(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4334901号
(P4334901)

(45)発行日　平成21年9月30日(2009.9.30)　(24)登録日　平成21年7月3日(2009.7.3)

(51)Int.Cl.　F I
G06F 9/48 (2006.01)　G06F 9/46 452D

請求項の数 25 (全 44 頁)

(21)出願番号　特願2003-97175(P2003-97175)
(22)出願日　平成15年3月31日(2003.3.31)
(62)分割の表示　特願2002-79358(P2002-79358)の分割
原出願日　平成14年3月20日(2002.3.20)
(65)公開番号　特開2003-303134(P2003-303134A)
(43)公開日　平成15年10月24日(2003.10.24)
審査請求日　平成17年3月22日(2005.3.22)
(31)優先権主張番号　09/816,020
(32)優先日　平成13年3月22日(2001.3.22)
(33)優先権主張国　米国(US)

前置審査

(73)特許権者　395015319
株式会社ソニー・コンピュータエンタテインメント
東京都港区南青山二丁目６番２１号
(74)代理人　100099324
弁理士　鈴木　正剛
(74)代理人　100108604
弁理士　村松　義人
(74)代理人　100111615
弁理士　佐野　良太
(72)発明者　鈴置　雅一
東京都港区赤坂７丁目１番１号　株式会社ソニー・コンピュータエンタテインメント内

最終頁に続く

(54)【発明の名称】コンピュータ処理システム及びコンピュータで実行される処理方法

(57)【特許請求の範囲】

【請求項１】

処理システムであって、

タスク処理が可能な複数の処理デバイスを有し、

選択されたタスクの処理時に、選択された処理デバイスによって用いられるタイムバジェットを定義するための絶対タイマを有し、各選択された処理デバイスは、各々対応する周波数を有するクロック信号に応答して作動するものであり、

前記タイムバジェットは、前記選択されたタスクを前記選択された処理デバイスによって完了させるための期間であって、タスクを処理するために選択された前記処理デバイスが用いる対応した前記各々対応するクロック周波数とは独立して設定され、かつ、前記タスクの完了後に前記処理デバイスが前記タイムバジェットの期間が経過するまではタスクの処理を行わないスタンバイ状態となるよう設定されるものであり、これにより、前記処理デバイスの実際の処理速度にかかわらず、タイムバジェットを用いて前記処理デバイスの処理結果を調整する、処理システム。

【請求項２】

前記期間は、前記選択された処理デバイスの各々に対して同一である、請求項１記載の処理システム。

【請求項３】

前記各々対応する周波数はそれぞれ異なるものである、請求項１記載の処理システム。

【請求項４】

前記選択された処理デバイスのうち第１のもののパワー管理は、前記タイムバジェットに基づいて実行される、請求項１記載の処理システム。
【請求項５】
各選択されたタスクに対するタイムバジェットは、ビジー時間とスタンバイ時間とを含む、請求項４記載の処理システム。
【請求項６】
前記第１の選択された一つの処理デバイスは、前記スタンバイ時間の間は、消費パワーレベルが小さくなる、請求項５記載の処理システム。
【請求項７】
前記スタンバイ時間は、前記選択された処理デバイスの対応する処理速度に基づいて、各選択された処理デバイスに対して動的に決定される、請求項６記載の処理システム。
【請求項８】
前記タイムバジェットは、前記期間が、前記選択された処理デバイスが前記タスクの処理を完了するために必要な時間を超えるように決定されるものである、請求項１記載の処理システム。
【請求項９】
処理システムであって、タスク処理が可能な複数の処理ユニットを有し、
前記タスクを実行可能な複数のサブ処理ユニットを有し、前記サブ処理ユニットの少なくとも一つは、前記処理ユニットと通信するよう動作可能なものであり、
前記タスクのうち選択されたタスクの処理において前記サブ処理ユニットのうち選択されたサブ処理ユニットと前記処理ユニットとにより使用されるタイムバジェットを決定するタイマを有し、前記選択されたサブ処理ユニットと前記処理ユニットのそれぞれは、各々対応する周波数を備えたクロック信号に応答して動作するものであり、
前記タイムバジェットは、前記選択されたタスクを前記選択されたサブ処理ユニットによって完了させるための期間であって、前記選択されたタスクを処理するために選択された前記処理ユニットが用いる対応したクロック周波数とは独立して設定され、かつ、前記タスクの完了後に前記処理デバイスが前記タイムバジェットの期間が経過するまではタスクの処理を行わないスタンバイ状態となるよう設定されるものであり、これにより、前記サブ処理ユニットの実際の処理速度にかかわらず、タイムバジェットを用いて前記サブ処理ユニットの処理結果を調整する、処理システム。
【請求項１０】
前記複数のサブ処理ユニットは、前記処理ユニットによって制御される、請求項９記載の処理システム。
【請求項１１】
前記各々対応する周波数は互いに異なるものである、請求項９記載の処理システム。
【請求項１２】
コンピュータタスクの処理方法であって、
複数のタスクを処理するための複数の処理デバイスを用意し、前記処理デバイスのそれぞれは、各々対応する周波数を有するクロック信号に応答して動作するものであり、
前記複数の処理デバイスで用いられるタイムバジェットを確立し、前記タイムバジェットは、前記各々対応する周波数とは独立して前記複数のタスクを完了させるための期間を提供し、かつ、前記タスクの完了後に前記処理デバイスが前記タイムバジェットの期間が経過するまではタスクの処理を行わないスタンバイ状態となるよう設定されるものであり、
前記複数のタスクを処理するように前記タイムバジェットに従って前記処理デバイスを動作させ、これにより、前記処理デバイスの実際の処理速度にかかわらず、タイムバジェットを用いて前記処理デバイスの処理結果を調整する、方法。
【請求項１３】
前記タスクは、前記処理デバイスの実際の処理時間にかかわらず、前記タイム・バジェットに基づいて処理が行われる、請求項１２記載の方法。

【請求項１４】
前記生成は、前記複数の処理デバイスの処理速度とは独立してなされる、請求項１３記載の方法。
【請求項１５】
前記タイムバジェットは、ビジー時間とスタンバイ時間とを含み、かつ、前記複数の処理デバイスの少なくとも一つは、前記スタンバイ時間においてスリープモードに入る、請求項１２記載の方法。
【請求項１６】
前記スタンバイ時間は、前記処理デバイスのそれぞれに対して動的に決定される、請求項１５記載の方法。
【請求項１７】
前記タイムバジェットは、前記タスクの完了時間が、前記処理デバイスによるタスク処理に必要な時間より長いものとなるよう設定される、請求項１２記載の方法。
【請求項１８】
前記タスクにおける命令又はマイクロコードを分析し、
前記命令又はマイクロコードに少なくとも一つのオペレーションなし（ＮＯＯＰ）命令が挿入される、請求項１２記載の方法。
【請求項１９】
前記選択された処理デバイスは、その処理速度が前記他の処理デバイスの処理速度とは異なるものである、請求項１８記載の方法。
【請求項２０】
通信ネットワークを通じて互いに接続可能な複数のコンピューティングデバイスを有し、各コンピューティングデバイスは、処理エレメントを少なくとも一つ有し、この少なくとも一つの処理エレメントは、
タスクの処理が可能な処理ユニットを有し、
前記タスクを実行可能な複数のサブ処理ユニットを有し、前記サブ処理ユニットの少なくとも一つは、前記処理ユニットと通信するよう動作可能なものであり、
前記タスクのうち選択されたタスクの処理において前記サブ処理ユニットのうち選択されたサブ処理ユニットと前記処理ユニットとにより使用されるタイムバジェットを決定するタイマを有し、前記選択されたサブ処理ユニットと前記処理ユニットのそれぞれは、各々対応する周波数を備えたクロック信号に応答して動作するものであり、
前記タイムバジェットは、前記各々対応する周波数とは独立して前記選択されたタスクを完了するための期間を提供し、かつ、前記タスクの完了後に前記処理デバイスが前記タイムバジェットの期間が経過するまではタスクの処理を行わないスタンバイ状態となるよう設定されるものであり、これにより、前記サブ処理ユニットの実際の処理速度にかかわらず、タイムバジェットを用いて前記サブ処理ユニットの処理結果を調整する、処理システム。
【請求項２１】
前記各々対応する周波数は互いに異なる、請求項２０記載の処理システム。
【請求項２２】
前記複数のサブ処理ユニットの少なくとも一つと、前記処理ユニットと、は、前記サブ処理ユニットの選択された一つによって実行される処理タスクのうち選択された一つにおける命令又はマイクロコードを分析して、前記タスクのうち他のタスクの処理の完了の時間に関連して選択されたタスクの完了の時間を決定するよう動作可能である、請求項２０記載の処理システム。
【請求項２３】
前記複数のサブ処理ユニットのうち少なくとも一つの前記処理ユニットは、更に、前記命令又はマイクロコードにオペレーションなし（ＮＯＯＰ）命令を一つ以上挿入することを指示して、前記タスクのうち他のタスクの処理の完了の時間に関連して選択されたタスクの処理の完了にかかる時間を所定の時間として確立するよう動作可能である、請求項２

２記載の処理システム。
【請求項２４】
複数のタスクを処理するための少なくとも一つの処理デバイスによって用いられるコンピュータプログラムが記録された記録媒体であって、前記少なくとも一つの処理デバイスは、それぞれ周波数が設定されたクロック信号に応答して動作するものであり、前記コンピュータプログラムは、
前記少なくとも一つの処理デバイスによって使用されるタイムバジェットを確立し、前記タイムバジェットは、前記各々対応する周波数とは独立に前記複数のタスクを完了するための時間を与えるもので、かつ、前記タスクの完了後に前記処理デバイスが前記タイムバジェットの期間が経過するまではタスクの処理を行わないスタンバイ状態となるよう設定されるものであり、
前記複数のタスクを処理するために前記タイムバジェットに従って前記少なくとも一つの処理デバイスを動作させるものであり、これにより、前記少なくとも一つの処理デバイスの実際の処理速度にかかわらず、タイムバジェットを用いて前記少なくとも一つの処理デバイスの処理結果を調整する、記録媒体。
【請求項２５】
前記コンピュータプログラムは、
コンピュータタスクの選択された一つにおける命令又はマイクロコードを分析し、前記命令又はマイクロコードにオペレーションなし（ＮＯＯＰ）命令を一つ以上挿入するものである、請求項２４記載の記録媒体。
【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・プロセッサ用アーキテクチャとコンピュータ・ネットワークとに関し、広帯域環境におけるコンピュータ・プロセッサおよびコンピュータ・ネットワーク用アーキテクチャに関する。さらに、本発明は、このようなアーキテクチャのためのプログラミング・モデルに関する。
【０００２】
【従来の技術】
コンピュータおよび現今のコンピュータ・ネットワーク(オフィスのネットワークで使用されるローカル・エリア・ネットワーク(ＬＡＮ)やインターネットのようなグローバルネットワークなど)の計算用装置は、スタンド・アローン型の計算用として主として設計されてきた。コンピュータ・ネットワークを介するデータとアプリケーション・プログラム(“アプリケーション”)の共用は、これらのコンピュータおよびコンピューティング・デバイスの主要な設計目標ではなかった。これらのコンピュータとコンピューティング・デバイスは、また、様々な異なるメーカー(モトローラ、インテル、テキサス・インスツルメント、ソニーなど)によって製造された広範囲の異なるタイプのプロセッサを用いて一般に設計されたものである。これらのプロセッサの各々は、それ自身の特定の命令セットと命令セット・アーキテクチャ(ＩＳＡ)とを持っている。すなわち、それ自身の特定のセットのアセンブリ言語命令と、これらの命令を実行する主演算装置と記憶装置のための構造とを有する。プログラマは、各プロセッサの命令セットとＩＳＡとを理解してこれらのプロセッサ用のアプリケーションを書くことを要求される。今日のコンピュータ・ネットワーク上でのコンピュータとコンピューティング・デバイスには異なった種類が混在していることから、データとアプリケーションの共用及びその処理は複雑になっている。さらに、この複数種が混在する環境に対する調整を行うために、多くの場合、同じアプリケーションであっても複数のバージョンを用意することが必要となっている。
【０００３】
グローバルネットワーク、特にインターネットと接続されたタイプのコンピュータやコンピューティング・デバイスは広範囲に及ぶ。パーソナル・コンピュータ(ＰＣ)とサーバーに加えて、これらのコンピューティング・デバイスの中にはセルラー電話、移動用コンピ

ュータ、個人用情報機器(ＰＤＡ)、セット・トップ・ボックス、デジタルテレビ並びにその他の装置が含まれる。コンピュータやコンピューティング・デバイスにおいて異種製品が混在する中でのデータやアプリケーションを共用することに起因して、大きな問題が生じている。

【０００４】

これらの問題を解決するためのいくつかの手法が試みられてきた。これらの手法の中には、特に、優れたインターフェースと複雑なプログラミング手法が含まれる。これらの解決方法では、処理パワーの実質的増加の実現がしばしば要求される。また、これらの解決方法では、アプリケーションの処理に必要な時間と、ネットワークを介するデータ伝送に必要な時間とが実質的に増加してしまうという結果がしばしば生じる。

【０００５】

一般に、データは、対応するアプリケーションとは別個に、インターネットを介して伝送される。このアプローチでは、アプリケーションに対応した各セットの伝送データにアプリケーション自体をも送る必要はなくなっている。従って、このアプローチによって、必要とされる帯域幅の量は最少化されるものの、ユーザーには不満の原因となることも多々ある。つまり、クライアント側のコンピュータでは、この伝送データを利用するための適正なアプリケーション、あるいは最新のアプリケーションを入手できない事態も生じ得る。また、このアプローチでは、ネットワーク上のプロセッサによって用いられている複数の異種ＩＳＡと命令セットに対応して、各アプリケーション毎にバージョンの異なる複数のアプリケーションを用意することが要求される。

【０００６】

Ｊａｖａ（登録商標）モデルでは、この問題の解決が試みられている。このモデルでは、厳しいセキュリティ・プロトコルに準拠する小さなアプリケーション(“アプレット(applet)”)が用いられている。アプレットは、ネットワークを介してサーバー側コンピュータから送信されてクライアント側コンピュータ(“クライアント”)によって実行される。異なるＩＳＡを使用しているクライアント毎に、同じアプレットであっても異なるバージョンを送信するという事態を避ける必要があるため、すべてのＪａｖａアプレットは、クライアント側のＪａｖａ仮想マシーン上で実行される。Ｊａｖａ仮想マシーンとは、ＪａｖａＩＳＡと命令セットとを備えたコンピュータをエミュレートするソフトウェアである。しかし、このソフトウェアは、クライアント側のＩＳＡとクライアント側の命令セットにより実行される。クライアント側ではＩＳＡと命令セットが各々異なるが、与えられるＪａｖａの仮想マシーンのバージョンは一つである。したがって、複数の各アプレットごとに異なるバージョンを用意する必要はない。各クライアントでは、当該クライアントにおけるＩＳＡと命令セットに対応した適正なＪａｖａ仮想マシーンだけをダウンロードすれば、ｊａｖａアプレットを実行できる。

