

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第4562919号  
(P4562919)

(45) 発行日 平成22年10月13日 (2010.10.13)

(24) 登録日 平成22年8月6日 (2010.8.6)

(51) Int. Cl.	F I
<b>G06F 12/00 (2006.01)</b>	G06F 12/00 580
<b>G06F 12/02 (2006.01)</b>	G06F 12/02 570A
<b>G06T 1/20 (2006.01)</b>	G06T 1/20 A
<b>G06T 1/60 (2006.01)</b>	G06T 1/60 450G

請求項の数 5 (全 12 頁)

(21) 出願番号	特願2000-594101 (P2000-594101)	(73) 特許権者	591003943
(86) (22) 出願日	平成12年1月12日 (2000.1.12)		インテル・コーポレーション
(65) 公表番号	特表2002-535763 (P2002-535763A)		アメリカ合衆国 95052 カリフォル
(43) 公表日	平成14年10月22日 (2002.10.22)		ニア州・サンタクララ・ミッション カレ
(86) 国際出願番号	PCT/US2000/000776		ッジ ブレーバード・2200
(87) 国際公開番号	W02000/042594	(74) 代理人	100064621
(87) 国際公開日	平成12年7月20日 (2000.7.20)		弁理士 山川 政樹
審査請求日	平成18年1月17日 (2006.1.17)	(72) 発明者	ドイル, ピーター
(31) 優先権主張番号	09/231,609		アメリカ合衆国・95762・カリフォル
(32) 優先日	平成11年1月15日 (1999.1.15)		ニア州・エル ドラド ヒルズ・テンブル
(33) 優先権主張国	米国 (US)	(72) 発明者	スリニバス, アジャ
			アメリカ合衆国・95762・カリフォル
			ニア州・エル ドラド ヒルズ・フラッシ
			ング プレイス・4215
			最終頁に続く

(54) 【発明の名称】 ダイナミック・ディスプレイ・メモリを実装するための方法および装置

(57) 【特許請求の範囲】

【請求項 1】

中央プロセッサと、

メインメモリまたはシステムメモリの少なくとも1つを含む第1メモリと、

ローカルメモリを含む第2メモリと、

入力デバイスと、

前記第1メモリおよび前記入力デバイスに結合されたバスと、

グラフィックス・デバイスと、そして 前記中央プロセッサに結合され、かつ前記バスに結合され、そしてまた前記第2メモリにも結合されたメモリ制御ハブであって、前記グラフィックス・デバイスを含むメモリ制御ハブとから構成され； 上記メモリ制御ハブは第1メモリ内と第2メモリ内のオペランドにアクセスするためにグラフィックス・メモリ制御コンポーネントを具備し、かつ第1メモリ内のオペランドにアクセスするためにメモリ制御コンポーネントを具備し、

前記グラフィックス・メモリ制御コンポーネントは、グラフィックス変換テーブルを使用して、前記メモリ内のグラフィックス・オペランドが第1メモリか或は第2メモリにあるのかの決定を行い、また前記グラフィックス変換テーブルは、複数のエントリを包含し、これらエントリの夫々は、仮想アドレスとシステム・アドレスの関連付けを行い、この仮想アドレスは、前記中央プロセッサによって使用され、また上記システム・アドレスは前記第1又は第2メモリのいずれかによって使用され、前記中央プロセッサは前記グラフィックス変換テーブルを修正することを特徴とするダイナミック・ディスプレイ・メモリ

10

20

を実装するための装置。

【請求項 2】

中央プロセッサと、

メインメモリまたはシステムメモリの少なくとも 1 つを含む第 1 メモリと、

ローカルメモリを含む第 2 メモリと、

入力デバイスと、

前記第 1 メモリおよび前記入力デバイスに結合されるバスと、

グラフィックス・デバイスと、および、

前記中央プロセッサに結合され、かつ前記バスに結合され、そしてまた前記第 2 メモリにも結合されたメモリ制御ハブであって、前記グラフィックス・デバイスを含むメモリ制御ハブとから構成され、

10

上記メモリ制御ハブは上記第 1 メモリと第 2 メモリ内のオペランドにアクセスするグラフィックス・メモリ制御コンポーネントとを具備すると共に第 1 メモリ内のオペランドにアクセスするメモリ制御コンポーネントを具備し、

前記グラフィックス・メモリ制御コンポーネントは、前記中央プロセッサからのグラフィックス・オペランドの仮想アドレスを、システム・アドレスに変換するものであり、前記システム・アドレスは、前記第 1 かまたは第 2 のメモリ内のグラフィックス・オペランドのロケーションに対応していることを特徴とするダイナミック・ディスプレイ・メモリを実装するための装置。

