または他のどこか）に格納され、ラスタライザ作業待ち行列４３２のデータ宛先を指すことが可能な一連のポインタ４３０を使用することにより、ＱＮＭ４８２は、ノード４０８内のラスタライザ４８５、ならびにノード４０２、４０４、および４０６内のラスタライザに作業待ち行列４３２を容易に構築することができる。
【００８７】
同様にして、ＱＮＭ４８２は、追加の機能ユニットのシステム４００への挿入を容易にサポートすることが可能である。
上述したように、例示として、補間機能実行のために新しい機能ユニットを挿入したいものと仮定する。
さらに、この機能ユニットは、ジオメトリアクセラレータとラスタライザとの間に機能的に挿入されるものと仮定する。
ポインタは、挿入される機能ユニットによって生成されるデータのソースを反映するように更新することができるため、ポインタベースのシステムを使用すると、追加の機能ユニットを容易に挿入することができる。
作業待ち行列インタプリタ４４０を使用して、この特徴を推進することもできる。
【００８８】
図１０と同様の図である図１１を手短に参照する。
しかし、ノード４０８の内部構成要素は図から省かれている。
さらに、ノード４０６に関連する作業待ち行列５３２が示される。
特に、ラスタライザ作業待ち行列４３２および５３２は、とりわけ、各ノード４０６および４０８のジオメトリアクセラレータからの内容を含み、ラスタライザ４８５（図１０）が計算を行うとき（メッセージ制御を介して）、作業待ち行列インタプリタ（またはポインタロジック）４４０は、必要な情報を得るために、必要に応じて作業待ち行列４３２、作業待ち行列５３２、または別の作業待ち行列を指すようにポインタの値を制御する。
この点に関して、消費側機能ユニット（たとえば、ラスタライザ）はリモートノードからデータを得ることができる。
そのため、ポインタは、ローカルＲＡＭ４７５内のロケーションを指す必要はなく、リモートＲＡＭ５７５内のロケーションを指すことができる。
【００８９】
上記から分かるように、実際、作業待ち行列を消費側に供給するように働くＱＮＭは、ポインタ変更を指示するメッセージが複数の生成側からの作業待ち行列部分の検索開始まで待たない可能性が高い。
待つ場合、変更発生時に新しい生成側からパイプラインを充填する不必要な待ち時間が発生し得る。
代わりに、ポインタが「ラウンドロビン」様式で変化することを知った上で待ち時間を隠すようにＱＮＭを構成することができる。
メッセージによりサブルーチン等へのポインタ変更を命令しなければならない場合であっても、インテリジェントな生成側はその変更を予見し、新しいポインタの値を予示するメッセージを送信し、メッセージストリームにおける適宜ポイントにて制御フローの変更をトリガする。
【００９０】
本発明により構築されるシステムおよび方法の好ましい動作環境、ならびに共通メモリを通してシステムのノード間でメッセージを受け渡すために設けられる構成要素について述