【０００７】

【発明が解決しようとする課題】

各々の異なるＩＳＡと命令セットに対して異なるバージョンのアプリケーションを書かなければならないという課題は解決されているものの、Ｊａｖａの処理モデルでは、クライアント側のコンピュータに対してソフトウェアの追加層が要求される。ソフトウェアのこの追加層のためにプロセッサの処理速度は著しく低下する。この速度の低下は、リアルタイムのマルチメディア・アプリケーションについて特に著しい。また、ダウンロードされたＪａｖａアプレットの中にはウィルス、処理上の誤動作などが含まれている可能性がある。これらのウィルスと誤動作はクライアントのデータベースの破損やその他の損害の原因となる可能性がある。Ｊａｖａモデルで用いられているセキュリティ用プロトコルでは、“サンドボックス(sandbox)”(ＪａｖａアプレットがそれＩ以上はデータを書き込むことができない、クライアント側のメモリ内のスペース)というソフトウェアを設けることにより、この問題の解決が試みられているとはいえ、このソフトウェア駆動型セキュリティ・モデルはその実行時に頻繁に不安定になり、より多くの処理が必要となる。

【０００８】

リアルタイムの、マルチメディア・ネットワーク用アプリケーションがますます重要なものになりつつある。これらのネットワーク用アプリケーションでは非常に高速な処理が要求される。将来、そのようなアプリケーション用として毎秒何千メガビットものデータが必要となるかもしれない。ネットワークの現今のアーキテクチャ、および、特にインターネットのアーキテクチャ、並びに、Ｊａｖａモデルなどで現在実施されているプログラミング・モデルでこのような処理速度に到達することは非常に難しい。

【０００９】

したがって、新しいコンピュータ・アーキテクチャと、コンピュータ・ネットワーク用の新しいアーキテクチャと、新しいプログラミング・モデルとが求められている。この新しいアーキテクチャとプログラミング・モデルとによって、計算上の負担が付加されることなく、ネットワークの様々なメンバー間でのデータとアプリケーションの共用という問題が解決されることが望ましい。また、この新しいコンピュータ・アーキテクチャと、プログラミング・モデルとによって、ネットワークのメンバー間でのアプリケーションとデータの共用時に生じる、固有のセキュリティ上の問題も解決されることが望ましい。

【００１０】

【課題を解決するための手段】

本発明の一実施形態においては、コンピュータと、コンピューティング・デバイスと、コンピュータ・ネットワーク（あるいはコンピュータ・ネットワークに代えて、コンピュータ・ネットワーク・システムや複数のコンピュータを備えたコンピュータ・システムというカテゴリーや形態とすることもできる）とのための新しいアーキテクチャが提供される。他の実施形態では、本発明は、これらのコンピュータ、コンピューティング・デバイスおよびコンピュータ・ネットワークのための新しいプログラミング・モデルを提供するものである。

【００１１】

本発明によれば、コンピュータ・ネットワークのすべてのメンバー(ネットワーク上のすべてのコンピュータとコンピューティング・デバイス)は共通のコンピューティング・モジュールから構成される。この共通のコンピューティング・モジュールは均一な構造を有し、好適には同じＩＳＡが使用される。ネットワークのメンバーとして、クライアント、サーバー、ＰＣ、移動用コンピュータ、ゲーム用マシーン、ＰＤＡ、セット・トップ・ボックス、電気機器、デジタルテレビ、および、コンピュータ・プロセッサを用いるその他の装置が挙げられる。均一なモジュラー構造によって、ネットワークのメンバーによるアプリケーションとデータの効率的高速処理と、ネットワークを介するアプリケーションとデータの高速伝送とが可能となる。またこの構造によって、様々なサイズと処理パワーを持つネットワークのメンバーの構成が単純化され、これらのメンバーによる処理用アプリケーションの作成が単純化される。

【００１２】

また、本発明によれば、コンピュータ・ネットワークにおいて、前記ネットワークと接続された複数のプロセッサを有し、前記プロセッサの各々が、同じ命令セット・アーキテクチャを有する複数の第１の処理ユニットと、前記第１の処理ユニットを制御するための第２の処理ユニットとを有し、前記第１の処理ユニットが、前記ネットワークを介して伝送されるソフトウェア・セルを処理するために作動可能であり、前記ソフトウェア・セルの各々が、前記命令セット・アーキテクチャと互換性のあるプログラムと、前記プログラムと関連付けられたデータと、前記ネットワークを介して伝送される前記ソフトウェア・セルのすべての間で前記ソフトウェア・セルを一意的に識別するための識別子（例えばセルの識別番号）と、を有することを特徴とするコンピュータ・ネットワークも提供される。

【００１３】

なお、本発明によれば、コンピュータ・ネットワークと接続される複数のプロセッサを有するコンピュータ・システムであって、前記プロセッサの各々が、同じ命令セット・アーキテクチャを有する複数の第１の処理ユニットと、前記第１の処理ユニットを制御するための第２の処理ユニットとを有し、前記第１の処理ユニットが、前記ネットワークを介し

て伝送されるソフトウェア・セルを処理するために作動可能であり、前記ソフトウェア・セルの各々が、前記命令セット・アーキテクチャと互換性のあるプログラムと、前記プログラムと関連付けられたデータと、前記ネットワークを介して伝送される前記ソフトウェア・セルのすべての間で前記ソフトウェア・セルを一意的に識別するための識別子（例えばセルの識別番号）と、を有することを特徴とするコンピュータ・システムも提供される。

【００１４】

加えて、本発明によれば、コンピュータ・ネットワークを介する伝送用ソフトウェア・セルのデータ・ストリームにおいて、前記コンピュータ・ネットワークが、複数のプロセッサを有し、前記ソフトウェア・セルの各々が、前記プロセッサの中の１以上によって処理するためのプログラムと、前記プログラムと関連付けられたデータと、前記ネットワークを介して伝送されるすべてのソフトウェア・セルの中で前記ソフトウェア・セルを一意的に識別するグローバルな識別子と、を有することを特徴とするデータ・ストリームも提供される。なお、上記構成において、「データ・ストリーム」という形態に代えて、「データ構造」という形態、あるいは「上述のような構造を有するデータ」という形態で本発明を提供することも可能である。

【００１５】

他の実施形態では、本発明は、ネットワークを介してデータとアプリケーションを伝送するための、また、ネットワークのメンバー間でデータとアプリケーションを処理するための新しいプログラミング・モデルを提供する。このプログラミング・モデルでは、ネットワークのいずれのメンバーでも処理できる、ネットワークを介して伝送されるソフトウェア・セルが使用される。各ソフトウェア・セルは同じ構造を有し、アプリケーションとデータの双方を含むことが可能である。モジュラー型コンピュータ・アーキテクチャによって提供される高速処理と伝送速度の結果、これらのセルの高速処理が可能となる。アプリケーション用コードは同じ共通の命令セットとＩＳＡに好適に基づくものである。各ソフトウェア・セルは、グローバルな識別子(グローバルＩＤ)と、セルの処理に必要な計算用リソースの量について説明する情報とを好適に含むことが望ましい。すべての計算用リソースは同じ基本構造を有し、同じＩＳＡが用いられているので、このセルの処理を実行する特定のリソースは、ネットワーク上のどこにでも配置が可能となり、動的に割り当てることができる。

【００１６】

基本となる処理用モジュールはプロセッサ・エレメント(ＰＥ)である。ＰＥは、好適には、処理ユニット(ＰＵ)、ダイレクト・メモリ・アクセス・コントローラ(ＤＭＡＣ)および複数の付加処理ユニット(ＡＰＵ)を具備することが望ましい。好ましい実施形態では、１つのＰＥは８つのＡＰＵを具備する。ＰＵとＡＰＵとは、クロスバ・アーキテクチャを好適に備えている共用ダイナミック・ランダム・アクセス・メモリ(ＤＲＡＭ)を用いてリアルタイムで通信を行う。ＰＵは、ＡＰＵによって、データとアプリケーションの処理のスケジュール管理と全般的管理とを行う。ＡＰＵは並列的かつ独立にこの処理を実行する。ＤＭＡＣは、共用ＤＲＡＭに格納されているデータとアプリケーションへのアクセス制御をＰＵとＡＰＵとによって行う。

【００１７】

このモジュラー構造によれば、ネットワークのメンバーによって用いられるＰＥの数は、そのメンバーが必要とする処理パワーに基づく。例えば、１台のサーバーは４つのＰＥを用いることができ、１台のワークステーションは２つのＰＥを用いることができ、１つのＰＤＡは１つのＰＥを用いることができる。特定のソフトウェア・セルの処理に割り当てられるＰＥのＡＰＵの数は、そのセル内のプログラムとデータの複雑さと大きさとによって決まる。

【００１８】

好ましい実施形態では、複数のＰＥが１つの共用ＤＲＡＭと関連付けられる。好適には、ＤＲＡＭは複数のセクションに分割され、これらのセクションの各々は複数のメモリ・バ

ンクに分割される。特に好ましい実施形態では、ＤＲＡＭは６４個のメモリ・バンクを有し、各バンクは１メガバイトの記憶容量を有する。ＤＲＡＭの各セクションは、好適には、バンク・コントローラによって制御されることが望ましく、ＰＥの各ＤＭＡＣは、好適には、各バンク・コントローラにアクセスすることが望ましい。したがって、この実施形態の各ＰＥのＤＭＡＣは、共用ＤＲＡＭの任意の部分へのアクセスが可能となる。

【００１９】

別の態様では、本発明は、共用ＤＲＡＭからのＡＰＵのデータの読み出しと、共用ＤＲＡＭへのデータの書き込みのための同期システムと方法とを提供する。このシステムによって、ＤＲＡＭを共用している複数のＡＰＵと複数のＰＥとの間のコンフリクトが防止される。このシステムと方法とによれば、ＤＲＡＭ領域が指定され、複数のフルーエンプティ・ビットが格納される。これらのフルーエンプティ・ビットの各々は、ＤＲＡＭの指定領域に対応する。この同期システムはＤＲＡＭのハードウェアの中に統合化されるので、ソフトウェアの中で実行されるデータ同期方式の計算上のオーバーヘッドはこのシステムによって防止される。

【００２０】

また本発明によって、ＤＲＡＭ内にサンドボックスが設けられ、１つのＡＰＵのプログラム処理用データから生じる、別のＡＰＵのプログラム処理用データの破損に対するセキュリティが与えられる。各サンドボックスによって、データの読み出しや書き込みが不可能となる共用ＤＲＡＭ領域が画定される。

【００２１】

別の態様では、本発明は、ＰＵがＡＰＵへコマンドを出して、アプリケーションとデータのＡＰＵによる処理を開始するためのシステムと方法とを提供するものである。これらのコマンドは、ＡＰＵ遠隔処理命令(ＡＲＰＣ)と呼ばれ、このコマンドによって、ＡＰＵがコプロセッサの役割を演じることなく、アプリケーションとデータのＡＰＵによる並列処理のＰＵによる全般的管理と調整が可能となる。

【００２２】

他の実施形態では、本発明によって、ストリーミング・データ処理用の専用パイプライン構造を設定するシステムと方法とが提供される。このシステムと方法によれば、ＰＵによってこれらのストリーミング・データの処理を行うために、ＡＰＵの調整グループと、これらのＡＰＵと関連するメモリサンドボックスの調整グループとが設定される。パイプ・ラインの専用ＡＰＵとメモリサンドボックスとは、データ処理が行われない時間中もパイプ・ライン専用のままである。言い換えれば、専用ＡＰＵ及びこれらの専用ＡＰＵと関連するサンドボックスとは、この期間中は予約状態となる。

【００２３】

他の実施形態では、本発明はタスク処理用の絶対タイマーを提供する。この絶対タイマーは、アプリケーションとデータの処理用としてＡＰＵが使用するクロック周波数に依存しない。アプリケーションは、絶対タイマーによって定義される、タスク用の時間に基づいて書かれる。ＡＰＵが使用しているクロック周波数が、ＡＰＵの機能の改善などに起因して増加しても、絶対タイマーによって定義される所定のタスク用の時間はそのまま同じである。この方式によれば、古いＡＰＵにおける遅い処理時間を前提として書かれた古いアプリケーションの処理を、これらの新しいＡＰＵでは行わせないこととする必要がなく、かつ、新しいバージョンのＡＰＵによる処理時間の向上を実現することが可能になる。

【００２４】

また本発明は、より処理速度が高速な新しいＡＰＵを、古いＡＰＵにおける遅い処理速度を前提として書かれた古いアプリケーションの処理に用いることを可能にする、他の方式をも提供するものである。この方式では、速度の改善によって生じるＡＰＵの並列処理の調整における問題の処理の間に、これらの古いアプリケーションの処理時にＡＰＵが使用している命令(マイクロコード)が分析される。ＡＰＵによる処理の順番がプログラムが予期する順番どおりに維持されるよう、“オペレーションなし”(“ＮＯＯＰ”)命令が、これらのＡＰＵのいくつかによって実行される命令の中へ挿入される。これらの命令の中へ

これらの“ＮＯＯＰ”を挿入することにより、ＡＰＵによるすべての命令を実行するための正しいタイミングが維持される。

【００２５】

他の実施形態では、本発明は、光導波路が統合化される集積回路を含むチップ・パッケージを提供するものである。

また、本発明によれば、プログラムと前記プログラムと関連付けられたデータとを格納する第１メモリと、前記プログラムとそれに関連付けられた前記データとを処理する複数の第１処理ユニットと、前記第１処理ユニットによる前記第１メモリへのアクセスを制御するためのメモリ・コントローラと、アクセス・テーブルとキー・テーブルを格納する第２メモリであって、前記アクセス・テーブルは、複数のアクセス・エントリを有し、各アクセス・エントリは、アクセス・キーと、前記アクセス・キーと関連付けられた前記第１メモリ内の１つのメモリ・スペースの識別子を含み、前記キー・テーブルは、複数のキー・エントリを有し、各キー・エントリは、前記第１処理ユニットの識別子と前記第１処理ユニットと関連するリクエスト・キーを含んで成る第２メモリと、前記第１処理ユニットによる前記プログラムとそれに関連付けられた前記データの処理を制御するための第２処理ユニットであって、前記アクセス・テーブルと前記キー・テーブルの作成と更新の操作が可能で、更に、前記第１処理ユニットによる前記プログラムの１つの処理を行う指示を出すことが可能となる前記第２処理ユニットとを有し、前記第１処理ユニットの１つは、前記１つのプログラムの処理において前記メモリ・コントローラにリクエストを出して、前記第１メモリ内の格納位置をアクセスすることが可能であり、前記メモリ・コントローラは、前記リクエストに応答して、前記キー・テーブル内の前記第１処理ユニットの１つと関連付けられたリクエスト・キーと、前記アクセス・テーブル内のアクセス・キーとを比較して、前記アクセス・キーの１つが、前記キー・テーブル内の前記第１処理ユニットの１つと関連付けられたリクエスト・キーと一致して、かつ、前記格納位置が、前記アクセス・テーブル内の前記１つのアクセス・キーと関連付けられたメモリ・スペースと一致する場合は、前記リクエストの実行が可能であることを特徴とするコンピュータ処理システムも提供される。