【請求項 3】

20

中央プロセッサ； メインメモリまたはシステムメモリの少なくとも 1 つを含む第 1 メモリと、

ローカルメモリを含む第 2 メモリと、

前記中央プロセッサに結合された入力デバイスと、

前記中央プロセッサに結合された出力デバイスと、

グラフィックス・コントローラと、

バスと、そして 前記中央プロセッサに結合され、かつ前記バスに結合され、また前記グラフィックス・コントローラに結合され、そして前記第 1 メモリおよび前記第 2 メモリに結合されたメモリ制御ハブであって、前記第 1 メモリ内および前記第 2 メモリ内でオペランドにアクセスするためにグラフィックス・メモリ制御コンポーネントを有し、そして更に前記第 1 メモリ内でオペランドにアクセスするためにメモリ制御コンポーネントを有するメモリ制御ハブとから構成され、

30

前記グラフィックス・コントローラは、グラフィックス・メモリ制御コンポーネントを使用して前記第 1 メモリ内又は前記第 2 メモリ内に収められている複数のグラフィックス・オペランドにアクセスし、そして 前記中央プロセッサは、グラフィックス・メモリ制御コンポーネントを使用して前記グラフィックス・オペランドにアクセスする、ことを特徴とするダイナミック・ディスプレイ・メモリを実装するための装置。

【請求項 4】

メモリにアクセスする方法において、

中央プロセッサが仮想アドレスのオペランドにアクセスするステップと、

40

メモリ制御コンポーネントが、前記オペランドがグラフィックス・オペランドであるか否かについて決定するステップと、

前記オペランドがグラフィックス・オペランドでない場合には、前記メモリ制御コンポーネントが前記仮想アドレスに対応するシステム・アドレスのオペランドにアクセスするステップと、そして 前記オペランドがグラフィックス・オペランドである場合には、前記メモリ制御コンポーネントのグラフィックス・メモリ制御コンポーネントが前記仮想アドレスに対応するシステム・アドレスの前記第 1 メモリ又は前記第 2 メモリのいずれか 1 つにあるオペランドにアクセスするステップであって、前記第 1 のメモリはメインメモリまたはシステムメモリの少なくとも 1 つを含み、前記第 2 のメモリはローカルメモリを含むステップと、を包含することを特徴とする方法。

50

**【請求項 5】**

中央プロセッサと、

メインメモリまたはシステムメモリの少なくとも 1 つを含む第 1 メモリと、

ローカルメモリを含む第 2 メモリと、

前記中央プロセッサに結合され、かつ前記第 1 および第 2 メモリの双方に結合されたメモリ・コントローラとから構成され、

前記メモリ・コントローラはグラフィックス制御コンポーネントおよびメモリ制御コンポーネントを有し、

前記グラフィックス制御コンポーネントは、前記中央プロセッサによってアクセスされたオペランドがグラフィックス・オペランドであるか否かについての決定を行い、前記オペランドがグラフィックス・オペランドである場合には前記オペランドのアドレスを第 1 メモリ又は第 2 メモリのどちらか一方内の前記オペランドのロケーションに対応するアドレスに変換することを特徴とするダイナミック・ディスプレイ・メモリを実装するための装置。

10

**【発明の詳細な説明】****【0001】**

(発明の分野)

本発明は、グラフィックスのチップセットに関し、より具体的に述べれば、グラフィックス・メモリのマネジメントに関する。

**【0002】**

20

(関連技術の説明)

一般に、自身のメモリを制御することができるグラフィックス・サブシステムを備えることはよく知られており、この種のサブシステムは、通常、システムバスによって、CPU、メインメモリ、およびその他のデバイス、たとえば補助記憶デバイス等に接続される。この種のシステムバスは、CPU、メインメモリ、およびその他のデバイスに接続される。これによりCPUは、バスに接続されているあらゆるものにアクセスすることができる。グラフィックス・サブシステムは、しばしば、当該グラフィックス・サブシステムからのみアクセス可能な高速メモリを含んでいる。それに加えて、通常はシステムバスを介して、この種のサブシステムがメインメモリ内のオペランドにアクセスすることもある。

**【0003】**

30

こういったシステムにおいては、CPUが、グラフィックス・オペランドに対するオペレーションを実行しなければならないことも少なくない。しかしながら、これらのオペランドのオーガナイゼーションは、グラフィックス・サブシステムによって制御される。そのため、CPUがグラフィックス・サブシステムからオペランドを獲得することが必要になる。これに対して、CPUまたは関連するメモリ・マネジメント・ユニット(MMU)がグラフィックス・オペランドのオーガナイゼーションを制御することがあり、その場合にはグラフィックス・サブシステムが、そのオペレーションのためにCPUまたはMMUからデータを獲得しなくてはならなくなる。いずれの場合においても、一方のデバイスが、そのタスクを実行するために他方のデバイスに対してデータを要求しなければならないことから、ある程度の効率の低下が招かれる。

40

**【0004】**

別のシステムにおいては、CPUとグラフィックス・サブシステムが、ともにグラフィックス・オペランドのオーガナイゼーションを制御する。この種のシステムにおいては、CPUおよびグラフィックス・サブシステムが、互いにオペランドを要求する必要がなくなるが、メモリ内においてグラフィックス・オペランドの移動があった時点で互いの情報の交換が必要となり、それが行われない場合にはアクセス不能となる。その結果、グラフィックス・オペランドに対する各オペレーションに、オーバーヘッドの増加がもたらされる。