べたが、次に、ＱＮＭの内部構造および動作を参照する。
先に述べたように、ＱＮＭは、相互通信ネットワークから各種ノードの機能ユニット（消費側および生成側の双方）へのデータおよび命令の流れを調整または配向する。
ＱＮＭ４８２の物理的および機能的な構造を図１０に示す。
【００９１】
次に、本発明の一実施形態により構築されＱＮＭ５８２の機能および特徴を実施する特定の内部構成要素を示すブロック図である図１２を参照する。
この点に関して、本発明の重要な特徴は、プログラム可能なＱＮＭの設計および実施である。
この文脈にて使用する「プログラム可能」なる用語は、ＱＮＭ５８２を含むノードに命令を渡すことによって、ホスト、あるいは他のリモートコンピュータまたはノードが制御可能な機能性を有するＱＮＭの実施を指す。
図１２の図中の各種ブロックは、本発明の範囲および精神に合致してハードウェア、ファームウェア、またはソフトウェアで実施し得るロジックまたは機能ユニットを表すことを理解されたい。
好ましくは、ＱＮＭは、特定用途向け集積回路（ＡＳＩＣ）の一部であり、このＡＳＩＣは、さらなる特徴および構成要素を同様に実現することができる。
【００９２】
図１２に示すように、ＱＮＭ５８２の中心には、ＲＡＭ５１０、複数の実行ユニット５４０および５４２、ならびにプログラム可能コントローラ等の制御機構５３０がある。
実行ユニット５４０および５４２は一般に、ＡＬＵ（論理演算装置）や他の同様のデバイス等、比較的単純なタスクを実行するように設計された構成要素である。
図示のように、ＲＡＭ５１０は、特に、メッセージ５１２およびプログラム可能命令５１４（他のノードから受け取る）を格納する空間を含む。
本発明の実施形態によれば、プログラム可能命令５１４は、明確な、すなわち特定の機能を実行するように実行ユニット５４０および５４２を構成するために使用することができる。
この点に関して、実行ユニット５４０および５４２は、過度に単純な設計においてではあるが、異なる様々なタスクを実行するように特に構成可能な汎用設計を有することができる。
プログラム可能命令５１４は、各種命令ユニット５４０および５４２の特定化された動作を制御または構成するために使用することができる。
【００９３】
図１２には、ネットワークインタフェース５０４および機能ユニットインタフェース５０８を含むインタフェースも示す。
一実施形態では、これらインタフェースは、それ自体特に設計することができる。
別の実施形態では、これらインタフェース５０４および５０８は、それ自体プログラム可能に構成することができる。
この点に関して、インタフェース５０４および５０８は、実行ユニット５４０および５４２のような、インタフェースとして一意に構成された実行ユニットとして見ることができる。
最も単純な形態では、インタフェース５０４および５０８は、単にＦＩＦＯ（先入れ先出し）デバイスであり得る。
【００９４】
コントローラ５３０は、ＱＮＭ５８２内の各種装置の全体の動作を制御し調整するように構成可能なロジックである。
例として、ホストコンピュータ等のリモートノードが、特定のプログラム可能命令をＱＮＭ５８２に連絡した場合、これら命令はネットワークインタフェース５０４を通して受け取られ、ＲＡＭ５１０内に格納される。
コントローラ５３０は、プログラム可能命令５１４を評価し、それに従って作用するよう

に構成可能である。
たとえば、プログラム可能命令が、実行ユニット５４０の構成が、受け取ったメッセージ５１２の特定部分またはフィールドに対して復号化動作を行うように命令するものである場合、コントローラ５３０は、プログラム可能命令５１４に従って実行ユニット５４０の構成を制御するように構成することができる。
【００９５】
プログラム可能命令５１４の形態およびフォーマットは様々であり得ることを理解されたい。
一実施形態では、プログラム可能命令５１４は単一の命令であってもよい。
別の実施形態では、プログラム可能命令５１４は、１つまたは複数の実行ユニット５４０および５４２の動作の構成に使用されるコードのより複雑なセグメントを含むことができる。
【００９６】
図１２には、リモートノードから命令または他の情報を受け取り、確実に情報がＲＡＭ５１０に適宜配向されるように構成されるロジックを表すブロック５５０も示す。
コントローラ５３０とは別個に示すが、実施形態によっては、ロジック５５０はプログラム可能コントローラ５３０に組み込まれ得ることを理解されたい。
【００９７】
ＱＮＭ５８２内の各種機能ユニットおよび装置を相互接続するように示す破線５６０は単に、これら装置間の通信ネットワークを表すことをさらに理解されたい。
この通信ネットワーク５６０は、ＱＮＭ５８２内の装置の特定の実施に応じて様々な形態または実施態様を採ることができる。
したがって、図１２における破線５６０は、必ずしも、各種装置間の直接接続として見るべきではなく、別様に言えば、本発明に対する制限として解釈すべきではない。
【００９８】
図１２は、機能ユニットインタフェース５０８と、ＱＮＭ５８２外にある各種機能ユニット５２２および５２４との間の相互通信も示す。
先に述べたように、計算システム内のノードは各種機能ユニットを含む。
機能ユニットによっては、生成側（すなわち、作業待ち行列を生成する）であるものも、また消費側（すなわち、作業待ち行列を消費する）であるものもある。
機能ユニット５０８は、各タイプの機能ユニット（すなわち、消費ユニット５２２および制作側機能ユニット５２４の双方）と通信して図１１に示されている。
もちろん、実際の実施では、機能ユニットインタフェース５０８は、各インタフェースが単一の機能ユニットのみとインタフェースするように、複数の機能ユニットインタフェースに区分化することができる。
図１２に示す実施形態は、比較的高レベルの構成要素のみを示すことを理解されたい。
【００９９】
ＱＮＭ４８２の本発明のプログラム可能性に関して、受け取られ、ＲＡＭ５１０に格納されるプログラム可能命令は、特定のタスク、機能、または動作を実行するようにＱＮＭ５８２内の実行ユニットを構成するために利用可能なことを理解されたい。
好ましくは、これらタスク、機能、または動作は、ＱＮＭ５８２を通して渡されるメッセージ５１２に対して行われる。
一実施形態では、実行ユニット（たとえば、実行ユニット５４０）は、動作してＲＡＭ５１０内に含まれるプログラム可能命令（単数または複数）を評価し、ふさわしい動作（すなわち、プログラム可能命令によって定義される動作）を実行するように特別に構成することができる。
一実施形態では、かかる命令（１つまたは複数の関連メッセージからの内容に結び付けられる）は、他の実行ユニットの動作を構成する構成情報を提供することができる。
【０１００】
このプログラム可能性およびメッセージ受け渡しプロセスを通してサポートし得る機能お