また、コンピュータで実行される処理方法において、第１メモリに、プログラムと、前記プログラムと関連付けられたデータと、を格納するステップと、

複数の第１処理ユニットを用いて、前記プログラムとそれに関連付けられた前記データとを処理するステップと、メモリ・コントローラを用いて、前記第１処理ユニットによる前記第１メモリへのアクセスを制御するステップと、第２処理ユニットを用いて、第２メモリに、アクセス・テーブルとキー・テーブルを作成するステップであって、前記アクセス・テーブルは、複数のアクセス・エントリを有し、各アクセス・エントリが、前記アクセス・キーと、前記アクセス・キーと関連付けられた前記第１メモリのメモリ・スペースの識別子を含み、前記キー・テーブルが、複数のキー・エントリを有し、各キー・エントリが、１つの前記第１処理ユニットの識別子と、前記第１処理ユニットと関連付けられたリクエスト・キーを含むように構成するステップと、前記第２処理ユニットを用いて、前記第１処理ユニットによる前記プログラムとそれに関連付けられた前記データの前記処理を制御するステップと、前記第２処理ユニットを用いて、前記第１処理ユニットのうちの１つに、前記プログラムのうちの１つの処理を命令するステップと、前記１つの第１処理ユニットから、前記１つのプログラムの処理において、前記メモリ・コントローラにリクエストを出して、前記第１メモリ内の格納位置へアクセスするステップと、前記リクエストに応答して、前記１つの第１処理ユニットと関連付けられたリクエスト・キーと、前記アクセス・テーブル内の前記アクセス・キーとを比較するステップと、１つの前記アクセス・キーが、前記キー・テーブル内の前記１つの処理ユニットと関連付けられたリクエスト・キーと一致し、前記格納位置が、前記アクセス・テーブル内で指定された前記１つのアクセス・キーと関連付けられたメモリ・スペースと一致する場合は、前記リクエストを実行するステップと、を有することを特徴とする方法も提供される。

また、プログラムと前記プログラムと関連付けられたデータを格納する第１メモリであっ

て、複数のアドレス可能な格納位置を有し、アドレス可能な各格納位置は、前記アドレス可能な格納位置に関連付けられるとともに前記アドレス可能な格納位置のアクセス・キーを含む追加メモリ・セグメント有して成る第１メモリと、前記プログラムとそれに関連付けられた前記データとを処理するための複数の第１処理ユニットと、前記第１メモリへの前記第１処理ユニットによるアクセス制御のためのメモリ・コントローラと、複数のキー・エントリから成るキー・テーブルを格納するための第２メモリであって、各キー・エントリが、前記第１処理ユニットの１つの識別子と、前記第１処理ユニットと関連付けられたリクエスト・キーとを含んでなる第２メモリと、前記第１処理ユニットによる前記プログラムとそれに関連付けられた前記データの処理を制御するための第２処理ユニットであって、前記アクセス・キーの割り当てと更新および前記アクセス・キー・テーブルの作成と更新が可能で、更に、前記第１処理ユニットのうちの１つによる前記プログラムの１つの処理を指示することが可能となる第２処理ユニットとを有し、前記第１処理ユニットの１つは、前記プログラムの１つの処理において前記メモリ・コントローラにリクエストを出して、前記第１メモリ内の格納位置をアクセスすることが可能であり、前記メモリ・コントローラは、前記リクエストに応答して、前記キー・テーブル内の前記第１処理ユニットの１つと関連付けられたリクエスト・キーと、前記１つのアドレス可能な格納位置と関連付けられている追加メモリ・セグメントに含まれているアクセス・キーとを比較して、前記キー・テーブル内の前記第１処理ユニットの１つと関連付けられたリクエスト・キーと、前記１つのアドレス可能な格納位置と関連付けられている追加メモリ・セグメントに含まれている前記アクセス・キーが一致する場合は、前記リクエストを実行することが可能であることを特徴とするコンピュータ処理システムも提供される。

更に、コンピュータ処理システムにてなされる処理方法において、第１メモリに、プログラムと前記プログラムの関連データを格納するステップであって、複数のアドレス可能な格納位置を有し、前記アドレス可能な各格納位置には、前記アドレス可能な格納位置に関連付けられた追加メモリ・セグメントを含むステップと、各々の前記アドレス可能な格納位置用の各々の前記追加メモリ・セグメントに、前記アドレス可能な格納位置のためのアクセス・キーを格納するステップと、複数の第１処理ユニットを用いて、前記プログラムとそれに関連付けられた前記データを処理するステップと、メモリ・コントローラを用いて、前記第１処理ユニットによる前記第１メモリへのアクセスを制御するステップと、第２メモリに、複数のキー・エントリから成るキー・テーブルを格納するステップであって、各キー・エントリが、前記第１処理ユニットと関連付けられた識別子と、前記第１処理ユニットと関連付けられたリクエスト・キーを含むステップと、前記第２処理ユニットを用いて、前記第１処理ユニットによる前記プログラムとそれに関連付けられた前記データの前記処理を制御するステップと、前記第２処理ユニットを用いて、各々の前記アクセス・キーを割り当てるステップと、前記第２処理ユニットを用いて、前記キー・テーブルを作成するステップと、前記第２処理ユニットを用いて、前記第１処理ユニットのいずれか１つに、前記プログラムの１つを実行する命令を出すステップと、前記第１処理ユニットの１つから、前記１つのプログラムの処理において、１つの前記アドレス可能な格納位置へアクセスするように、前記メモリ・コントローラへリクエストを出すステップと、

前記リクエストに応答して、前記キー・テーブル内で前記１つの第１処理ユニットと関連付けられたリクエスト・キーと、前記１つのアドレス可能な格納位置に格納されているアクセス・キーとを比較するステップと、前記キー・テーブル内の前記１つの処理ユニットと関連付けられたリクエスト・キーと前記アドレス可能な格納位置と関連付けられた前記追加メモリ・セグメント内に格納されている前記アクセス・キーが一致する場合は、前記リクエストを実行するステップと、を有することを特徴とする方法も提供される。

【００２６】

【発明の実施の形態】

図１に、本発明によるコンピュータ・システム１０１のアーキテクチャ全体を示す。

【００２７】

この図に例示されているように、システム１０１にはネットワーク１０４が含まれ、複数

のコンピュータとコンピューティング・デバイスがこのネットワークと接続されている。ネットワーク１０４の例として、ＬＡＮ、インターネットのようなグローバルネットワーク、あるいは他のコンピュータ・ネットワークが挙げられる。

【００２８】

ネットワーク１０４と接続されたコンピュータとコンピューティング・デバイス(ネットワークの“メンバー”)の中には、クライアント側コンピュータ１０６、サーバーコンピュータ１０８、個人用情報機器(ＰＤＡ)１１０、デジタルテレビ(ＤＴＶ)１１２およびその他の有線または無線コンピュータとコンピューティング・デバイスなどが含まれる。ネットワーク１０４のメンバーによって用いられるプロセッサは、同じ共通のコンピューティング・モジュールから構成される。またこれらのプロセッサは、好適には、ＩＳＡがすべて同じで、好適には同じ命令セットに従って処理を実行する。個々のプロセッサ内に含まれるモジュールの数は、そのプロセッサが必要とする処理パワーによって決められる。

【００２９】

例えば、システム１０１のサーバー１０８は、クライアント１０６より多いデータ処理およびアプリケーション処理を実行するので、クライアント１０６より多いコンピューティング・モジュールを含むことになる。一方、ＰＤＡ１１０では最低量の処理しか実行されない。したがって、ＰＤＡ１１０には最少の数のコンピューティング・モジュールしか含まれない。ＤＴＶ１１２はクライアント１０６とサーバー１０８の間の処理レベルを実行する。したがって、ＤＴＶ１１２にはクライアント１０６とサーバー１０８の間のいくつかのコンピューティング・モジュールが含まれる。以下に説明するように、各コンピューティング・モジュールの中には、処理用コントローラと、ネットワーク１０４を介して伝送されるデータおよびアプリケーションの並列処理を実行する複数の同一処理ユニットとが含まれる。

【００３０】

システム１０１がこのように均質な構成を有することから、アダプタビリティ、処理速度および処理効率が改善される。システム１０１の各メンバーが、同じコンピューティング・モジュールのうち１つまたはそれ以上(またはコンピューティング・モジュールの一部)を用いて処理を実行するので、データとアプリケーションの実際の処理をどのコンピュータまたはコンピューティング・デバイスで実行するかは重要ではなくなる。さらに、個々のアプリケーションとデータの処理は、ネットワークのメンバーの間で分担することができる。システム全体を通じて、システム１０１が処理したデータとアプリケーションを含むセルを一意的に識別することにより、この処理がどこで行われたかにかかわらず、処理を要求したコンピュータまたはコンピューティング・デバイスへその処理結果を伝送することが可能となる。この処理を実行するモジュールが共通の構造と共通のＩＳＡとを有するので、プロセッサ間の互換性を達成するためのソフトウェアの追加層の計算上の負担が回避される。このアーキテクチャとプログラミング・モデルによって、リアルタイムのマルチメディア・アプリケーションなどの実行に必要な処理速度が改善される。

【００３１】

システム１０１によって改善される処理速度と効率というさらなる利点を利用するために、このシステムによって処理されるデータとアプリケーションとは、一意的に識別される、それぞれフォーマットが同じであるソフトウェア・セル１０２へとパッケージ化される。各ソフトウェア・セル１０２は、アプリケーションとデータの双方を含むあるいは含み得る。また各ソフトウェア・セルには、ネットワーク１０４とシステム１０１全体の中でセルを識別するためのセル識別子が含まれ、その一例としては、ソフトウェア・セルをグローバルに識別するＩＤが含まれる。ソフトウェア・セルのこの構造的均一性と、ネットワークの中でのソフトウェア・セルの一意的識別とによって、ネットワークの任意のコンピュータまたはコンピューティング・デバイスでのアプリケーションとデータの処理が改善される。例えば、クライアント１０６は、ソフトウェア・セル１０２の作成を行うこともできるが、クライアント１０６側の処理能力は限られていることから、このソフトウェア・セルをサーバー１０８へ伝送して処理してもらうこともできる。したがって、ソフト

ウェア・セルは、ネットワーク１０４全体を移動してネットワーク上での処理用リソースの可用性に基づく処理を行うことが可能となる。

【００３２】

また、システム１０１のプロセッサとソフトウェア・セルが均質な構造を有することで、今日の異質なネットワークの混在という問題の多くを防ぐことができる。例えば任意の命令セットを用いる任意のどのＩＳＡ上でもアプリケーションの処理を許容しようとする非効率的なプログラミング・モデル(Ｊａｖａのバーチャル・マシーンのような仮想マシーンなど)が回避される。したがって、システム１０１は、今日のネットワークよりもはるかに効率的、かつ、はるかに効果的に広帯域処理の実現が可能となる。

【００３３】

ネットワーク１０４のすべてのメンバーのための基本となる処理用モジュールはプロセッサ・エレメント(ＰＥ)である。図２にＰＥの構造が例示されている。この図に示すように、ＰＥ２０１は、処理ユニット(ＰＵ)２０３、ＤＭＡＣ２０５、複数の付加処理ユニット(ＡＰＵ)、すなわち、ＡＰＵ２０７、ＡＰＵ２０９、ＡＰＵ２１１、ＡＰＵ２１３、ＡＰＵ２１５、ＡＰＵ２１７、ＡＰＵ２１９、ＡＰＵ２２１を具備する。ローカルＰＥバス２２３は、ＡＰＵと、ＤＭＡＣ２０５と、ＰＵ２０３との間でデータとアプリケーションとを伝送する。ローカルＰＥバス２２３は、従来型のアーキテクチャなどを備えていてもよいし、あるいは、パケット交換式ネットワークとして実現されてもよい。パケット交換式ネットワークとして実現される場合、より多くのハードウェアが必要となり、その一方で、利用可能な帯域幅が増加する。

【００３４】

ＰＥ２０１は、デジタル論理回路を実現する様々な方法を用いて構成可能である。しかし、ＰＥ２０１は、好適には、シリコン基板上の単一の集積回路として構成されることが望ましい。基板用代替材料の中には、ガリウム砒素、ガリウム・アルミニウム砒素、砒素および多種多様のドーパントを用いるその他のいわゆるIII－Ｂ化合物が含まれる。またＰＥ２０１は、超伝導材料(高速単一磁束量子(ＲＳＦＱ)論理処理など)を用いて実現することもできる。

【００３５】

ＰＥ２０１は、高帯域メモリ接続部２２７を介してダイナミック・ランダム・アクセス・メモリ(ＤＲＡＭ)２２５と密接に関連する。ＤＲＡＭ２２５はＰＥ２０１用メイン・メモリとして機能する。ＤＲＡＭ２２５は好適には、ダイナミック・ランダム・アクセス・メモリであることが望ましいとはいえ、他の手段、例えばスタティック・ランダム・アクセス・メモリ(ＳＲＡＭ)として、磁気ランダム・アクセス・メモリ(ＭＲＡＭ)、光メモリまたはホログラフィ・メモリなどを用いてＤＲＡＭ２２５を実現することもできる。ＤＭＡＣ２０５によって、ＤＲＡＭ２２５と、ＰＥ２０１のＡＰＵとＰＵとの間のデータ転送が改善される。以下さらに説明するように、ＤＭＡＣ２０５によって、各ＡＰＵに対するＤＲＡＭ２２５内の排他的領域が指定されるが、この排他的領域の中へはＡＰＵだけしかデータの書き込みができず、また、ＡＰＵだけしかこの排他的領域からのデータ読み出しを行うことができない。この排他的領域は“サンドボックス”と呼ばれる。

【００３６】

ＰＵ２０３は、データとアプリケーションのスタンド・アローン型処理が可能な標準的プロセッサなどであってもよい。作動時に、ＰＵは、ＡＰＵによって、データとアプリケーションの処理のスケジュール管理と全般的管理とを行う。ＡＰＵは好適には、単一命令、複数データ(ＳＩＭＤ)プロセッサであることが望ましい。ＰＵ２０３の制御によって、ＡＰＵは、並列的かつ独立にこれらのデータとアプリケーションの処理を実行する。ＤＭＡＣ２０５は、共用ＤＲＡＭ２２５に格納されているデータとアプリケーションへのＰＵ２０３とＡＰＵによるアクセス制御を行う。ＰＥ２０１は、好適には８個のＡＰＵを含むことが望ましいとはいえ、必要とする処理パワーに応じて、ＰＥ内でこの数より多少上下する個数のＡＰＵを用いてもよい。また、ＰＥ２０１のようないくつかのＰＥを結合(まとめてパッケージ化)して処理パワーの改善を図ることもできる。