**【0005】**

図 1 は、従来技術のシステムを示している。この図を参照すると、グラフィックス・アド

50

レス変換器 100 (GAT 100) がグラフィックス・デバイス・コントローラ 120 (GDC 120) に接続されており、さらにそれがグラフィックス・デバイス 130 に接続されている。また GAT 100 は、バスに接続されており、それによってメインメモリ 160、補助記憶 170、メモリ・マネジメント・ユニット 150 (MMU 150) に接続されている。中央処理ユニット 140 (CPU 140) は、MMU 150 に接続されており、それによってメインメモリ 160 および補助記憶 170 にアクセスする。また CPU 140 は、GAT 100 に対する制御接続を有し、それによって CPU 140 による GAT 100 の制御が可能になる。メインメモリ 160 は、セグメント・バッファ 110 を包含している。

#### 【0006】

CPU 140 は、メインメモリ 160 ならびに補助記憶 170 内に記憶されているグラフィックス・オペランドに対するオペレーションを行う。これを容易にするために、MMU 150 は、各種オペランドがどこに記憶されているかについてのレコードを維持しつつ、メインメモリ 160 および補助記憶 170 をマネージする。メモリ内においてオペランドが移動させられると、MMU 150 は、そのオペランドのロケーションのレコードを更新する。GDC 120 もまた、メインメモリ 160 ならびに補助記憶 170 内に記憶されているグラフィックス・オペランドに対するオペレーションを行う。これを容易にするために、GAT 100 は、グラフィックス・オペランドがどこに記憶されているかについてのレコードを維持しており、メモリ内においてオペランドの移動があったときには、それらのレコードを更新する。その結果、CPU 140 もしくは GDC 120 が、グラフィックス・オペランドの移動をもたらすアクションを行った場合には、必ず MMU 150 ならびに GAT 100 双方のレコードが更新されなければならないことになる。MMU 150 および GAT 100 のレコードの間の一貫性を維持することは、メインメモリ 160 もしくは補助記憶 170 のいずれに対するアクセスにおいても多くのエラーに遭遇する可能性があることから、高度に同期化されたオペレーションを必要とする。

#### 【0007】

たとえば、CPU 140 が、補助記憶 170 からメモリのセグメントをメインメモリ 160 内のセグメント・バッファ 110 に移動し、それによってセグメント・バッファ 110 内の以前の内容を上書きすることがある。その種のアクションが発生したとき、MMU 150 は、そのレコードを更新して、セグメント・バッファ 110 内にはどのようなオペランドがあり、どのオペランドがセグメント・バッファ 110 から削除されたかを継続的に追跡する。これらのオペランドのうちのいずれかがグラフィックス・オペランドである場合には、CPU 140 が GAT 100 をその制御の下に置いて、GAT 100 に対して、関連する各種グラフィックス・オペランドに関するレコードを更新させる。さらに、CPU 140 がセグメント・バッファ 110 を上書きしたとき、GDC 120 がセグメント・バッファ 110 にアクセスしていると、GDC 120 は壊れたデータもしくは正しくないデータに対してオペレーションを行うことになる。

#### 【0008】

##### (発明の要約)

本発明は、ダイナミック・ディスプレイ・メモリを実装するための方法および装置である。本発明の一実施態様は、中央処理ユニットとメモリの間の仲裁に適したメモリ制御ハブである。このメモリ制御ハブは、グラフィックス・メモリ制御コンポーネントおよびメモリ制御コンポーネントを包含している。

#### 【0009】

添付図面には、限定の意図ではなく例示を目的として本発明を示した。

#### 【0010】

##### (詳細な説明)

本発明は、グラフィックス・データを使用する任意のシステムにおける改良されたグラフィックス・オペランドの処理およびオーバーヘッド・プロセッシングの除去を視野に入れ

10

20

30

40

50

ている。以下、ダイナミック・ディスプレイ・メモリを実装するための方法および装置について説明する。以下の説明においては、本発明の完全な理解の提供に資するために例示を目的として多数の具体的な詳細を示す。しかしながら、当業者であれば、これらの具体的な詳細を用いなくても本発明の実施が可能なのは明らかであろう。なお、本発明が不明瞭になることを避けるため、構造ならびにデバイスの表現についてはブロック図の形式を用いている。

#### 【 0 0 1 1 】

この明細書において「一実施形態」もしくは「実施形態」と言うときは、当該実施形態に関連して説明されている特定の特徴、構造もしくは特性が、本発明の実施形態の少なくとも1つに含まれていることを意味する。また本明細書の随所に用いている「一実施形態において」という表現は、必ずしもすべてが同じ実施形態を参照している必要がないものとする。