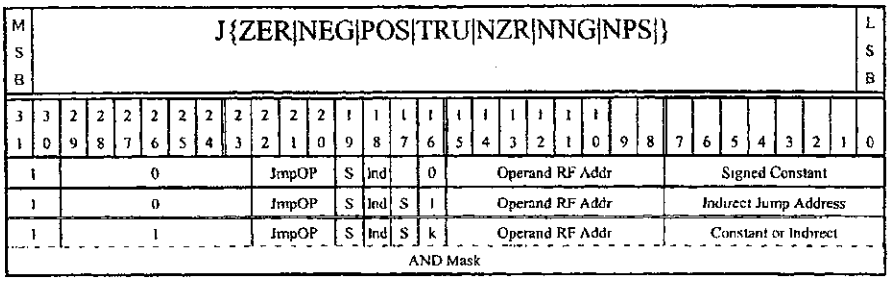
よび動作は種々様々であることが理解されよう。
たとえば、一実施形態では、ＭＯＶＥメッセージを利用して、レジスタファイルアドレス空間内の演算子からデータを取得して、レジスタファイルアドレス空間を有するオペランド宛先にコピーする柔軟な機構を提供することができる。
複数のワードは連続アドレスからのものであってもよく、または、単一の間接アドレスを介して、構造内のインデックスに関連付けられたワードを指定することもできる。
【０１０１】
別の実施形態では、ＭＯＶＥＬＩＳＴメッセージは、オペランドおよび演算子の間接アドレスの埋め込みリストを可能にする明示制御を提供することができる。
以下の表７に、かかるＭＯＶＥＬＩＳＴ（またはＭＶＬｉｓｔ）メッセージの例示的なフォーマットを提供する。
もちろん、他のメッセージおよび命令フォーマットも、本発明の範囲および精神に一致して提供することができる。
【０１０２】
【表７】



フィールド	ビット	機能
オペレーションカウント	１４：８	フレキシブル：当業者により理解されるように定義される
オペレータカウント	６：０	フレキシブル：当業者により理解されるように定義される
オペランドカウント		フレキシブル：当業者により理解されるように定義される
ＭＯＰ	２２	０・ＭＶ－演算子からオペランドをロード １・ＱＧＯレジスタファイルに格納されているサブルーチンにジャンプ；ＷＱメッセージと同じ
Ｓ	１９	オペランド間接アドレスもしくは完全なＲＦアドレスの構造／インデックス部分を選択。ｉｎｄ＝１かつｉｎｄｉｒｅｃｔ＿ａｄｄｒｅｓｓ［３１］＝０である場合かつその場合に限って当てはまる。
Ｓ	１７	オペランド間接アドレスもしくは完全なＲＦアドレスの構造／インデックス部分を選択。ｉｎｄ＝１かつｉｎｄｉｒｅｃｔ＿ａｄｄｒｅｓｓ［３１］＝０である場合かつその場合に限って当てはまる。
ｋ	１６	ｋ＝０埋め込み演算子ワードは定数；演算子は間接規則子

【０１０３】
「演算カウント」、「オペランドデータカウント」、および「演算子データカウント」フィールドは、メッセージデータカウントから独立している。
オペランドカウントは、存在する場合、メッセージデータカウントと演算子カウントとの差によって推論することができる。
指定されない、すなわち０の場合には、デフォルトで１にすることができる。
１つの演算は、上述したように拡張される一連のオペランドに個々に適用される指示関数である。
指示されたように拡張される一連の演算子は、指示された演算回数を終えるために必要な分だけ繰り返すことができる。
アドレスがデータカウント＞１を有し、かつアドレスが間接アドレスを参照する場合、データカウント連続アドレスを読み取って、データカウント間接アドレスにアクセスすることができる。