【００３７】
例えば、図３に示すように、１以上のチップ・パッケージなどの中に４つのＰＥをパッケージ化(まとめて結合)してネットワーク１０４のメンバー用の単一プロセッサを形成してもよい。この構成は広帯域エンジン(ＢＥ)と呼ばれる。図３に示すように、ＢＥ３０１には４つのＰＥ(ＰＥ３０３、ＰＥ３０５、ＰＥ３０７、ＰＥ３０９)が含まれる。これらのＰＥ間の通信はＢＥバス３１１を介して行われる。広帯域メモリ接続部３１３によって共用ＤＲＡＭ３１５とこれらのＰＥ間の通信が行われる。ＢＥバス３１１の代わりに、ＢＥ３０１のＰＥ間の通信は、ＤＲＡＭ３１５とこのメモリ接続部とを介して行うことができる。

【００３８】
入力／出力(Ｉ／Ｏ)インターフェース３１７と外部バス３１９とは、広帯域エンジン３０１とネットワーク１０４のその他のメンバー間で通信を行う。ＢＥ３０１の各ＰＥは、ＰＥのＡＰＵによって行われるアプリケーションとデータの並列的かつ独立した処理と同様の並列的かつ独立した方法で、データとアプリケーションの処理を実行する。

【００３９】
図４はＡＰＵの構造を例示する図である。ＡＰＵ４０２には、ローカル・メモリ４０６、レジスタ４１０、４つの浮動小数点演算ユニット４１２および４つの整数演算ユニット４１４が含まれる。しかし、ここでもまた、必要とする処理パワーに応じて、４個より多少上下する個数の浮動小数点演算ユニット４１２と整数演算ユニット４１４を用いてもよい。１つの好ましい実施形態では、ローカル・メモリ４０６には１２８キロバイトの記憶容量が含まれ、レジスタ４１０の容量は１２８×１２８ビットである。浮動小数点演算ユニット４１２は、毎秒３２０億浮動小数点演算(３２ＧＬＯＰＳ)の速度で好適に作動し、整数演算ユニット４１４は、毎秒３２０億回の演算速度(３２ＧＯＰ)で好適に作動する。

【００４０】
ローカル・メモリ４０６はキャッシュ・メモリではない。ローカル・メモリ４０６は、好適にはＳＲＡＭとして構成されることが望ましい。ＡＰＵに対するキャッシュ・コヒーレンシー、つまりキャッシュの整合性のサポートは不要である。ＰＵでは、当該ＰＵで開始されるダイレクト・メモリー・アクセス(ＤＭＡ)をサポートするためにキャッシュの整合性が要求される場合もある。しかし、ＡＰＵによって開始されるＤＭＡに対する、あるいは、外部装置からのおよび外部装置へのアクセスに対するキャッシュの整合性のサポートは不要である。

【００４１】
ＡＰＵ４０２にはさらに、ＡＰＵへおよびＡＰＵからアプリケーションとデータとを伝送するためのバス４０４が含まれる。１つの好ましい実施形態ではこのバスは１０２４ビットの幅を持つ。ＡＰＵ４０２にはさらに内部バス４０８、４２０、４１８が含まれる。１つの好ましい実施形態では、バス４０８は２５６ビットの幅を持ち、ローカル・メモリ４０６とレジスタ４１０間で通信を行う。バス４２０と４１８とは、それぞれ、レジスタ４１０と浮動小数点演算ユニット４１２との間、および、レジスタ４１０と整数演算ユニット４１４間で通信を行う。ある好ましい実施形態では、レジスタ４１０から浮動小数点演算ユニット４１２または整数演算ユニット４１４へのバス４１８と４２０の幅は、３８４ビットであり、浮動小数点演算ユニット４１２または整数演算ユニット４１４からレジスタ４１０へのバス４１８と４２０の幅は１２８ビットである。浮動小数点演算ユニット４１２または整数演算ユニット４１４からレジスタ４１０への幅より広い、レジスタ４１０から浮動小数点演算ユニットまたは整数演算ユニットへの上記バスの広い幅によって、レジスタ４１０からのより広いデータ・フローが処理中に許容される。最大３ワードが各計算には必要となる。しかし、各計算の結果は、一般に、ただ１ワードだけである。

【００４２】
図５～１０は、ネットワーク１０４のメンバーのプロセッサのモジュラー構造をさらに例示する図である。例えば、図５に示すように、１つのプロセッサには単一のＰＥ５０２を含むことができる。上述のように、このＰＥには、一般に、ＰＵ、ＤＭＡＣおよび８個の

ＡＰＵが含まれる。各ＡＰＵにはローカル・ストレージ(ＬＳ)が含まれる。一方、プロセッサは、ビジュアライザ(ＶＳ)５０５の構造を有する場合もある。図５に示すように、ＶＳ５０５はＰＵ５１２、ＤＭＡＣ５１４および４つのＡＰＵ(ＡＰＵ５１６、ＡＰＵ５１８、ＡＰＵ５２０、ＡＰＵ５２２)を有する。ＰＥのその他の４つのＡＰＵによって通常占有されるチップ・パッケージ内のスペースは、この場合、ピクセル・エンジン５０８、画像用キャッシュ５１０およびブラウン管コントローラ(ＣＲＴＣ)５０４によって占有される。ＰＥ５０２またはＶＳ５０５に求められる通信速度に応じて、チップ・パッケージの中に光インターフェース５０６が含まれる場合もある。

【００４３】

この標準化されたモジュラー構造を用いて、多数の他のプロセッサの変更例を容易にかつ効率的に構成することが可能となる。例えば、図６に示すプロセッサは、２つのチップ・パッケージ(ＢＥを備えるチップ・パッケージ６０２と、４つのＶＳを含むチップ・パッケージ６０４)を有する。入出力部(Ｉ／Ｏ)６０６によって、チップ・パッケージ６０２のＢＥとネットワーク１０４との間にインターフェースが設けられる。バス６０８はチップ・パッケージ６０２とチップ・パッケージ６０４との間の通信を行う。入出用プロセッサ(ＩＯＰ)６１０によってデータ・フローが制御され、Ｉ／Ｏ６０６へのまたはＩ／Ｏ６０６からの入出力が行われる。Ｉ／Ｏ６０６はＡＳＩＣ（Application Specific Integrated Circit)として製造が可能である。ＶＳからの出力はビデオ信号６１２である。

【００４４】

図７は、ネットワーク１０４のその他のメンバーへ超高速通信を行う２つの光インターフェース７０４と７０６とを備えたＢＥ７０２用のチップ・パッケージ(またはローカルに接続された他のチップ・パッケージ)を例示する。ＢＥ７０２は、ネットワーク１０４上でサーバーなどとして機能することができる。

【００４５】

図８のチップ・パッケージは、２つのＰＥ８０２及び８０４および２つのＶＳ８０６及び８０８を有する。Ｉ／Ｏ８１０は、チップ・パッケージとネットワーク１０４との間にインターフェースを与える。チップ・パッケージからの出力はビデオ信号である。この構成は画像処理用ワークステーションなどとして機能することができる。

【００４６】

図９はさらに別の構成を例示する。この構成は、図８に例示されている構成の処理パワーの１／２を含む。２つのＰＥの代わりに１つのＰＥ９０２が設けられ、２つのＶＳの代わりに１つのＶＳ９０４が設けられる。Ｉ／Ｏ９０６は、図８に例示されているＩ／Ｏの帯域幅の１／２の帯域幅を有する。またこのようなプロセッサは、画像処理用ワークステーションとして機能することができる。

【００４７】

最後の構成が図１０に図示されている。このプロセッサは、単一のＶＳ１００２とＩ／Ｏ１００４だけから構成される。この構成はＰＤＡなどとして機能することができる。

【００４８】

図１１は、ネットワーク１０４のプロセッサのチップ・パッケージの中への光インターフェースの統合を例示する図である。これらの光インターフェースによって、光信号は電気信号に変換され、電気信号は光信号に変換される。また、これらの光インターフェースは、ガリウム砒素、アルミニウム・ガリウム砒素、ゲルマニウムその他の元素や化合物などを含む様々な材料から構成することができる。この図に示すように、光インターフェース１１０４と１１０６とはＢＥ１１０２のチップ・パッケージの上に組み立てられる。ＢＥバス１１０８はＢＥ１１０２のＰＥ、すなわち、ＰＥ１１１０、ＰＥ１１１２、ＰＥ１１１４、ＰＥ１１１６およびこれらの光インターフェースとの間での通信を行う。光インターフェース１１０４には２つのポート(ポート１１１８とポート１１２０)が含まれ、また光インターフェース１１０６には２つのポート(ポート１１２２とポート１１２４)が含まれる。ポート１１１８、１１２０、１１２２、１１２４は、光導波路１１２６、１１２８、１１３０、１１３２とそれぞれ接続される。光信号は、光インターフェース１１０４と

１１０６のポートを介して、これらの光導波路の中を通り、ＢＥ１１０２へおよびＢＥ１１０２から伝送される。

【００４９】

このような光導波路と各ＢＥの４つの光ポートとを用いて様々な構成において複数のＢＥをまとめて接続してもよい。例えば、図１２に示すように、このような光ポートを介して２つまたはそれ以上のＢＥ(ＢＥ１１５２、ＢＥ１１５４、ＢＥ１１５６など)を直列に接続することができる。この例では、ＢＥ１１５２の光インターフェース１１６６は、その光ポートを介しＢＥ１１５４の光インターフェース１１６０の光ポートと接続される。同様に、ＢＥ１１５４の光インターフェース１１６２の光ポートは、ＢＥ１１５６の光インターフェース１１６４の光ポートと接続される。

【００５０】

図１３にマトリクス構成が例示される。この構成では、各ＢＥの光インターフェースは２つの他のＢＥと接続される。この図に示すように、ＢＥ１１７２の光インターフェース１１８８の光ポートの中の１つが、ＢＥ１１７６の光インターフェース１１８２の光ポートと接続される。光インターフェース１１８８のもう一方の光ポートは、ＢＥ１１７８の光インターフェース１１８４の光ポートと接続される。同様に、ＢＥ１１７４の光インターフェース１１９０の１つの光ポートはＢＥ１１７８の光インターフェース１１８４のもう一方の光ポートと接続される。光インターフェース１１９０のもう一方の光ポートは、ＢＥ１１８０の光インターフェース１１８６の光ポートと接続される。このマトリックス構成は他のＢＥに対しても同様に拡張することができる。

【００５１】

シリアル構成かマトリックス構成のいずれかを用いて、任意の所望のサイズとパワーから成るネットワーク１０４用プロセッサの構成が可能となる。言うまでもなく、ＢＥの光インターフェースに対して、または、ＢＥよりＰＥ数の少ないプロセッサに対して追加ポートを加えて、他の構成を形成してもよい。

【００５２】

図１４はＢＥのＤＲＡＭに対する制御システムと構造を例示する図である。同様の制御システムと構造が、別のサイズを持ち、多少異なる数のＰＥを含むプロセッサの中で用いられる。この図に示すように、クロスバ交換機によって、ＢＥ１２０１を備える４つのＰＥからなる各ＤＭＡＣ１２１０が８つのバンク・コントロール１２０６と接続される。各バンク・コントロール１２０６によって、ＤＲＡＭ１２０４の８つのバンク１２０８(４つだけしか図示されていない)が制御される。したがって、ＤＲＡＭ１２０４は、合計６４のバンクを具備することになる。好ましい実施形態では、ＤＲＡＭ１２０４は６４メガバイトの容量を持ち、各バンクは１メガバイトの容量を持っている。各バンク内の最小のアドレス指定可能単位は、この好ましい実施形態では１０２４ビットのブロックである。

【００５３】

ＢＥ１２０１にはスイッチ・ユニット１２１２も含まれる。スイッチ・ユニット１２１２によって、ＢＥ１２０１と密接に接続されているＢＥの他のＡＰＵのＤＲＡＭ１２０４へのアクセスが可能となる。したがって、第２のＢＥを第１のＢＥと密接に接続することが可能となり、さらに、各ＢＥの各ＡＰＵは、ＡＰＵが通常アクセス可能なメモリ・ロケーションの数の２倍のアドレス指定を行うことが可能となる。スイッチ・ユニット１２１２のようなスイッチ・ユニットを介して、第１のＢＥのＤＲＡＭから第２のＢＥのＤＲＡＭへのデータの直接読み出し、または、第２のＢＥのＤＲＡＭから第１のＢＥのＤＲＡＭへのデータの直接書き込みを行うことが可能となる。

【００５４】

例えば、図１５に示すように、このような書き込みを行うために、第１のＢＥのＡＰＵ(ＢＥ１２２２のＡＰＵ１２２０など)によって、第２のＢＥのＤＲＡＭ(通常の場合のようなＢＥ１２２２のＤＲＡＭ１２２４ではなく、ＢＥ１２２６のＤＲＡＭ１２２８など)のメモリ・ロケーションへの書き込みコマンドが出される。ＢＥ１２２２のＤＭＡＣ１２３０は、クロスバ交換機１２２１を介して、バンク・コントロール１２３４へ書き込みコマ

ンドを送り、バンク・コントロール１２３４は、バンク・コントロール１２３４と接続された外部ポート１２３２へコマンドを伝送する。ＢＥ１２２６のＤＭＡＣ１２３８は書き込みコマンドを受け取り、ＢＥ１２２６のスイッチ・ユニット１２４０へこのコマンドを転送する。スイッチ・ユニット１２４０は書き込みコマンドの中に含まれるＤＲＡＭアドレスを識別し、ＢＥ１２２６のバンク・コントロール１２４２を介して、ＤＲＡＭ１２２８のバンク１２４４へ、ＤＲＡＭアドレス内に格納するデータを送る。したがって、スイッチ・ユニット１２４０によって、ＤＲＡＭ１２２４とＤＲＡＭ１２２８の双方は、ＢＥ１２２２のＡＰＵ用の単一メモリ空間として機能することが可能になる。

【００５５】

図１６はＤＲＡＭの６４個のバンク構成を図示する。これらのバンクは、８つの行(１３０２、１３０４、１３０６、１３０８、１３１０、１３１２、１３１４、１３１６)と８つの列(１３２０、１３２２、１３２４、１３２６、１３２８、１３３０、１３３２、１３３４)とで構成されている。各行は１つのバンク・コントローラによって制御される。したがって、各バンク・コントローラは８メガバイトのメモリを制御する。

【００５６】

図１７と１８は、最小のアドレス指定可能な格納単位(１０２４ビットのブロックなど)でのＤＲＡＭの格納とアクセスを行うための異なる構成を例示する。図１７で、ＤＭＡＣ１４０２は単一のバンク１４０４の中に８つの１０２４ビット・ブロック１４０６を格納する。図１８では、ＤＭＡＣ１４１２によって、１０２４ビットを含むデータ・ブロックの読み出しと書き込みが行われるものの、これらのブロックは、２つのバンク(バンク１４１４とバンク１４１６)の間で分配される。したがって、これらのバンクの各々には１６個のデータ・ブロックが含まれ、データの各ブロックには５１２ビットが含まれる。この分配によって、ＤＲＡＭのアクセスをさらに高速なものに改善することが可能となり、ある種のアプリケーションの処理に役立つ。