10

#### 【 0 0 1 2 】

図2は、システムの一実施形態を示している。CPU 210は、中央処理ユニットであり、この分野において周知である。グラフィックス・メモリ制御220が、CPU 210および、システム残部230に結合されている。グラフィックス・メモリ制御220は、システム残部230内に含まれるメモリ内のグラフィックス・オペランドのロケーションを追跡すること、およびCPU 210からのグラフィックス・オペランドの仮想アドレスをシステム残部230による使用に適したシステム・アドレスに変換することが可能なロジックを具体化する。つまり、CPU 210がオペランドにアクセスするとき、グラフィックス・メモリ制御220は、当該オペランドがグラフィックス・オペランドであるか否かを判断する。それがグラフィックス・オペランドである場合には、グラフィックス・メモリ制御220は、CPU 210によって示された仮想アドレスに対応するシステム・メモリ・アドレスを決定する。続いてグラフィックス・メモリ制御220は、適切なシステム・アドレスを使用してシステム残部230内の当該オペランドにアクセスし、CPU 210のためのアクセスを完成する。

20

#### 【 0 0 1 3 】

そのオペランドがグラフィックス・オペランドでないと判断されると、グラフィックス・メモリ制御220は、システム残部230がCPU 210によるメモリ・アクセスに適切に応答できるようにする。この種の応答は、この分野においてよく知られており、限定する意図はないが、メモリ・アクセスの完成、エラーのシグナリング、あるいは仮想アドレスから対応する物理アドレスへの変換ならびにそれによるオペランドのアクセスが挙げられる。メモリに対するCPUのアクセスには、読み出しならびに書き込みアクセスが含まれ、その種のアクセスの完成には、適切なロケーションに対するそのオペランドの書き込み、もしくは適切なロケーションからのそのオペランドの読み出しが含まれる。

30

#### 【 0 0 1 4 】

図2の装置は図3を参照することによってさらによく理解することができる。図3に示したプロセスは、開始ステップ300から始まり、CPUアクセス・ステップ310に進む。CPUアクセス・ステップ310は、CPU 210によるグラフィックス・オペランドのアクセスに関係し、CPU 210は、仮想アドレスに基づいてメモリ・ロケーションに対するメモリ・アクセスを実行することによってグラフィックス・オペランドにアクセスする。その後、プロセスはグラフィックス・マッピング・ステップ320に進み、そこでグラフィックス・メモリ制御220がマップもしくはその他の変換を行い、CPU 210によって供給された仮想アドレスからシステム残部230内における使用に適したシステム・アドレスもしくはその他のアドレスを求める。さらにプロセスは、システム・アクセス・ステップ330に進み、システム残部230が、このシステム・アドレスを使用して適切なメモリ・アクセスを実行してグラフィックス・オペランドの位置決めをし、その後、終了ステップ340においてプロセスが終了する。

40

#### 【 0 0 1 5 】

当業者にとっては明らかであろうが、図2のブロック図は、CPU 210およびグラフ

50

ィックス・メモリ制御 220 を別体のコンポーネントとして描かれている。しかしながら、CPU 210 およびグラフィックス・メモリ制御 220 を単一の集積回路の部分とすることもできる。

#### 【0016】

次に図 4 を参照すると、システムの別な実施形態がより詳細に示されている。図 4 において、CPU 410 は、MMU 420 を含み、MCH 430 に結合されている。MCH 430 は、グラフィックス・デバイス 440、アドレス・リオーダ・ステージ 450、および GTT 460 (グラフィックス変換テーブル) を含んでいる。MCH 430 には、さらにローカル・メモリ 480、メインメモリ 470、ディスプレイ 490、および I/O デバイス 496 が結合されている。ローカル・メモリ 480 は、グラフィックス・オペランド 485 を含んでおり、メインメモリ 470 は、グラフィックス・オペランド 475 を含んでいる。MCH 430 と I/O デバイス 496 は、I/O バス 493 を介して結合されている。グラフィックス・デバイス 440 および CPU 410 は、いずれもアドレス・リオーダ・ステージ 450 に対するアクセスを有する。一実施形態においては、一貫性を理由として、GTT 460 の修正が CPU 410 のみに限られ、したがって CPU 410 だけがグラフィックス・オペランドのメモリ内におけるロケーションを変更することができる。

#### 【0017】

図 4 に示したシステムのオペレーションは、図 5 に示したオペレーションの方法を参照することによってさらによく理解することができる。CPU アクセス・ステップ 510 は、CPU 410 がグラフィックス・オペランドの仮想アドレスに対してアクセスを実行すること表している。MMU プロセッシング・ステップ 520 は、MMU 420 がマップもしくはその他の変換を行い、CPU 410 によって供給された仮想アドレスから CPU 410 外のメモリのアクセスにおける使用に適したシステム・アドレスを求めること表している。ここで、CPU 410 によってアクセスされたグラフィックス・オペランドが CPU 410 内のキャッシュに収められていた場合に、MMU 420 が CPU 410 外のメモリにアクセスしないことがある。しかしながら、ほとんどのグラフィックス・オペランドはキャッシュ不可能であり、したがってメモリ・アクセスは CPU の外に向けられる。