【０１０４】
ここでも、上記メッセージ構造は制限ではなく、機能は当業者によって理解される様々な方法で実施し得ることを当業者は理解するであろう。
【０１０５】
別の実施形態では、ＪＵＭＰメッセージおよび関数を提供することができる。
かかるＪＵＭＰメッセージは、オペランドの最初のワードに対してブール条件を評価し、真の場合には、演算子ワードにより指示されるアドレスで置換するか、あるいは範囲［－１２８．．１２７］の埋め込み符号付き定数を付加することによって、スレッドプログラムカウンタを変更することができる。
以下の表８に、かかるＪＵＭＰメッセージの１つのフォーマットを示す。
【０１０６】
【表８】



Operand RF Addr: オペランドＲＦアドレス
Signed Constant: 符号付き定数
Indirect Jump Addr: 間接ジャンプアドレス
Constant or Indirect: 定数または間接
AND Mask: ANDマスク

フィールド	ビット	機能
ＪｍｐＯＰ	２２：２０	ジャンプは、[－３２．．３１][－１２８．．１２７]の範囲に関係する。最上位マスク解除ビットは符号。 ０．ＪＺＥＲ－オペランドがゼロ（≡０）の場合にジャンプ １．ＪＮＥＧ－オペランドが負（＜０）の場合にジャンプ ２．ＪＰＯＳ－オペランドが正（＞０）の場合にジャンプ ３．ＪＴＲＵ－常にジャンプ ４．ＪＮＺＲ－オペランドが非ゼロ（≠０）の場合にジャンプ ５．ＪＮＮＧ－オペランドが負ではない（≧０）の場合にジャンプ ６．ＪＮＰＳ－オペランドが正ではない（≦０）の場合にジャンプ

【０１０７】
２つ以上の埋め込みデータワードが含まれる場合、ＪＵＭＰは、ＪＵＭＰテーブル（以下の表９参照）を形成することができ、ＪＵＭＰテーブルは、選択されるオペランド解析規則および単一ワードアクションメッセージのリストである。
これは、メッセージヘッダのオペランド解析を容易にする。
オペランド解析規則は、オペランドワードの一部を選択して、指定範囲内にあるかどうかをチェックする。
範囲内にある場合、オペランドワードの第２の部分に対してオフセットおよびクランピングが行われ、ＪＵＭＰテーブルに索引付けられる。
一実施形態では、ｅｎａｂｌｅ＝（ｒａｎｇｅ０≦（ｏｐｅｒａｎｄ＆ｍａｓｋ０）≦ｒａｎｇｅ１）；ｓｅｌｅｃｔ＝（ｏｐｅｒａｎｄ＆ｍａｓｋ２）－ｒａｎｇｅ２；ｉｆ（ｅｎａｂｌｅ＆＆（ｓｅｌｅｃｔ≦ｒａｎｇｅ３））ｅｘｅｃｕｔｅ ｔａｂｌｅ［ｓｅｌｅｃｔ］；ｏｔｈｅｒｗｉｓｅ ｅｘｅｃｕｔｅｔａｂｌｅ［ｒａｎｇｅ３＋１］；ｏｐｅｒａｔｉｏｎ ｔａｂｌｅ［ｓｅｌｅｃｔ］において、ｓｅｌｅｃｔは、ｍａｓｋ２内の０に対応するあらゆるビットを右シフトすることによって統合される。
ＤａｔａＣｏｕｎｔは、（ｒａｎｇｅ３－ｒａｎｇｅ２＋２）であることができる。

【０１０８】
【表９】

MSB	JUMP Table JUMPテーブル																												LSB
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Data Count >1								TblOP		S	ind		ind		Operand RF Addr								Operator RF Address						
データカウント									Message_w_0DC[0]															演算子RFアドレス						
••• Message_w_0DC[range]																															
Message_w_0DC[range+1]																															

Operand RF Addr：オペランドＲＦアドレス

フィールド　ビット

Field	Bits	Function 機能							
TblOP	22:20	Excecutes 1 word message (dc=0) selected by indicated rules.							
				mask 0	range 0	range1		mask2	range2	range3
		0	T	0	0	0	2^m	op[0]	0	mask2
		1	≠ 0	op[0]	1	~0	2^m	op[1]	0	mask2
		2	=	op[0]	op[1]	range0	2^m	op[2]	0	mask2
		3	⇔	op[0]	op[1]	op[2]	2^m	op[3]	0	mask2
		4	⇔	op[0]	op[1]	op[2]	1	0	0	0
		5	T	0	0	0	$\leq 2^m$	op[0]	op[1]	op[2]
		6	=	op[0]	op[1]	range0	$\leq 2^m$	op[2]	op[3]	op[4]
		7	⇔	op[0]	op[1]	op[2]	$\leq 2^m$	op[3]	op[4]	op[5]