【００５７】

図１９はＰＥ内のＤＭＡＣ１５０６のアーキテクチャを例示する。この図に例示されているように、各ＡＰＵ１５０２がＤＭＡＣ１５０６の構造上のノード１５０４へ直接アクセスを行うように、ＤＭＡＣ１５０６を含む構造上のハードウェアは全てのＰＥを通じて配設される。各ノードは、ノードが直接アクセスを行う対象のＡＰＵによるメモリ・アクセスに適した論理処理を実行する。

【００５８】

図２０はＤＭＡＣの他の実施形態、すなわち、非分配型アーキテクチャを図示する。この場合、ＤＭＡＣ１６０６の構造上のハードウェアは集中型である。ＡＰＵ１６０２とＰＵ１６０４は、ローカルＰＥバス１６０７を介してＤＭＡＣ１６０６を用いて通信を行う。ＤＭＡＣ１６０６はクロスバー・スイッチを介してバス１６０８と接続される。バス１６０８はＤＲＡＭ１６１０と接続されている。

【００５９】

上述のように１つのＰＥの複数のＡＰＵのすべては、独立に、共用ＤＲＡＭ内のデータへのアクセスが可能である。その結果、第１のＡＰＵがあるデータをそのローカル・ストレージで処理しているときに、第２のＡＰＵがこれらのデータを要求する場合もある。その時点で共用ＤＲＡＭから第２のＡＰＵへ当該データが出力された場合、データの値を変化させ得る第１のＡＰＵの進行中の処理に起因して、そのデータが無効になる場合がある。したがって、その時点で第２のプロセッサが共用ＤＲＡＭからデータを受け取った場合、第２のプロセッサでエラー結果が生じるおそれがある。例えば、このようなデータとしては、グローバル変数用の具体的な値が挙げられる。第１のプロセッサがその処理中その値を変えた場合、第２のプロセッサはもう使用されていない値を受け取ることになる。したがって、共用ＤＲＡＭの範囲内でメモリ・ロケーションからのおよびメモリ・ロケーションへのＡＰＵによるデータの読み出しと書き込みを同期させる何らかの方式が必要となる。この方式では、別のＡＰＵがそのローカル・ストレージで現在働きかけている対象データであって、したがって最新のものではないデータのメモリ・ロケーションからの読み出

しと、最新のデータを格納するメモリ・ロケーションの中へのデータの書き込みと、を行わないようにする必要がある。

【００６０】

これらの問題を解決するために、ＤＲＡＭの各アドレス指定が可能なメモリ・ロケーションに対して、そのメモリ・ロケーションの中に格納されているデータに関連する状態情報を格納するために、ＤＲＡＭの中でメモリの追加セグメントの割り振りが行われる。この状態情報の中には、フル／エンプティ(Ｆ／Ｅ)ビットと、メモリ・ロケーションからデータを要求するＡＰＵの識別子(ＡＰＵ　ＩＤ)と、要求されたデータを読み出す読み出し先となるＡＰＵのローカル・ストレージのアドレス(ＬＳアドレス)とが含まれる。ＤＲＡＭのアドレス指定が可能なメモリ・ロケーションは任意のサイズとすることができる。ある好ましい実施形態ではこのサイズは１０２４ビットである。

【００６１】

Ｆ／Ｅビットの１への設定は、メモリ・ロケーションに格納されているデータが最新のものであることを示す。一方、Ｆ／Ｅビットの０への設定は、関連するメモリ・ロケーションに格納されたデータが最新のものではないことを示す。このビットが０に設定されているとき、ＡＰＵがそのデータを要求しても、ＡＰＵによってそのデータの即時読み出しは妨げられる。この場合、そのデータを要求しているＡＰＵを識別するＡＰＵ　ＩＤと、データが最新のものになっているとき、そのデータを読み出す読み出し先となるこのＡＰＵのローカル・ストレージ内のメモリ・ロケーションを識別するＬＳアドレスとが、追加メモリ・セグメントの中へ入力される。

【００６２】

また追加メモリ・セグメントは、ＡＰＵのローカル・ストレージ内の各メモリ・ロケーションに対して割り振られる。この追加メモリ・セグメントは、“ビジー・ビット”と呼ばれる１ビットを格納する。このビジー・ビットは、ＤＲＡＭから検索される固有データの格納用として関連するＬＳメモリ・ロケーションの予約を行うために使用される。ローカル・ストレージ内の特定のメモリ・ロケーションに対してビジー・ビットが１に設定されている場合、これらの固有データの書き込み用としてのみＡＰＵはこのメモリ・ロケーションを使用することができる。一方、ビジー・ビットが、ローカル・ストレージ内の特定のメモリ・ロケーションに対して０に設定されている場合、ＡＰＵは、任意のデータの書き込み用としてこのメモリ・ロケーションを使用することができる。

【００６３】

Ｆ／Ｅビット、ＡＰＵ　ＩＤ、ＬＳアドレスおよびビジー・ビットが、ＰＥの共用ＤＲＡＭからの、および、ＰＥの共用ＤＲＡＭへのデータの読み出しと書き込みを同期させるために使用される方法を示す例が図２１－３５に例示されている。

【００６４】

図２１に示すように、１以上のＰＥ(ＰＥ１７２０など)がＤＲＡＭ１７０２を使用する。ＰＥ１７２０にはＡＰＵ１７２２とＡＰＵ１７４０とが含まれる。ＡＰＵ１７２２には制御論理回路１７２４が含まれ、ＡＰＵ１７４０には制御論理回路１７４２が含まれる。ＡＰＵ１７２２にはローカル・ストレージ１７２６も含まれる。このローカル・ストレージには複数のアドレス可能なメモリ・ロケーション１７２８が含まれる。ＡＰＵ１７４０にはローカル・ストレージ１７４４が含まれ、このローカル・ストレージにも複数のアドレス可能なメモリ・ロケーション１７４６が含まれる。これらのアドレス可能なメモリ・ロケーションのすべては好適にはサイズが１０２４ビットであることが望ましい。

【００６５】

メモリの追加セグメントは各ＬＳのアドレス可能なメモリ・ロケーションと関連付けられる。例えば、メモリ・セグメント１７２９と１７３４とはそれぞれ、ローカルなメモリ・ロケーション１７３１と１７３２とに関連付けられ、メモリ・セグメント１７５２はローカルなメモリ・ロケーション１７５０と関連付けられる。上述のような“ビジー・ビット”はこれらの追加メモリ・セグメントの各々の中に格納される。ローカルなメモリ・ロケーション１７３２は、このメモリ・ロケーションがデータを含むことを示すいくつかの

×印を用いて示されている。

【００６６】

ＤＲＡＭ１７０２には、メモリ・ロケーション１７０６と１７０８とを含む複数のアドレス可能なメモリ・ロケーション１７０４が含まれる。これらのメモリ・ロケーションは、好適にはサイズが１０２４ビットであることが望ましい。メモリの追加セグメントはまたこれらのメモリ・ロケーションの各々とも関連付けられる。例えば、追加メモリ・セグメント１７６０はメモリ・ロケーション１７０６と関連し、追加メモリ・セグメント１７６２はメモリ・ロケーション１７０８と関連付けられる。各メモリ・ロケーションに格納されるデータに関連する状態情報は、メモリ・ロケーションと関連付けられたメモリ・セグメントに格納される。この状態情報の中には、上述のように、Ｆ／Ｅビット、ＡＰＵ　ＩＤおよびＬＳアドレスが含まれる。例えば、メモリ・ロケーション１７０８については、この状態情報にはＦ／Ｅビット１７１２、ＡＰＵ　ＩＤ１７１４およびＬＳアドレス１７１６が含まれる。

【００６７】

この状態情報とビジー・ビットとを用いて、ＰＥのＡＰＵ、または１グループのＰＥ間での、共用ＤＲＡＭからの、および、同期した共用ＤＲＡＭからの読み出しと、同期した共用ＤＲＡＭへのデータの書き込みを行うことができる。

【００６８】

図２２はＡＰＵ１７２２のＬＳメモリ・ロケーション１７３２から、ＤＲＡＭ１７０２のメモリ・ロケーション１７０８へのデータの同期書き込みの開始を例示する図である。ＡＰＵ１７２２の制御論理回路１７２４によってこれらのデータの同期書き込みが開始される。メモリ・ロケーション１７０８がエンプティであるため、Ｆ／Ｅビット１７１２は０に設定される。その結果、メモリ・ロケーション１７０８の中へＬＳメモリ・ロケーション１７３２内のデータを書き込むことが可能となる。一方、このビットが１に設定されていて、メモリ・ロケーション１７０８がフル状態であり、最新の有効データを含むことが示されている場合、制御回路１７２２はエラー・メッセージを受け取ることになり、このメモリ・ロケーションへのデータの書き込みは禁止される。

【００６９】

メモリ・ロケーション１７０８への成功したデータの同期書き込みの結果が図２３に示されている。この書き込まれたデータはメモリ・ロケーション１７０８の中に格納され、Ｆ／Ｅビット１７１２は１に設定される。この設定によって、メモリ・ロケーション１７０８がフル状態であること、および、このメモリ・ロケーションの中のデータが最新の有効データであることが示される。

【００７０】

図２４は、ＤＲＡＭ１７０２のメモリ・ロケーション１７０８からローカル・ストレージ１７４４のＬＳメモリ・ロケーション１７５０へのデータの同期読み出しの開始を例示する図である。この読み出しを開始するために、ＬＳメモリ・ロケーション１７５０のメモリ・セグメント１７５２の中のビジー・ビットが１に設定されて、上記データ用としてこのメモリ・ロケーションが予約される。このビジー・ビットを１に設定することによって、ＡＰＵ１７４０がこのメモリ・ロケーションに他のデータを格納することはなくなっている。

【００７１】

図２５に示すように、制御論理回路１７４２は次にＤＲＡＭ１７０２のメモリ・ロケーション１７０８に対して同期読取りコマンドを出す。このメモリ・ロケーションと関連付けられるＦ／Ｅビット１７１２は１に設定されているので、メモリ・ロケーション１７０８の中に格納されたデータは最新の、有効データであると見なされる。その結果、メモリ・ロケーション１７０８からＬＳメモリ・ロケーション１７５０へのデータ転送の準備の際に、Ｆ／Ｅビット１７１２は０に設定される。この設定が図２６に示されている。このビットを０に設定されているということは、これらのデータの読み出しの後に、メモリ・ロケーション１７０８のデータは無効になることを示す。

【００７２】
図２７に示すように、メモリ・ロケーション１７０８内のデータは、次に、メモリ・ロケーション１７０８からＬＳメモリ・ロケーション１７５０へ読み出される。図２８は最終状態を示す図である。メモリ・ロケーション１７０８のデータのコピーはＬＳメモリ・ロケーション１７５０に格納される。Ｆ／Ｅビット１７１２は０に設定され、メモリ・ロケーション１７０８のデータが無効であることが示される。この無効は、ＡＰＵ１７４０によって行われた上記データの変更の結果である。メモリ・セグメント１７５２内のビジー・ビットもまた０に設定される。この設定によって、ＡＰＵ１７４０がＬＳメモリ・ロケーション１７５０を任意の目的に利用できること、すなわち、このＬＳメモリ・ロケーションがもはや固有データの受信を待機している予約状態ではないことが示される。したがって、任意の目的のためにＡＰＵ１７４０によるＬＳメモリ・ロケーション１７５０へのアクセスが可能となる。

【００７３】
図２９～図３５には、ＤＲＡＭ１７０２のメモリ・ロケーション用のＦ／Ｅビットが、０に設定されていて、このメモリ・ロケーションのデータが最新のものでもなく有効なものでもないことが示されている場合の、ＤＲＡＭ１７０２(メモリ・ロケーション１７０８など)のメモリ・ロケーションから、ＡＰＵのローカル・ストレージ(ローカル・ストレージ１７４４のＬＳメモリ・ロケーション１７５２など)のＬＳメモリ・ロケーションへのデータの同期読み出しが例示されている。図２９に示すように、この転送を開始するために、ＬＳメモリ・ロケーション１７５０のメモリ・セグメント１７５２内のビジー・ビットは１に設定され、このデータ転送用としてこのＬＳメモリ・ロケーションが予約される。図３０に示すように、制御論理回路１７４２は、次に、ＤＲＡＭ１７０２のメモリ・ロケーション１７０８に対して同期読取りコマンドを出す。このメモリ・ロケーションと関連付けられたＦ／Ｅビット(Ｆ／Ｅビット１７１２)は０に設定されているので、メモリ・ロケーション１７０８に格納されているデータは無効である。その結果、信号は制御論理回路１７４２へ伝送され、このメモリ・ロケーションからのデータの即時読み出しが阻止される。

【００７４】
図３１に示すように、ＡＰＵ　ＩＤ１７１４とこの読取りコマンド用のＬＳアドレス１７１６とはメモリ・セグメント１７６２の中へ書き込まれる。この場合、ＡＰＵ１７４０用のＡＰＵ　ＩＤと、ＬＳメモリ・ロケーション１７５０用のＬＳメモリ・ロケーションとはメモリ・セグメント１７６２の中へ書き込まれる。したがって、メモリ・ロケーション１７０８の範囲内のデータが最新のものになっているとき、このＡＰＵ　ＩＤとＬＳメモリ・ロケーションは、最新のデータを伝送する伝送先のメモリ・ロケーションを決定するために使用される。

【００７５】
メモリ・ロケーション１７０８内のデータは、ＡＰＵがこのメモリ・ロケーションの中へデータを書き込むと、有効で最新のデータとなる。ＡＰＵ１７２２のメモリ・ロケーション１７３２などからメモリ・ロケーション１７０８の中へのデータの同期書き込みが図２９に例示されている。このメモリ・ロケーション用のＦ／Ｅビット１７１２が０に設定されているため、これらのデータのこの同期書き込みは許される。

【００７６】
図３３に示すように、この書き込み後、メモリ・ロケーション１７０８の中のデータは最新の有効データになる。したがって、メモリ・セグメント１７６２から得られるＡＰＵＩＤ１７１４とＬＳアドレス１７１６とは、メモリ・セグメント１７６２から即座に読み出され、次いでこの情報はこのセグメントから削除される。メモリ・ロケーション１７０８の中のデータの即時読み出しを予期して、Ｆ／Ｅビット１７１２もまた０に設定される。図３４に示すように、ＡＰＵ　ＩＤ１７１４とＬＳアドレス１７１６とを読み出すと、ＡＰＵ１７４０のＬＳメモリ・ロケーション１７５０へメモリ・ロケーション１７０８内の有効データを読み出すためにこの情報は直ちに使用される。最終状態が図３５に図示され

ている。この図は、メモリ・ロケーション１７０８からメモリ・ロケーション１７５０へコピーされた有効データと、０に設定されたメモリ・セグメント１７５２内のビジー・ビットと、０に設定されたメモリ・セグメント１７６２内のＦ／Ｅビット１７１２とを図示する。このビジー・ビットの０への設定によって、任意の目的のためにＡＰＵ１７４０がＬＳメモリ・ロケーション１７５０のアクセスを行うことが可能になる。このＦ／Ｅビットの０への設定によって、メモリ・ロケーション１７０８内のデータがもはや最新のものでもなく、有効なものでもないことが示される。