#### 【0018】

判断ステップ 530 において MCH 430 は、MMU 420 からのシステム・アドレスがグラフィックス・メモリの範囲内に含まれるか否かをチェックする。グラフィックス・メモリの範囲は、グラフィックス・デバイス 440 による使用のために GTT 460 によってマップされたアドレスの範囲である。システム・アドレスがグラフィックス・メモリの範囲内になれば、プロセスがアクセス・ステップ 540 に進み、MCH 430 が、通常の方式に従ってそのシステム・アドレスによるメモリ・アクセスを実行する。通常これは、ある種のアドレス変換、そのアドレスが特定のメモリ・デバイスに導くか否かの判断、およびその特定のデバイスに対するアクセスを伴う。

#### 【0019】

システム・アドレスがグラフィックス・メモリの範囲内に含まれるときには、プロセスは判断ステップ 550 に進み、そのアドレスが囲い込み済み領域内に含まれるか否かをアドレス・リオーダ・ステージ 450 が判断する。アドレス・リオーダ・ステージ 450 の一実施形態は、囲い込み済み領域としてアドレス・リオーダ・ステージ 450 が使用するよう割り当てられたメモリの特定部分を区切る情報を収めた囲い込みレジスタを包含している。これらの囲い込み済み領域は、他のメモリと異なる態様においてオーガナイゼーションし、あるいはシステム・メモリの残部と何らかの形で異なるものとすることができる。一実施形態においては、囲い込み済み領域の内容がタイリングもしくは再オーガナイゼーションされるが、これはグラフィックス・オペランドに関連付けされたメモリが、矩形、方形、立体、その他の形状等の空間形式を論理的に模したタイルを形成するようにオーガナイゼーションできることを意味する。システム・アドレスが囲い込み済み領域内に含

10

20

30

40

50

まれると判断されると、リオーダ・ステップ 560 においてシステム・アドレスに対する適切なリオーダが実行される。この種のリオーダは、一般に、いくつかの単純な算術的再計算を含み、ルックアップ・テーブルの使用を通じても実行することができる。

#### 【0020】

リオーダ・ステップ 560 に続いて、マッピング・ステップ 570 において、リオーダ後のアドレスが物理アドレスにマップされる。リオーダが必要ない場合にも、同様にマッピング・ステップ 570 において、MMU 420 から供給されたシステム・アドレスが物理アドレスにマップされる。このマッピング・ステップは、通常、変換テーブルの使用を必要とし、この場合においては、GTT 460、すなわちアドレスもしくはシステム・アドレスの範囲とメインメモリまたはローカル・メモリ内の特定のロケーションとの対応関係を示すエントリを含んでいるグラフィックス変換テーブルが使用される。MCH 430 によるアクセス・ステップ 540 のメモリ・アクセスにおいても、類似の変換テーブルが使用される。最後に、アクセス・ステップ 580 において、アクセス・ステップ 540 と類似の態様に従って変換後のアドレスが使用されてアクセスが行なわれる。その後、終了ステップ 590 においてこのプロセスが終了する。

#### 【0021】

図 6 は、システムのさらに別の実施形態を示している。CPU 610 は、MMU 620 を含み、メモリ制御 630 に結合されている。メモリ制御 630 は、グラフィックス・メモリ制御 640 を含み、バス 660 に結合されている。バス 660 には、さらにローカル・メモリ 650、システム・メモリ 690、入力デバイス 680、および出力デバイス 670 が接続されている。CPU 610 がオペランドに対するアクセスを要求した後は、メモリ制御 630 が、CPU 610 から供給されたアドレスを変換し、バス 660 に結合されている任意の他のコンポーネント内のオペランドにバス 660 でアクセスすることができる。そのオペランドがグラフィックス・オペランドであった場合には、グラフィックス・メモリ制御 640 が、CPU 610 から供給されたアドレスの適切な操作ならびに変換を行って、メモリ制御 630 に関して説明したアクセスと同じ類のアクセスを行う。

#### 【0022】

図 8 は、システムのさらに別の実施形態およびグラフィックス・オペランドに対するアクセスの方法を示している。グラフィックス・オペランド仮想アドレス 805 は、CPU で実行されているプログラムから見たアドレスである。MMU 810 は、CPU の内部メモリ・マネジメント・ユニットである。一実施形態においては、それが、仮想アドレスとシステム・アドレスの対応関係を示すエントリを含んだルックアップ・テーブルを使用して仮想アドレスをシステム・アドレスに変換する。メモリ範囲 815 は、MMU 810 によってマップされたメモリの構造であり、MMU 810 が生成したグラフィックス・オペランドに関する各システム・アドレスは、このメモリ・スペースのいずれかの部分をアドレスする。図示した部分は、一実施形態において CPU がアクセスすることができるグラフィックス・メモリであり、このメモリ範囲の他の部分は、入力デバイスまたは出力デバイス等のデバイスに対応する。