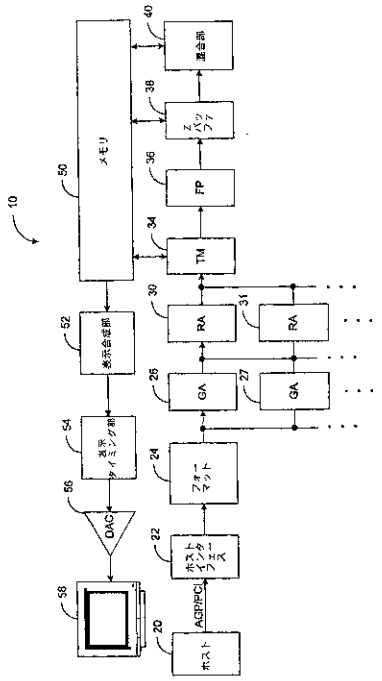
Excutes 1 word message (dc=0) selected by indicated rules:
指示される規則に従って選択される１ワードメッセージ（dc=0）を実行
注：「m」＝mask2に設定されるビット数

【０１０９】
演算子は、示すように連続レジスタファイルアドレスにおけるワードのリストを関連付けることができ、共通関数の演算子を効率的に使用するために、非関連演算子は示すデフォルト値とすることができる。
【０１１０】
次に、本発明の一態様によるＱＮＭのトップレベルの動作を示すフローチャートである図１３を手短に参照する。
この点に関して、ＱＮＭは、ホスト等のリモートノードからプログラム可能命令（またはプログラム可能命令セット）を受け取るように構成される（ステップ６１２）。
次に、このプログラム可能命令（単数または複数）は、ＱＮＭ内のＲＡＭに格納される（ステップ６１４）。
次いで、このプログラム可能命令の内容を使用して、特定のタスク、動作、または機能を実行するようにＱＮＭ内の１つまたは複数の実行ユニットを構成することができる（ステップ６１６）。
構成されると、実行ユニットは後に、ＱＮＭを通して渡されるメッセージに対してこれら指定のタスク、機能、および動作を行う（ステップ６１８）。
ＱＮＭに送信されるプログラム可能命令の内容を変化させることにより、実行ユニットを介してのＱＮＭの機能または演算を制御可能に再構成できることが理解されよう。

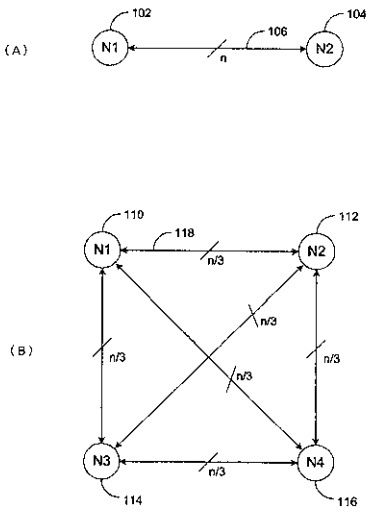
【図面の簡単な説明】
【図１】従来技術によるコンピュータグラフィックスシステムのブロック図である。
【図２】図２（Ａ）および図２（Ｂ）は、機能ノードが直列リンクを通して相互接続された本発明による高レベルノードアーキテクチャを示す図である。
【図３】ジオメトリアクセラレータと、ラスタライザと、ローカルＲＡＭと通信するキャッシュとを含む機能ノードを示す図である。
【図４】本発明の一実施形態によるマルチチップアーキテクチャでのノード構成を示す図である。
【図５】本発明の一実施形態の単一ノードにて実現可能な様々な機能要素を示す図である。
【図６】機能ユニットの演算の制御に使用される作業待ち行列を示すブロック図である。
【図７】４つの並列ジオメトリアクセラレータおよび２つのシェーダを有するノードグラフィックスシステムを通しての例示的な作業待ち行列フローを示す図である。
【図８】図５のノードと同様であるが、異なる機能ユニットを有するノードを示す図である。
【図９】図５に示すものと同様のグラフィックスノードが使用可能な潜在的なメモリセグメント化方式の一例を示す図である。
【図１０】図５のノードと同様の複数のノードを有するシステムを示すと共に、ノード間のメッセージの受け渡しの概念を示す図である。
【図１１】図５のノードと同様の複数のノードを有するシステムを示すと共に、異なるノードを有する複数の作業待ち行列を示す図である。
【図１２】待ち行列ネットワークマネージャの特定の内部構成要素および機能特徴を示す図である。
【図１３】本発明の一実施形態により構築された待ち行列ネットワークマネージャのトップレベルの動作を示すフローチャートである。
【符号の説明】
２０・・・ホスト、
２２・・・ホストインタフェース、
２４・・・フォーマット、
３８・・・Ｚバッファ、
４０・・・混合部、
５０・・・メモリ、
５２・・・表示合成部、
５４・・・表示タイミング部、
１２６・・・キャッシュ、
１７８・・・メッセージルーティング機構、
１８１・・・ネットワークインタフェース、
１８２・・・待ち行列ネットワークマネージャ（ＱＮＭ）
１８４・・・領域Ｈタイラ、
１８５・・・ラスタライザ、
１８６・・・シェーダ、
１８７・・・Ｚ／混合部、
１８８・・・キャッシュを備えたメモリコントローラ、
１９０・・・作業待ち行列、
１９２・・・メッセージ、
１９４・・・パケット、
２０２・・・ホストインタフェース、
２２０・・・ホストＣＰＵドライバ、
２２２・・・Ｃ／Ｄバッファ、
２３０・・・ホストインタフェースノードＱＮＭ、