【００７７】

図３６は、上述のオペレーションと、ＤＲＡＭのメモリ・ロケーションの様々な状態とを要約する図であり、この状態は、Ｆ／Ｅビットの状態と、ＡＰＵ　ＩＤと、メモリ・ロケーションに対応するメモリ・セグメントの中に格納されたＬＳアドレスとに基づく。このメモリ・ロケーションは、３つの状態を持つことが可能である。これらの３つの状態として、Ｆ／Ｅビットが０に設定され、ＡＰＵＩＤまたはＬＳアドレスに対して情報が提供されないエンプティ状態１８８０と、Ｆ／Ｅビットが１に設定され、ＡＰＵ　ＩＤまたはＬＳアドレスに対して情報が提供されないフル状態１８８２と、Ｆ／Ｅビットが０に設定され、ＡＰＵ　ＩＤとＬＳアドレスに対して情報が提供されるブロッキング状態１８８４とがある。

【００７８】

この図に示すように、エンプティ状態１８８０では、同期書き込みオペレーションが許され、フル状態１８８２への遷移という結果が得られる。しかし、メモリ・ロケーションがエンプティ状態であるときはメモリ・ロケーション内のデータが最新のものではないので、同期読み出しオペレーションに対しては、ブロッキング状態１８８４へ遷移するという結果となる。

【００７９】

フル状態１８８２では、同期読み出しオペレーションが許され、エンプティ状態１８８０への遷移という結果が得られる。一方、有効データの上書きを避けるために、フル状態１８８２の同期書き込みオペレーションは禁止される。このような書き込みオペレーションがこの状態で試みられる場合、状態の変化は生じず、エラー・メッセージがＡＰＵの対応する制御論理回路へ伝送される。

【００８０】

ブロッキング状態１８８４では、メモリ・ロケーションの中へのデータの同期書き込みが許され、エンプティ状態１８８０への遷移という結果が得られる。一方、ブロッキング状態１８８４での同期読み出しオペレーションは禁止される。このブロッキング状態を生じさせることとなった前回同期読み出しオペレーションとのコンフリクトを阻止するためである。同期読み出しオペレーションが、ブロッキング状態１８８４で試みられた場合、状態変化は生じないでＡＰＵの対応する制御論理回路へエラー・メッセージが伝送される。

【００８１】

共用ＤＲＡＭからのデータの同期読み出しと、共用ＤＲＡＭへのデータの同期書き込みを行う上述の方式は、外部装置からのデータ読み出しと外部装置へのデータ書き込み用プロセッサとして通常専用の計算用リソースを取り除くためにも利用が可能である。この入出力(Ｉ／Ｏ)機能はＰＵによって行うこともできる。しかし、この同期方式の変更を利用して、適切なプログラムを実行するＡＰＵがこの機能を実行してもよい。例えば、この方式を利用して、外部装置によって開始された、Ｉ／Ｏインターフェースからのデータ伝送を求める割込み要求を受け取るＰＵは、このＡＰＵにこの要求の処理を委任してもよい。次いで、ＡＰＵはＩ／Ｏインターフェースに対して同期書き込みコマンドを出す。今度はこのインターフェースによって、現在ＤＲＡＭの中へデータを書き込むことができる旨の信号が外部装置へ送られる。次にＡＰＵはＤＲＡＭに対して同期読取りコマンドを出し、ＤＲＡＭの関連するメモリ空間をブロッキング状態に設定する。ＡＰＵはまた、データを受け取る必要があるＡＰＵのローカル・ストレージのメモリ・ロケーションに対してビジー・ビットを１に設定する。ブロッキング状態では、ＤＲＡＭの関連するメモリ空間と関連

付けられた追加メモリ・セグメントの中に、ＡＰＵのＩＤとＡＰＵのローカル・ストレージの関連するメモリ・ロケーションのアドレスが含まれる。次に外部装置は同期書き込みコマンドを出し、ＤＲＡＭの関連するメモリ空間へデータが直接書き込まれる。このメモリ空間はブロッキング状態にあるので、データは、このスペースの中から、追加メモリ・セグメントの中で識別されたＡＰＵのローカル・ストレージのメモリ・ロケーションの中へ直ちに読み出される。次いで、これらのメモリ・ロケーション用のビジー・ビットは０に設定される。外部装置がデータの書き込みを完了したとき、ＡＰＵは、伝送が完了した旨を示す信号をＰＵへ出す。

【００８２】

したがって、この方式を用いて、ＰＵに対する最小の計算上の負荷で、外部装置からのデータ転送処理を行うことができる。しかし、この機能を委任されたＡＰＵはＰＵに対して割込み要求を出せることが望ましく、外部装置がＤＲＡＭに対して直接アクセスを行うことが望ましい。

【００８３】

各ＰＥのＤＲＡＭには複数の“サンドボックス”が含まれる。サンドボックスによって共用ＤＲＡＭ領域が画定され、この領域を越えて、特定のＡＰＵまたは１組のＡＰＵがデータの読み出しや書き込みを行うことはできない。これらのサンドボックスによって、１つのＡＰＵが処理するデータに起因する、別のＡＰＵによって処理されるデータの破損に対するセキュリティが与えられる。またこれらのサンドボックスによって、ソフトウェア・セルが全ＤＲＡＭの中でデータの破損を生じる可能性なく、ネットワーク１０４から特定のサンドボックスの中へソフトウェア・セルのダウンロードを行うことが許される。本発明では、サンドボックスは、ＤＲＡＭとＤＭＡＣとから成るハードウェアの中に設けられる。ソフトウェアの代わりに、このハードウェア内にこれらのサンドボックスを設けることにより、速度とセキュリティという利点が得られる。

【００８４】

ＰＥのＰＵはＡＰＵへ割り当てられるサンドボックスの制御を行う。ＰＵは、オペレーティング・システムのような信頼のおけるプログラムだけしか通常作動させないので、本方式によってセキュリティが危険にさらされることはない。本方式に従って、ＰＵはキー管理テーブルの構築と維持とを行う。図３７にこのキー管理テーブルが例示されている。この図に示すように、キー管理テーブル１９０２内の各エントリには、ＡＰＵ用の識別子（ＩＤ)１９０４と、そのＡＰＵ用のＡＰＵキー１９０６と、キー・マスク１９０８とが含まれる。このキー・マスクの用途について以下説明する。キー管理テーブル１９０２は、スタティック・ランダム・アクセス・メモリ(ＳＲＡＭ)のような比較的高速のメモリに好適に格納され，ＤＭＡＣと関連付けられる。キー管理テーブル１９０２へのエントリはＰＵによって制御される。ＡＰＵが、ＤＲＡＭの特定の格納位置（ストレージロケーション）へのデータの書き込みあるいはＤＲＡＭの特定の格納位置からのデータの読み出しを要求すると、ＤＭＡＣは、その格納位置と関連付けられたメモリ・アクセス・キーに対して、キー管理テーブル１９０２内のそのＡＰＵへ割り当てられたＡＰＵキー１９０６の評価を行う。

【００８５】

図３８に示すように、ＤＲＡＭ２００２の各アドレス可能な格納位置２００６に対して専用メモリ・セグメント２０１０が割り当てられる。この格納位置用のメモリ・アクセス・キー２０１２はこの専用メモリ・セグメントの中に格納される。上述のように、やはり各アドレス可能な格納位置２００６と関連付けられたさらなる追加専用メモリ・セグメント２００８によって、格納位置へのデータ書き込みと、格納位置からのデータの読み出しを行うための同期情報が格納される。

【００８６】

作動時に、ＡＰＵはＤＭＡＣへＤＭＡコマンドを出す。このコマンドには、ＤＲＡＭ２００２の格納位置２００６のアドレスが含まれる。このコマンドを実行する前に、ＤＭＡＣは、キー管理テーブル１９０２におけるＡＰＵのＩＤ１９０４を用いて要求を行っている

ＡＰＵのキー１９０６を調べる。次いで、ＤＭＡＣは、ＡＰＵがアクセスを求める対象先であるＤＲＡＭの格納位置と関連付けられた専用メモリ・セグメント２０１０内に格納されているメモリ・アクセス・キー２０１２と、要求を行っているＡＰＵのＡＰＵキー１９０６との比較を行う。２つのキーが一致しない場合、ＤＭＡコマンドは実行されない。一方、２つのキーが一致した場合、ＤＭＡコマンドは進行し、要求されたメモリ・アクセスが実行される。

【００８７】

図３９に他の実施形態の一例を示す。この例では、ＰＵはメモリ・アクセス管理テーブル２１０２の維持も行う。メモリ・アクセス管理テーブル２１０２にはＤＲＡＭ内にある各サンドボックス用のエントリが含まれる。図３９の特定の例では、ＤＲＡＭには６４個のサンドボックスが含まれる。メモリ・アクセス管理テーブル２１０２内の各エントリには、サンドボックス用識別子(ＩＤ)２１０４と、ベース・メモリ・アドレス２１０６と、サンドボックス・サイズ２１０８と、メモリ・アクセス・キー２１１０と、アクセス・キーマスク２１１０とが含まれる。ベース・メモリ・アドレス２１０６によって、ＤＲＡＭ内にアドレスが設けられ、このアドレスによって特定のメモリ・サンドボックスの最初の部分が示される。サンドボックス・サイズ２１０８によってサンドボックスのサイズが与えられ、したがって、このサイズによって特定のサンドボックスのエンドポイントが与えられる。

【００８８】

図４０は、キー管理テーブル１９０２とメモリ・アクセス管理テーブル２１０２とを用いてＤＭＡコマンドを実行するためのステップを示すフロー・チャートである。ステップ２２０２では、ＡＰＵによって、サンドボックス内の特定の一つあるいは複数のメモリ・ロケーションに対するアクセス用ＤＭＡコマンドがＤＭＡＣへ出される。このコマンドには、アクセス要求を行う対象先である特定のサンドボックスの識別を行うサンドボックスＩＤ２１０４が含まれる。ステップ２２０４では、ＤＭＡＣは、ＡＰＵのＩＤ１９０４を利用して、キー管理テーブル１９０２内の要求を行っているＡＰＵのキー１９０６を調べる。ステップ２２０６で、ＤＭＡＣは、メモリ・アクセス管理テーブル２１０２で、サンドボックスと関連付けられたメモリ・アクセス・キー２１１０を調べるコマンドで、サンドボックスＩＤ２１０４を利用する。ステップ２２０８で、ＤＭＡＣは、要求を行っているＡＰＵへ割り当てられているＡＰＵキー１９０６をサンドボックスと関連付けられたアクセス・キー２１１０と比較する。ステップ２２１０で、この２つのキーが一致するかどうかの決定が行われる。この２つのキーが一致しない場合、処理はステップ２２１２へ移行し、そこでＤＭＡコマンドは先へ進まず、要求を行っているＡＰＵとＰＵのいずれかまたはその双方へエラー・メッセージが送信される。一方、ステップ２２１０で、２つのキーの一致が得られた場合、処理はステップ２２１４へ進み、そこでＤＭＡＣはＤＭＡコマンドを実行する。

【００８９】

ＡＰＵキー用およびメモリ・アクセス・キー用のキー・マスクによってこのシステムに大きな柔軟性が与えられる。キー用のキー・マスクによって、マスクされたビットはワイルド・カードに変換される。例えば、ＡＰＵキー１９０６と関連付けられたキー・マスク１９０８が、キー・マスク１９０８内のこれらのビットを１に設定することなどにより、その最後の２ビットが“マスク”に設定されている場合、ＡＰＵキーは１または０のいずれかになることができ、そのままメモリ・アクセス・キーに一致することになる。例えば、ＡＰＵキーが１０１０であるとする。通常、このＡＰＵキーによって１０１０のアクセス・キーを持つサンドボックスへのアクセスだけが可能になる。しかし、このＡＰＵキー用のＡＰＵキー・マスクが０００１に設定されている場合、このＡＰＵキーを用いて１０１０または１０１１のいずれかのアクセス・キーを持つサンドボックスへのアクセスを行うことが可能となる。同様に、１０１０または１０１１のいずれかのＡＰＵキーを持つＡＰＵによって、０００１に設定されたマスクを持つアクセス・キー１０１０のアクセスを行うことが可能である。ＡＰＵキー・マスクとメモリ・キー・マスクの双方を同時に使用す

ることができるので、多数のバリエーションのサンドボックスに対するＡＰＵによるアクセシビリティの設定が可能となる。

【００９０】

また本発明はシステム１０１のプロセッサ用の新しいプログラミング・モデルも提供するものである。このプログラミング・モデルではソフトウェア・セル１０２が用いられる。ネットワーク１０４上の任意のプロセッサへ処理用としてこれらのセルの伝送を行うことが可能である。またこの新しいプログラミング・モデルでは、システム１０１のユニークなモジュラー形アーキテクチャと、システム１０１のプロセッサとが利用される。

【００９１】

ソフトウェア・セルはＡＰＵのローカル・ストレージからＡＰＵによって直接処理される。ＡＰＵは、ＤＲＡＭ内のいずれのデータまたはプログラムに対しても直接働きかけることは行わない。ＤＲＡＭ内のデータとプログラムは、ＡＰＵがこれらのデータとプログラムの処理を行う前に、ＡＰＵのローカル・ストレージの中に読み込まれる。したがって、ＡＰＵのローカル・ストレージには、プログラム・カウンタと、スタックと、これらのプログラムを実行するための他のソフトウェア・エレメントとが含まれることになる。ＰＵは、ＤＭＡＣに対してＤＭＡコマンドを出すことによりＡＰＵの制御を行う。

【００９２】

ソフトウェア・セル１０２の構造が図４１に例示されている。この図に示すように、ソフトウェア・セル２３０２などのソフトウェア・セルの中には、ルート選定情報セクション２３０４と本体部分２３０６とが含まれる。ルート選定情報セクション２３０４に含まれる情報は、ネットワーク１０４のプロトコルに依って決められる。ルート選定情報セクション２３０４の中には、ヘッダ２３０８、宛先ＩＤ２３１０、ソースＩＤ２３１２および応答ＩＤ２３１４が含まれる。宛先ＩＤにはネットワーク・アドレスが含まれる。ＴＣＰ／ＩＰプロトコルの下で、例えば、ネットワーク・アドレスはインターネット・プロトコル(ＩＰ)アドレスである。さらに宛先ＩＤ２３１０には、処理のためにセルを伝送すべき伝送先のＰＥ及びＡＰＵの識別子が含まれる。ソースＩＤ２３１４にはネットワーク・アドレスが含まれ、このソースＩＤによってＰＥとＡＰＵとが識別され、このＰＥとＡＰＵとからセルが起動し、必要な場合に、宛先ＰＥとＡＰＵとがセルに関する追加情報を得ることが可能となる。応答ＩＤ２３１４にはネットワーク・アドレスが含まれ、この応答ＩＤ２３１４によって、セルに関するクエリとセルの処理の結果とを送る送り先のＰＥとＡＰＵとが識別される。