#### 【0023】

グラフィックス・メモリ・スペース 825 は、グラフィックス・デバイスから見たグラフィックス・メモリの構造である。グラフィックス・デバイス・アクセス 820 は、一実施形態において、グラフィックス・デバイスが CPU によってアクセス可能なメモリの残部に対するアクセスを有していないことから、グラフィックス・デバイスがオフセット N を伴わずにメモリにアクセスすること、すなわち CPU および MMU 810 によってグラフィックス・メモリ・スペースに対するアクセスに使用される部分のオフセットを伴わずにメモリにアクセスすることを示している。メモリ範囲 815 およびグラフィックス・メモリ・スペース 825 のいずれも本質的に線形であり、これは CPU 上におけるプログラムの実行にとって、またグラフィックス・デバイスによるアクセスにとって必要な構造である（一実施形態においては、そのサイズが 64 MB になる）。

## 【 0 0 2 4 】

グラフィックス・デバイス・アクセス 8 2 0 からアドレスが与えられると、あるいは M M U 8 1 0 からメモリにアクセスするためのシステム・アドレスが与えられると、アドレス・リオーダ・ステージ 8 3 5 がそれらのアドレスに対するオペレーションを行う。アドレス・リオーダ・ステージ 8 3 5 は、囲い込みレジスタ 8 3 0 の内容とそれを照合することによって、与えられたアドレスが囲い込み済み領域内に含まれるか否かを判断する。そのアドレスが囲い込み済み領域内に含まれるときには、アドレス・リオーダ・ステージ 8 3 5 が、囲い込みレジスタ 8 3 0 内の他の情報、すなわちリオーダ済みアドレス・スペース 8 4 0 内のメモリをどのようにオーガナイゼーションするかということを指定する情報に基づいてそのアドレスを変換する。リオーダ済みアドレス・スペース 8 4 0 は、メモリと C P U もしくはグラフィックス・デバイスの間における転送レートを最適化するために、異なる方式に従ってメモリをオーガナイゼーションすることができる。オーガナイゼーションの方式は 2 つ挙げられ、一方は線形オーガナイゼーション、他方はタイリング・オーガナイゼーションである。線形オーガナイゼーションされたアドレス・スペース、たとえば線形スペース 8 4 3、8 4 9、および 8 5 8 はすべて、アドレス・リオーダ・ステージ 8 3 5 の観点から考えてメモリ内に逐次到来するアドレスを有する。

10

## 【 0 0 2 5 】

タイリングされたアドレス、たとえばタイリング済みスペース 8 4 6、8 5 2、および 8 5 5 におけるアドレスは、図 7 に示した態様に整列される。各タイルは、そのタイル内のロケーションを行ごとにカウントするアドレスを有し、全体的な構造としては、所定のタイルの各アドレスがそれに後続するタイル内のすべてのアドレスより前のアドレスになり、それに先行するタイル内のすべてのアドレスより後のアドレスになる。一実施形態においては、タイルのサイズが 2 k B に制限され、タイリング済みスペースの幅（タイル数により測定）が 2 の累乗でなければならない。タイリング済みスペース 8 4 6、8 5 2、および 8 5 5 に用いられているピッチがタイリング済みスペースの幅である。しかしながら、1 つのタイル内のすべてのアドレスが実際のオペランドに対応している必要はなく、したがってタイリング済みスペース 8 4 6、8 5 2、および 8 5 5 内の「x」によってマークされたアドレスは、実際のオペランドに対応する必要はない。それに加えて、この種の不要なタイルは、スクラッチ・メモリ・ページに対応させることもできる。当業者にとっては明らかであろうが、上記以外のサイズ、形状、および制限を用いてタイルを設計することは可能であり、またタイル内のアドレスを図 7 に示した方法と異なる方法を用いて整理することもできる。

20

30

## 【 0 0 2 6 】

メモリとグラフィックス・デバイスもしくは C P U のいずれか間のグラフィックス・オペランドの転送におけるシステム・リソースの最適な使用、もしくはほぼ最適な使用に向けて形状およびサイズを設定できるので、タイリング済みスペースは有用である。つまり、これらの形状は、グラフィックス・オブジェクトもしくは表面に対応するように設計される。わかりやすく述べれば、タイリング済みスペースは、システムのオペレーションの間に動的に割り当ておよび割り当て解除を行うことができる。タイリング済みスペース内におけるアドレスの整理は、各種の方法を用いて行うことが可能であり、それには図 7 に示した行優先（X 軸）順序だけでなく、列優先（Y 軸）順序およびその他の整理方法も含まれる。