２４０・・・直列リンクネットワーク、
２８１・・・ネットワークインタフェース、
２９０・・・ホストインタフェースユニット、
１８０・・・ＧＡ／ＲＡノード、
４０２・・・ノード１、
４３０・・・ポインタ、
４３２・・・ラスタライザ作業待ち行列、
４４０・・・作業待ち行列インタプリタ（メッセージロジック）、
４８３・・・ジオメトリアクセラレータ、
４８５・・・ラスタライザ、
５０８・・・機能ユニットインタフェース、
５１４・・・プログラム可能命令、
５２２・・・消費側機能ユニット、
５２４・・・制作側機能ユニット、
５３０・・・コントローラ、
５４０・・・実行ユニット１、

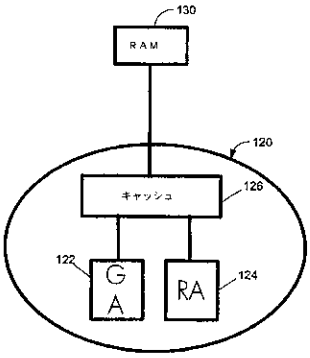
【図１】



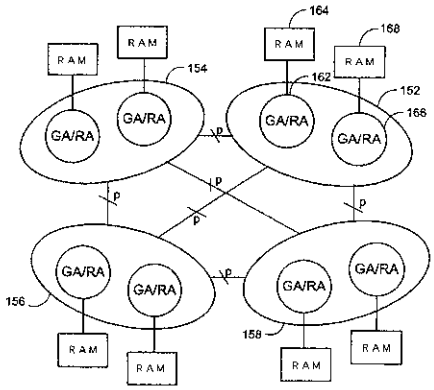
【図２】



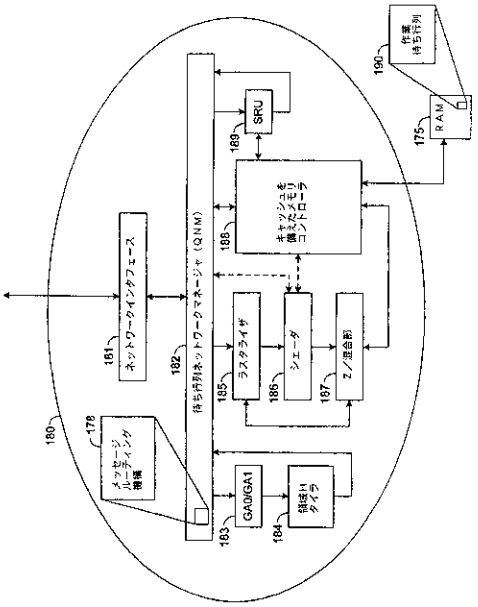
【図３】


130
RAM
120
キャッシュ
126
122
G
A
RA
124

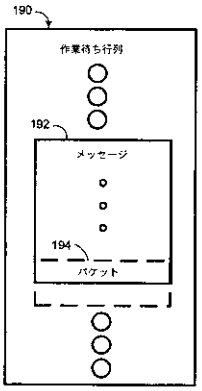
【図４】


164
168
RAM
RAM
RAM
RAM
154
162
152
168
GA/RA
GA/RA
GA/RA
GA/RA
P
P
P
P
P
GA/RA
GA/RA
GA/RA
GA/RA
156
158
RAM
RAM
RAM
RAM