【００９３】

セルの本体部分２３０６にはネットワークのプロトコルとは無関係の情報が含まれる。図４１の分解部分はセルの本体部分２３０６の細部を図示する。セルの本体部分２３０６のヘッダ２３２０によってセル本体の開始部が識別される。セル・インターフェース２３２２にはセルの利用に必要な情報が含まれる。この情報の中には、グローバルな一意的ＩＤ２３２４と、要求されるＡＰＵ２３２６と、サンドボックス・サイズ２３２８と、前回のセルのＩＤ２３３０とが含まれる。

【００９４】

グローバルな一意的ＩＤ２３２４によって、ネットワーク１０４全体を通じてソフトウェア・セル２３０２が一意的に識別される。グローバルな一意的ＩＤ２３２４が、ソースＩＤ２３１２(ソースＩＤ２３１２内のＰＥまたはＡＰＵの一意的識別子など)と、ソフトウェア・セル２３０２の作成または伝送の時刻と日付とに基づいて作成される。必要なＡＰＵ２３２６によってセルの実行に必要な最低数のＡＰＵが与えられる。サンドボックス・サイズ２３２８によって、セルの実行に必要なＤＲＡＭと関連する必要なＡＰＵ内に、保護されたメモリ量が与えられる。前回のセルＩＤ２３３０によって、シーケンシャルな実行を要求する１グループのセル(ストリーミング・データなど)内の前回のセルの識別子が提供される。

【００９５】

実行セクション２３３２の中にはセルのコア情報が含まれる。この情報の中にはＤＭＡコ

マンド・リスト２３３４と、プログラム２３３６と、データ２３３８とが含まれる。プログラム２３３６には、ＡＰＵプログラム２３６０と２３６２などのＡＰＵによって実行されるプログラム(“アプレット”と呼ばれる)が含まれ、データ２３３８にはこれらのプログラムを用いて処理されるデータが含まれる。ＤＭＡコマンド・リスト２３３４には、プログラムの起動に必要な一連のＤＭＡコマンドが含まれる。これらのＤＭＡコマンドにはＤＭＡコマンド　２３４０、２３５０、２３５５、２３５８が含まれる。ＰＵはＤＭＡＣへこれらのＤＭＡコマンドを出す。

【００９６】

ＤＭＡコマンド２３４０にはＶＩＤ２３４２が含まれる。ＶＩＤ２３４２は、ＤＭＡコマンドが出されたとき物理ＩＤに対して対応づけられるＡＰＵのバーチャルＩＤである。ＤＭＡコマンド２３４０にはロード・コマンド２３４４とアドレス２３４６も含まれる。ロード・コマンド２３４４は、ＡＰＵにＤＲＡＭから特定の情報を読み出しローカル・ストレージの中へ入れるように命令する。アドレス２３４６によってこの特定情報を含むＤＲＡＭ内のバーチャル・アドレスが与えられる。この特定情報は、プログラム・セクション２３３６からのプログラムや、データ・セクション２３３８からのデータや、あるいはその他のデータなどであってもよい。最終的に、ＤＭＡコマンド２３４０にはローカル・ストレージのアドレス２３４８が含まれる。このアドレスによって、情報をロードできそうなローカル・ストレージのアドレスが識別される。ＤＭＡコマンド２３５０には類似の情報が含まれる。その他のＤＭＡコマンドも使用可能である。

【００９７】

ＤＭＡコマンド・リスト２３３４には一連のキック・コマンド(キック・コマンド２３５５と２３５８など)も含まれる。キック・コマンドとは、ＰＵによってＡＰＵへ出されるセルの処理を開始するコマンドである。ＤＭＡキック・コマンド２３５５には、バーチャルＡＰＵ　ＩＤ２３５２と、キック・コマンド２３５４と、プログラム・カウンタ２３５６とが含まれる。バーチャルＡＰＵ　ＩＤ２３５２はキックすべき対象ＡＰＵを識別し、キック・コマンド２３５４は関連するキック・コマンドを与え、プログラム・カウンタ２３５６は、プログラムの実行用プログラム・カウンタのためのアドレスを与える。ＤＭＡキック・コマンド２３５８は、同じＡＰＵまたは別のＡＰＵに対して同様の情報を与える。

【００９８】

上述したように、ＰＵは独立したプロセッサとしてＡＰＵを扱い、コプロセッサとして扱うものではない。したがって、ＡＰＵによる処理を制御するために、ＰＵは、遠隔手順呼出しに類似したコマンドを使用する。これらのコマンドは“ＡＰＵ遠隔手順呼出し(ＡＲＰＣ)”と呼ばれる。ＰＵは、一連のＤＭＡコマンドをＤＭＡＣへ出すことによりＡＲＰＣを実行する。ＤＭＡＣは、ＡＰＵプログラムとそれと関連するスタック・フレームとをＡＰＵのローカル・ストレージの中へロードする。次いで、ＰＵはＡＰＵへ最初のキックを出し、ＡＰＵプログラムを実行する。

【００９９】

図４２は、アプレットを実行するためのＡＲＰＣのステップを例示する。指定ＡＰＵによるアプレットの処理の開始時にＰＵが実行するこれらのステップが、図４２の第１の部分２４０２に示され、指定ＡＰＵが実行するステップが、図４２の第２の部分２４０４に示されている。

【０１００】

ステップ２４１０で、ＰＵはアプレットを評価し、次いで、アプレットの処理用ＡＰＵを指定する。ステップ２４１２で、ＰＵは、必要な単複のサンドボックス用のメモリ・アクセス・キーの設定を行うＤＭＡコマンドをＤＭＡＣへ出すことにより、アプレットの実行用スペースをＤＲＡＭ内に割り振る。ステップ２４１４で、ＰＵは、指定ＡＰＵへの割込み要求による、アプレットの完了信号の伝送を可能にする。ステップ２４１８で、ＰＵは、ＤＲＡＭからＡＰＵのローカル・ストレージへアプレットをロードするＤＭＡコマンドをＤＭＡＣへ出す。ステップ２４２０で、ＤＭＡコマンドが実行され、アプレットがＤＲ

ＡＭからローカル・ストレージへ読み出される。ステップ２４２２で、ＰＵは、アプレットと関連付けられたスタック・フレームをＤＲＡＭからＡＰＵのローカル・ストレージへロードするＤＭＡコマンドをＤＭＡＣへ出す。ステップ２４２３で、ＤＭＡコマンドが実行され、スタック・フレームがＤＲＡＭからＡＰＵのローカル・ストレージへ読み出される。ステップ２４２４で、ＰＵは、ＤＭＡＣがＡＰＵへキーを割り当てて、ステップ２４１２で指定された、一又は複数のハードウェア・サンドボックスからのデータ読み出しと、その一又は複数のハードウェア・サンドボックスへのデータ書き込みを行うことをＡＰＵに許可するＤＭＡコマンドを出す。ステップ２４２６で、ＤＭＡＣは、ＡＰＵへ割り当てられたキーを用いてキー管理テーブル(ＫＴＡＢ)の更新を行う。ステップ２４２８で、ＰＵは、プログラムの処理を開始するＤＭＡコマンド“キック”をＡＰＵへ出す。特定のアプレットに応じて、特定のＡＲＰＣの実行時にＰＵによって他のＤＭＡコマンドを出してもよい。

【０１０１】

上記のように、図４２の第２の部分２４０４は、アプレットの実行時にＡＰＵによって行われるステップを例示するものである。ステップ２４３０で、ＡＰＵは、ステップ２４２８で出されるキック・コマンドに応じてアプレットの実行を開始する。ステップ２４３２で、アプレットの指示で、ＡＰＵは、アプレットの関連スタック・フレームの評価を行う。ステップ２４３４で、ＡＰＵは、ＤＭＡＣへ複数のＤＭＡコマンドを出し、スタック・フレームが必要に応じてＤＲＡＭからＡＰＵのローカル・ストレージへ指定するデータのロードを行う。ステップ２４３６で、これらのＤＭＡコマンドが実行され、データは、ＤＲＡＭからＡＰＵのローカル・ストレージへ読み出される。ステップ２４３８でＡＰＵはアプレットを実行し、ある結果を出力する。ステップ２４４０で、ＡＰＵはＤＭＡＣへＤＭＡコマンドを出し、ＤＲＡＭにその結果を格納する。ステップ２４４２で、ＤＭＡコマンドが実行され、アプレットの結果がＡＰＵのローカル・ストレージからＤＲＡＭへ書き込まれる。ステップ２４４４で、ＡＰＵはＰＵへ割込み要求を出し、ＡＲＰＣが完了したことを示す信号伝送を行う。

【０１０２】

ＰＵの指示の下で独立にタスクを実行するＡＰＵの能力によって、１グループのＡＰＵと、１グループのＡＰＵと関連付けられたメモリ・リソースとを拡張タスクの実行専用にすることが可能になる。例えば、１つのＰＵは、１以上のＡＰＵと、これらの１以上のＡＰＵと関連付けられた１グループのメモリサンドボックスとを、拡張された時間中ネットワーク１０４を介して伝送されてくるデータの受信専用とし、また、１以上の他のＡＰＵとそれらと関連付けられたメモリ・サンドボックスへ、この時間中受信したデータのさらなる処理を行うための送信専用とすることができる。この能力は、ネットワーク１０４を介して伝送されるストリーミング・データ(ストリーミングＭＰＥＧまたはストリーミングＡＴＲＡＣオーディオまたはビデオ・データなど)の処理にとって特に好適である。ＰＵは、１以上のＡＰＵおよびそれらと関連付けられたメモリ・サンドボックスをこれらのデータの受信専用とし、１以上の他のＡＰＵおよびそれらと関連付けられたメモリ・サンドボックスをこれらのデータの解凍と処理専用とすることができる。言い換えれば、ＰＵは、ＡＰＵのグループとそれらと関連付けられたメモリ・サンドボックスとの間でこのようなデータ処理を行うための専用パイプライン関係の確立を行うことができる。

【０１０３】

しかし、このような処理を効率的に実行するためには、パイプ・ラインの専用ＡＰＵとメモリサンドボックスとが、データ・ストリームを含むアプレットの処理が行われない時間中もパイプ・ライン専用のままであることが望ましい。言い換えれば、専用ＡＰＵおよびそれらと関連するサンドボックスが、これらの時間中予約状態のままに置かれることが望ましい。アプレットの処理の完了時における、ＡＰＵとその関連付けられた一又は複数のメモリ・サンドボックスを予約、即ちリザーブ状態としておくことは、“常駐終了”と呼ばれる。常駐終了はＰＵからの命令に応じて行われる。

【０１０４】

図４３、４４、４５は、１グループのＡＰＵおよびそれらと関連するサンドボックスを含む、ストリーミング・データ(ストリーミングＭＰＥＧデータなど)を処理するための専用パイプライン構造の設定を例示する。図４３に示すように、このパイプライン構造の構成要素にはＰＥ２５０２とＤＲＡＭ２５１８とが含まれる。ＰＥ２５０２の中には、ＰＵ２５０４、ＤＭＡＣ２５０６およびＡＰＵ２５０８、ＡＰＵ２５１０、ＡＰＵ２５１２を含む複数のＡＰＵが含まれる。ＰＵ２５０４、ＤＭＡＣ２５０６およびこれらのＡＰＵ間の通信はＰＥバス２５１４を介して行われる。広帯域幅のバス２５１６によってＤＭＡＣ２５０６はＤＲＡＭ２５１８と接続される。ＤＲＡＭ２５１８の中には、複数のサンドボックス(サンドボックス２５２０、サンドボックス２５２２、サンドボックス２５２４、サンドボックス２５２６など)が含まれる。

【０１０５】

図４４に、専用パイプラインを設定するためのステップを例示する。ステップ２６１０で、ＰＵ２５０４は、ネットワーク・アプレットを処理するようにＡＰＵ２５０８を割り当てる。ネットワーク・アプレットは、ネットワーク１０４のネットワーク・プロトコルの処理用プログラムを有する。この場合、このプロトコルは　伝送制御プロトコル／インターネット用プロトコル(ＴＣＰ／ＩＰ)である。このプロトコルに従うＴＣＰ／ＩＰデータ・パケットはネットワーク１０４を介して伝送される。受信時に、ＡＰＵ２５０８はこれらのパケットを処理し、パケット内のデータを組み立て、ソフトウェア・セル１０２の中へ入れる。ステップ２６１２で、ＰＵ２５０４は、ネットワーク・アプレットの処理の完了時に常駐終了を実行するようにＡＰＵ２５０８に指示する。ステップ２６１４で、ＰＵ２５０４は、ＡＰＵ２５１０及び２５１２がＭＰＥＧアプレットの処理を行うように割り当てる。ステップ２６１５で、ＰＵ２５０４は、ＭＰＥＧアプレットの処理の完了時に常駐終了を実行するようにＡＰＵ２５１０及び２５１２に指示する。ステップ２６１６で、ＰＵ２５０４は、ＡＰＵ２５１０によるアクセス用ソース・サンドボックス及びＡＰＵ２５０８によるアクセス用宛先サンドボックスとしてサンドボックス２５２０を指定する。ステップ２６１８で、ＰＵ２５０４は、ＡＰＵ２５１０によるアクセス用宛先サンドボックス及びＡＰＵ２５１２によるアクセス用ソースサンドボックスとしてサンドボックス２５２２を指定する。ステップ２６２０で、ＰＵ２５０４は、ＡＰＵ２５１２によるアクセス用宛先サンドボックス及びパイプライン内の後段のＡＰＵによるアクセス用ソースサンドボックスとしてサンドボックス２５２４を指定する。ステップ２６２２で、ＰＵ２５０４は、パイプライン内の後段のＡＰＵによるアクセス用宛先サンドボックス及びアクセス用ソースサンドボックスとしてサンドボックス２５２６を指定する。ステップ２６２４で、ＡＰＵ２５１０とＡＰＵ２５１２とは、それぞれ、ソース・サンドボックス２５２０とソース・サンドボックス２５２２の範囲内のメモリ・ブロックへ同期読取りコマンドを送り、これらのメモリ・ブロックをブロッキング状態に設定する。最後に、処理はステップ２６２８へ移り、そこで、専用パイプラインの設定が完了し、パイプ・ライン専用のリソースが予約される。このようにして、ＡＰＵ２５０８、２５１０、２５１２等およびそれらと関連するサンドボックス２５２０、２５２２、２５２４および２５２６は予約状態に入る。

【０１０６】

図４５に、この専用パイプラインによるストリーミングＭＰＥＧデータの処理ステップを例示する。ステップ２６３０で、ＡＰＵ２５０８は、ネットワーク・アプレットを処理し、そのローカル・ストレージの中で、ＴＣＰ／ＩＰデータ・パケットをネットワーク１０４から受信する。ステップ２６３２で、ＡＰＵ２５０８は、これらのＴＣＰ／ＩＰデータ・パケットを処理し、これらのパケット内のデータをアセンブルし、ソフトウェア・セル１０２の中へ入れる。ステップ２６３４で、ＡＰＵ２５０８はソフトウェア・セルのヘッダ２３２０(図２３)をチェックし、セルがＭＰＥＧデータを含むかどうかの判定を行う。セルがＭＰＥＧデータを含まない場合、ステップ２６３６で、ＡＰＵ２５０８は、専用パイプライン内に含まれない他のＡＰＵによって他のデータを処理するために、ＤＲＡＭ２５１８内に指定される汎用サンドボックスへそのセルを伝送する。またＡＰＵ２５０８は