40

## 【 0 0 2 7 】

図 8 に戻るが、リオーダ済みアドレス・スペース 8 4 0 内のアドレスに対するアクセスは、G T T 8 6 5（グラフィックス変換テーブル）と協働する G T L B 8 6 0（グラフィックス変換ルックアサイド・バッファ）を通じてなされる。G T T 8 6 5 自体は、通常、一実施形態におけるシステム・メモリ 8 7 0 内に記憶され、グラフィックス・メモリ・スペース 8 2 5 内のアドレスに割り当てられているシステム・メモリ 8 7 0 の部分の内側に記憶される必要はない。一実施形態においては、G T L B 8 6 0 および G T T 8 6 5 に、アドレスのセットとシステム・メモリ 8 7 0 もしくはローカル・メモリ 8 7 5 内

50



のロケーションのセットを関連付けするルックアップ・テーブルの形式が用いられる。この分野においてよく知られているように、TLBまたは変換テーブルは、各種の方法を用いて実装することができる。しかしながら、GTLB 860およびGTT 865は、グラフィックス・デバイスによる使用に特化されており、グラフィックス・オペランドに関するアドレスとメモリを関連付けするためにだけ使用できることから、他のTLBおよび変換テーブルとは異なる。この制約は、GTLB 860ないしはGTT 865のコンポーネントによってもたらされるものではなく、むしろGTLB 860およびGTT 865を包含するシステム設計によってもたらされる。GTLB 860は、好ましくはメモリ制御ハブ内に含められ、またGTT 865は、そのメモリ制御ハブを介してアクセス可能になる。

10

**【0028】**

システム・メモリ870は、通常、システムのランダム・アクセス・メモリを表すが、別の形式のストレージとすることもできる。また実施形態によっては、ローカル・メモリ875を含まないこともある。ローカル・メモリ875は、通常、グラフィックス・デバイスを伴う使用のための専用メモリを表し、システムが機能する上ではなくてもよい。

**【0029】**

以上の詳細な説明においては、具体的な実施形態を参照して本発明の方法および装置の説明を行ってきた。しかしながら、本発明の精神ならびに範囲はより広範であり、それから逸脱することなく、それらに対する各種の修正ないしは変更が可能であることは明らかである。したがって、本件明細書ならびに図面は、例示に過ぎず、限定を意図したものではないことを理解する必要がある。