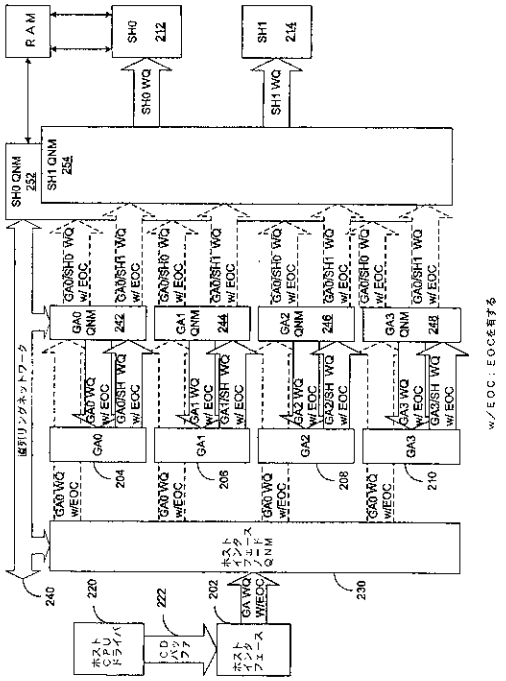
【図５】


190
作業待ち行列
175
RAM
SRU
169
キャッシュ&ローカルメモリコントローラ
188
ネットワークインタフェース
181
待ち行列ネットワークマネージャ（QNM）
182
ラスタライザ
185
シェーダ
186
Z／混合部
187
180
178
メッセージパッシング機構
183
GA0/GA1
184
割込みタイマ

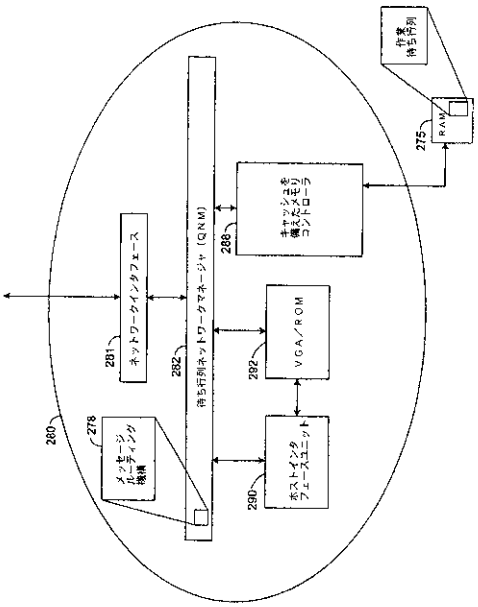
【図６】


190
作業待ち行列
192
メッセージ
194
パケット

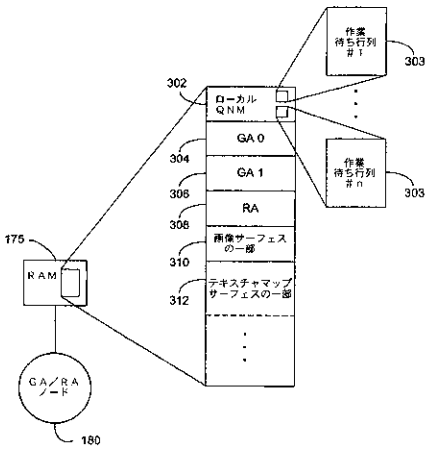
【図７】


RAM
SH0
212
SH1
214
SH0 WQ
SH1 WQ
SH0 QNM
252
SH1 QNM
254
GA0/SH0 WQ w/ EOC
GA0/SH1 WQ w/ EOC
GA0 QNM
242
GA1 QNM
244
GA2 QNM
246
GA3 QNM
248
GA0 WQ w/EOC
GA1 WQ w/EOC
GA2 WQ w/EOC
GA3 WQ w/EOC
GA0
204
GA1
206
GA2
208
GA3
210
ホスト インタフェース ノード QNM
230
ホスト CPU ドライバ
220
I/O バッファ
222
ホスト インタフェース
202
240
w/EOC：EOCを有する