20

**【図面の簡単な説明】**

【図1】 従来技術のグラフィックス・ディスプレイ・システムを示している。

【図2】 システムの一実施形態を示している。

【図3】 システムのオペレーションの、可能性のあるモードを表したフローチャートである。

【図4】 システムの別の実施形態を示している。

【図5】 システムのオペレーションの、可能性のあるモードを表したフローチャートである。

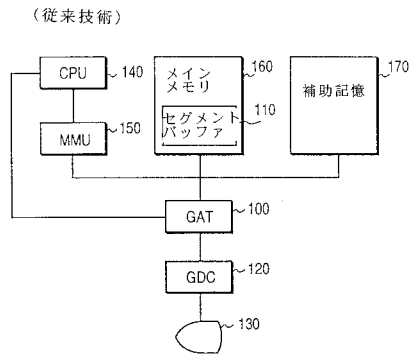
【図6】 システムの別の実施形態を示している。

30

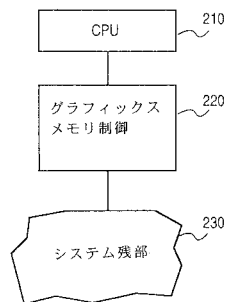
【図7】 タイリングされたメモリを示している。

【図8】 システム内におけるメモリ・アクセスを示している。

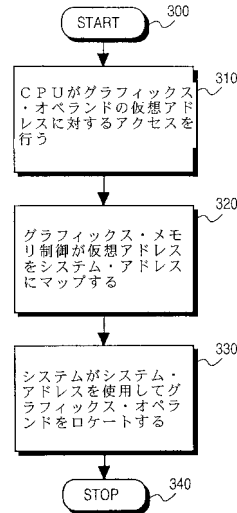
【図 1】



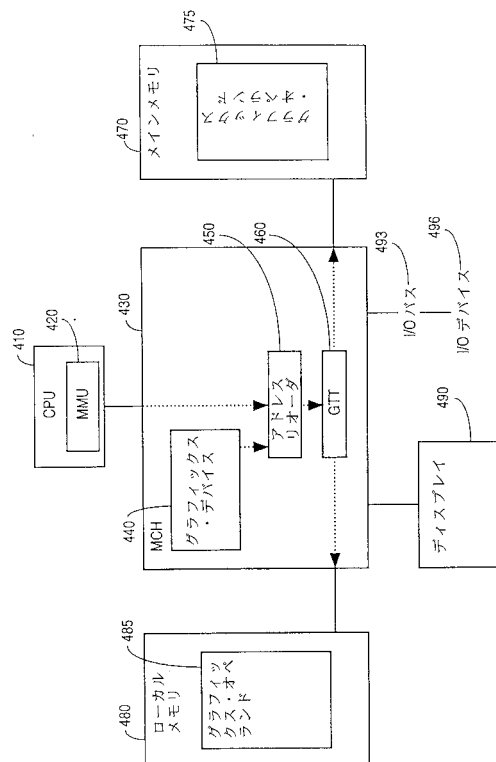
【図 2】



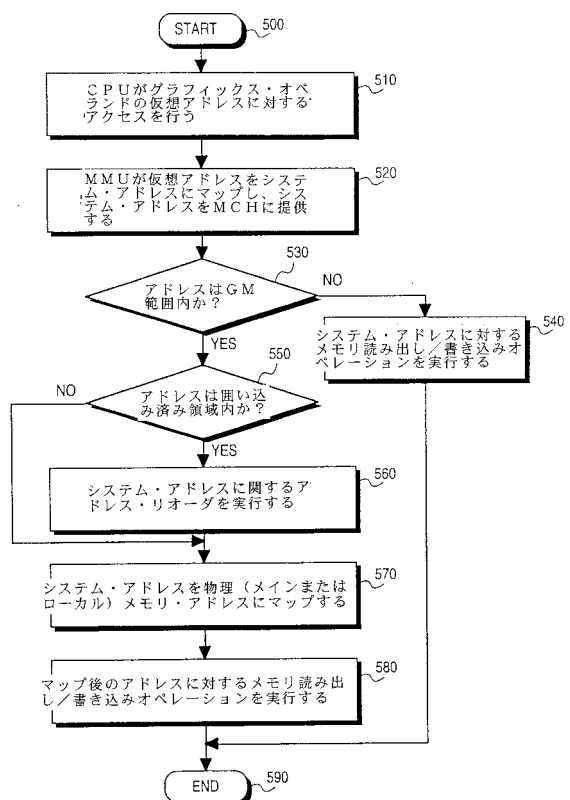
【図 3】



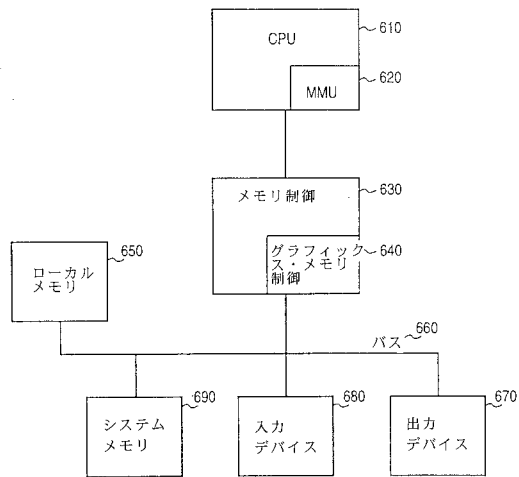
【図 4】



【図 5】



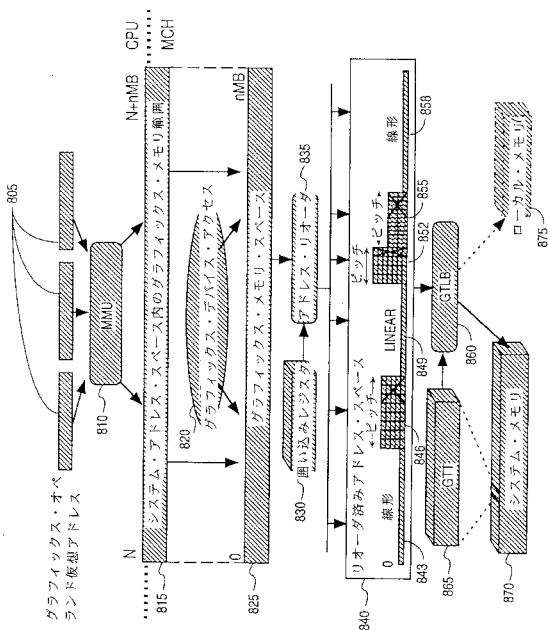
【図 6】



【図 7】

1	2	3	4	17	18	19	20	33-48
5	6	7	8	21	22	23	24	
9	10	11	12	25	26	27	28	
13	14	15	16	29	30	31	32	
49	50	51	52	65-80				81-96
53	54	55	56					
57	58	59	60					
61	62	63	64					
97	98	99	100	113-128				129-144
101	102	103	104					
105	106	107	108					
109	110	111	112					

【図 8】



---

フロントページの続き

審査官 高 橋 正 徳

(56)参考文献 国際公開第 9 5 / 0 1 5 5 2 8 ( W O , A 1 )

特開平 1 1 - 0 1 5 7 7 4 ( J P , A )

特表平 0 9 - 5 0 5 4 2 4 ( J P , A )

特開平 0 5 - 2 5 7 7 9 3 ( J P , A )

特開平 1 0 - 2 2 2 4 5 9 ( J P , A )

特開平 0 8 - 0 5 0 5 7 3 ( J P , A )

特開平 0 4 - 0 8 4 1 9 2 ( J P , A )

(58)調査した分野(Int.Cl. , D B 名)

G06F 12/00-12/06,

G06F 13/16-13/18