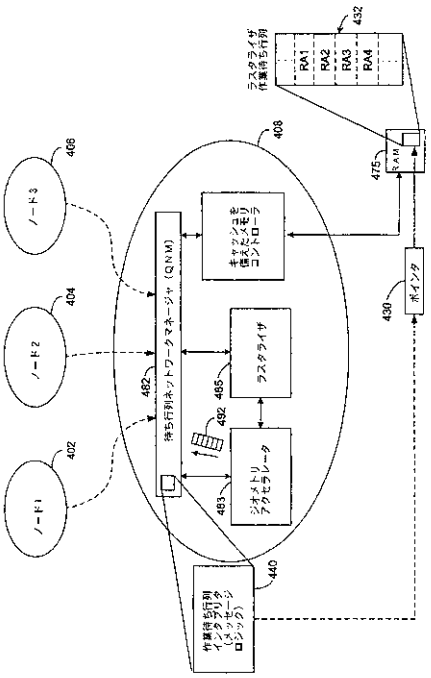
【図８】


作業待ち行列
275
RAM
キャッシュされたメモリコントローラ
288
待ち行列ネットワークマネージャ（QNM）
ネットワークインタフェース
281
282
VGA/ROM
292
メッセージパッシング機構
278
ホストインタフェースユニット
290
280

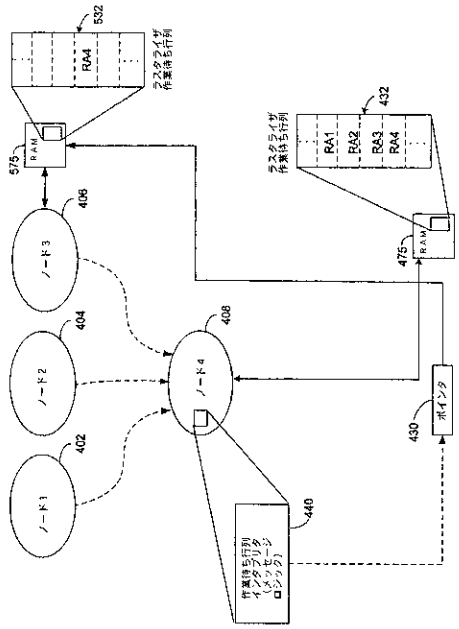
【図９】


作業待ち行列 #1
303
ローカル QNM
302
GA 0
304
GA 1
306
RA
308
画像サーフェスの一部
310
テキスチャマップサーフェスの一部
312
作業待ち行列 #n
303
175
RAM
GA/RA ノード
180

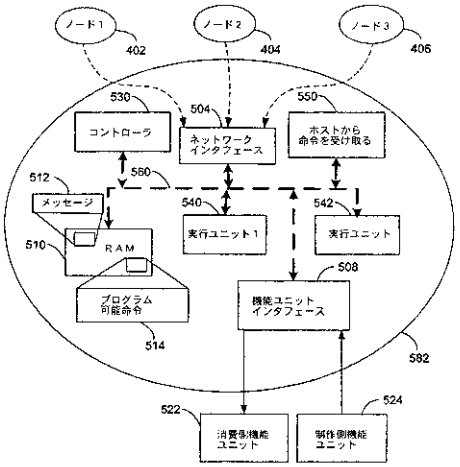
【図１０】


ラスタライザ作業待ち行列
RA1
RA2
RA3
RA4
432
408
406
404
402
ノード3
ノード2
ノード1
RAM
475
キャッシュされたメモリコントローラ
待ち行列ネットワークマネージャ（QNM）
ラスタライザ
ジオメトリアクセラレータ
482
485
492
483
ポインタ
430
440
作業待ち行列（メッセージロジック）

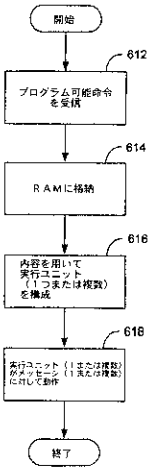
【図１１】


532
RA4
575
RAM
406
ノード３
404
ノード２
402
ノード１
408
ノード４
432
RA1
RA2
RA3
RA4
475
430
ポインタ
440

【図１２】


ノード１
402
ノード２
404
ノード３
406
530
コントローラ
504
ネットワーク
インタフェース
550
ホストから
命令を受け取る
560
512
メッセージ
540
実行ユニット１
542
実行ユニット
510
RAM
プログラム
可能命令
514
508
機能ユニット
インタフェース
582
522
消費側機能
ユニット
524
制作側機能
ユニット

【図１３】


開始
612
プログラム可能命令
を受信
614
ＲＡＭに格納
616
内容を用いて
実行ユニット
（１つまたは複数）
を構成
618
実行ユニット（１または複数）
がメッセージ（１または複数）
に対して動作
終了