この伝送についてＰＵ２５０４に通知する。
【０１０７】
一方、ソフトウェア・セルがＭＰＥＧデータを含む場合、ステップ２６３８で、ＡＰＵ２５０８はそのセルの前のセルのＩＤ２３３０(図４１)をチェックし、そのセルが属するＭＰＥＧデータ・ストリームを識別する。ステップ２６４０で、ＡＰＵ２５０８はセルの処理用の専用パイプラインのＡＰＵを選択する。この場合、ＡＰＵ２５０８は、これらのデータを処理するＡＰＵ２５１０を選択する。この選択は前回のセルＩＤ２３３０とロード・バランシング・ファクタ（負荷平衡係数）とに基づく。例えば、そのソフトウェア・セルが属するＭＰＥＧデータ・ストリームの前回のソフトウェア・セルが処理用としてＡＰＵ２５１０へ送られたことが前のセルＩＤ２３３０によって示されている場合、現在のソフトウェア・セルも通常の処理用としてＡＰＵ２５１０へ送られる。ステップ２６４２で、ＡＰＵ２５０８は、サンドボックス２５２０へＭＰＥＧデータを書き込む同期書き込みコマンドを出す。このサンドボックスは予めブロッキング状態に設定されているので、ステップ２６４４で、ＭＰＥＧデータは、サンドボックス２５２０からＡＰＵ２５１０のローカル・ストレージへ自動的に読み出される。ステップ２６４６で、ＡＰＵ２５１０はそのローカル・ストレージでＭＰＥＧデータを処理してビデオ・データを生成する。ステップ２６４８で、ＡＰＵ２５１０はサンドボックス２５２２へビデオ・データを書き込む。ステップ２６５０で、ＡＰＵ２５１０はサンドボックス２５２０へ同期読取りコマンドを出し、このサンドボックスに追加ＭＰＥＧデータの受信を準備させる。ステップ２６５２で、ＡＰＵ２５１０は常駐終了処理を行う。この処理によってこのＡＰＵは予約状態に入り、この予約状態の間ＡＰＵは、ＭＰＥＧデータ・ストリームの中で追加ＭＰＥＧデータの処理を行うべく待機する。
【０１０８】
他のタイプのデータ処理用として１グループのＡＰＵおよびそれらと関連するサンドボックス間でその他の専用構造の設定が可能である。例えば、図４６に示すように、ＡＰＵの専用グループ(ＡＰＵ２７０２、２７０８、２７１４など)を設定し、３次元オブジェクトに対して幾何学的変換を実行して２次元ディスプレイ・リストの生成を行うことが可能となる。これらの２次元ディスプレイ・リストを他のＡＰＵによってさらに処理(レンダー)し画素データの生成を行うようにすることが可能である。この処理を実行するために、３次元オブジェクトと、これらのオブジェクト処理から結果として生じるディスプレイ・リストの格納用として、サンドボックスが、ＡＰＵ２７０２、２７０８、２４１４の専用となる。例えば、ソース・サンドボックス２７０４、２７１０、２７１６は、それぞれ、ＡＰＵ２７０２、ＡＰＵ２７０８、ＡＰＵ２７１４によって処理された３次元オブジェクトの格納専用となる。同様に、宛先サンドボックス２７０６、２７１２、２７１８は、それぞれ、ＡＰＵ２７０２、ＡＰＵ２７０８、ＡＰＵ２７１４によるこれらの３次元オブジェクトの処理から結果として生じるディスプレイ・リストの格納専用となる。
【０１０９】
調整用ＡＰＵ２７２０は、そのローカル・ストレージにおける、宛先サンドボックス２７０６、２７１２、２７１８からのディスプレイ・リストの受信専用である。ＡＰＵ２７２０は、これらのディスプレイ・リスト間での調整を行い、画素データのレンダリングのためにこれらのディスプレイ・リストを他のＡＰＵへ送る。
【０１１０】
システム１０１のプロセッサは絶対タイマーも使用する。この絶対タイマーはＡＰＵとＰＥの他のエレメントへクロック信号を出力する。このクロック信号はこれらのエレメントを駆動するクロック信号に依存せず、かつ、このクロック信号より高速である。この絶対タイマーの利用が図４７に例示されている。
【０１１１】
この図に示すように、この絶対タイマーによってＡＰＵによるタスク・パフォーマンスのためのタイム・バジェット（割り当て時間）が決定される。このタイム・バジェットによって、これらのタスクの完了時間が設定されるが、この時間はＡＰＵによるタスク処理に

必要な時間より長い時間になる。その結果、各タスクについて、タイム・バジェットの範囲内に、ビジーな時間とスタンバイ時間とが存在することになる。すべてのアプレットは、ＡＰＵの実際の処理時間にかかわらず、このタイム・バジェットに基づいて処理を行うように書かれる。

【０１１２】

例えば、ＰＥの特定のＡＰＵ用として、タイム・バジェット２８０４のビジー時間２８０２中に特定のタスクを行うことができる。ビジー時間２８０２がタイム・バジェット２８０４未満であるため、スタンバイ時間２８０６がタイム・バジェット中に生じる。このスタンバイ時間中、ＡＰＵは、ＡＰＵが消費するパワーが少なくなるスリープモードに入る。

【０１１３】

タイム・バジェット２８０４が満了するまでまで、他のＡＰＵまたはＰＥの他のエレメントがタスク処理の結果を予想することはない。したがって、ＡＰＵの実際の処理速度にかかわらず、絶対タイマーによって決定されるタイム・バジェットを用いてＡＰＵの処理結果が常時調整される。

【０１１４】

将来、ＡＰＵによる処理速度はさらに高速になる。しかし、絶対タイマーによって設定されるタイム・バジェットは同じままである。例えば、図４７に示すように、将来のＡＰＵは、さらに短時間でタスクを実行することになり、したがって、スタンバイ時間はさらに長くなるであろう。したがって、ビジー時間２８０８はビジー時間２８０２より短くなり、スタンバイ時間２８１０はスタンバイ時間２８０６より長くなる。しかし、絶対タイマーによって設定された同じタイム・バジェットに基づいて処理を行うようにプログラムが書かれているので、ＡＰＵ間での処理結果の調整が維持される。その結果、さらに高速のＡＰＵが、その処理の結果が予想される時点でコンフリクトを生じることなく、低速のＡＰＵ用として書かれたプログラムの処理を行うことが可能となる。

【０１１５】

動作速度の向上や動作速度が異なることに起因するＡＰＵの並列処理の調整問題に対しては、ＡＰＵ間での調整を決定する絶対タイマーに代えて、ＰＵまたは１以上の指定ＡＰＵにおいて、ＡＰＵが実行している特定の命令(マイクロコード)の分析をアプレットの処理時に行うようにすることもできる。“オペレーションなし”(“ＮＯＯＰ”)命令を命令の中へ挿入し、ＡＰＵのいくつかによってこの命令を実行して、アプレットによって予想されるＡＰＵによる処理を１ステップずつ適切に行うことが可能となる。命令の中へこれらのＮＯＯＰを挿入することにより、すべての命令のＡＰＵによる実行を行うための正しいタイミングの維持が可能となる。

【０１１６】

以上特定の実施形態に関して本明細書で本発明について説明したが、これらの実施形態は本発明の原理と適用を示す単に例示的なものであると理解すべきである。したがって、添付の請求項によって画定されているような本発明の精神と範囲から逸脱することなく、以上の例示の実施形態に対して多数の改変を行うことが可能であり、また、他の構成を考案することが可能である。

【図面の簡単な説明】

【図１】　本発明によるコンピュータ・ネットワークのアーキテクチャ全体を例示する。

【図２】　本発明によるプロセッサ・エレメント(ＰＥ)の構造を例示する図である。

【図３】　本発明による広帯域エンジン(ＢＥ)の構造を例示する図である。

【図４】　本発明による付加処理ユニット(ＡＰＵ)の構造を例示する図である。

【図５】　本発明によるプロセッサ・エレメントと、ビジュアライザ(ＶＳ)と、光インターフェースとの構造を例示する図である。

【図６】　本発明によるプロセッサ・エレメントの１つの組合せを例示する図である。

【図７】　本発明によるプロセッサ・エレメントの別の組合せを例示する図である。

【図８】　本発明によるプロセッサ・エレメントのさらに別の組合せを例示する図である

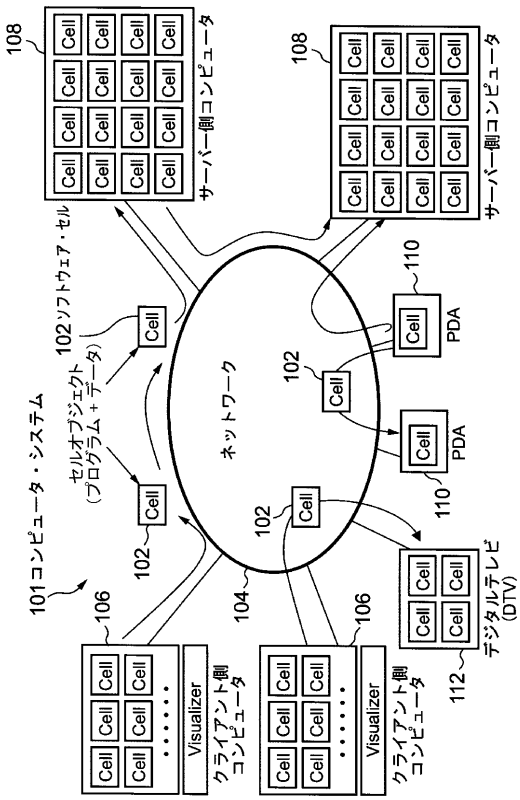
。
【図９】　本発明によるプロセッサ・エレメントのさらに別の組合せを例示する図である。
【図１０】　本発明によるプロセッサ・エレメントのさらに別の組合せを例示する図である。
【図１１】　本発明によるチップ・パッケージ内での光インターフェースの統合化を例示する図である。
【図１２】　図１１の光インターフェースを用いるプロセッサの１つの構成を示す図である。
【図１３】　図１１の光インターフェースを用いるプロセッサの別の構成を示す図である。
【図１４】　本発明によるメモリ・システムの構造を例示する図である。
【図１５】　本発明による第１の広帯域エンジンから第２の広帯域エンジンへのデータの書き込みを例示する図である。
【図１６】　本発明によるプロセッサ・エレメントための共用メモリの構造を示す図である。
【図１７】　図１６に示すメモリ・バンク用の１つの構造を例示する図である。
【図１８】　図１６に示すメモリ・バンク用の別の構造を例示する図である。
【図１９】　本発明によるＤＭＡＣのための構造を例示する図である。
【図２０】　本発明によるＤＭＡＣのための代替の構造を例示する図である。
【図２１】　本発明によるデータ同期オペレーションを例示する図である。
【図２２】　本発明によるデータ同期オペレーションを例示する図である。
【図２３】　本発明によるデータ同期オペレーションを例示する図である。
【図２４】　本発明によるデータ同期オペレーションを例示する図である。
【図２５】　本発明によるデータ同期オペレーションを例示する図である。
【図２６】　本発明によるデータ同期オペレーションを例示する図である。
【図２７】　本発明によるデータ同期オペレーションを例示する図である。
【図２８】　本発明によるデータ同期オペレーションを例示する図である。
【図２９】　本発明によるデータ同期オペレーションを例示する図である。
【図３０】　本発明によるデータ同期オペレーションを例示する図である。
【図３１】　本発明によるデータ同期オペレーションを例示する図である。
【図３２】　本発明によるデータ同期オペレーションを例示する図である。
【図３３】　本発明によるデータ同期オペレーションを例示する図である。
【図３４】　本発明によるデータ同期オペレーションを例示する図である。
【図３５】　本発明によるデータ同期オペレーションを例示する図である。
【図３６】　本発明のデータ同期方式によるメモリ・ロケーションの様々な状態を例示する３つの状態のメモリ図である。
【図３７】　本発明によるハードウェア・サンドボックス用のキー管理テーブルの構造を例示する図である。
【図３８】　本発明によるハードウェア・サンドボックス用メモリ・アクセス・キーの格納方式を例示する図である。
【図３９】　本発明によるハードウェア・サンドボックス用メモリ・アクセス管理テーブルの構造を例示する図である。
【図４０】　図３７のキー管理テーブルと図３９のメモリ・アクセス管理テーブルとを用いてメモリ・サンドボックスにアクセスするステップを示すフロー・チャートである。
【図４１】　本発明によるソフトウェア・セルの構造を例示する図である。
【図４２】　本発明による、ＡＰＵへ遠隔処理命令を出すステップを示すフロー・チャートである。
【図４３】　本発明による、ストリーミング・データ処理用専用パイプラインの構造を例示する図である。

【図４４】　本発明によるストリーミング・データの処理時の図４３の専用パイプラインによって実行されるステップを示すフロー・チャートである。
【図４５】　本発明によるストリーミング・データの処理時の図４３の専用パイプラインによって実行されるステップを示すフロー・チャートである。
【図４６】　本発明によるストリーミング・データ処理用の専用パイプラインの他の構造を例示する図である。
【図４７】　本発明によるＡＰＵによるアプリケーションとデータの並列処理を調整するための絶対タイマー方式を例示する図である。
【符号の説明】

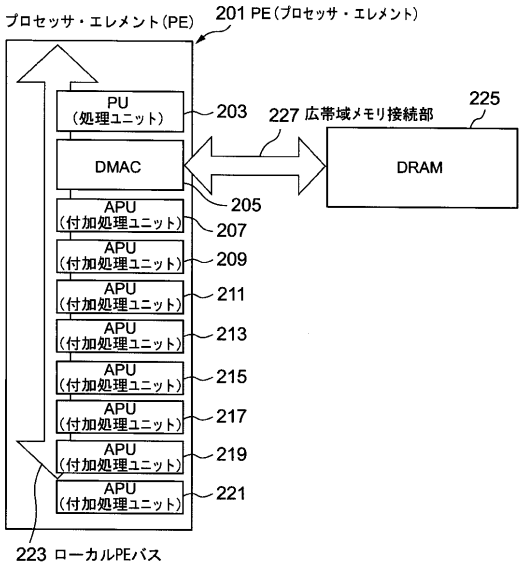
１０１	システム
１０１０	キー
１０２	セル
１０４	ネットワーク
１０６	クライアント
１０８	サーバーコンピュータ
１１０４	光インターフェース
１１０８	バス
１１１８	ポート
１１２２	ポート
１１２６	光導波路
１１６０	光インターフェース
１１６２	光インターフェース
１１６４	光インターフェース
１１６６	光インターフェース
１１８２	光インターフェース
１１８４	光インターフェース
１１８６	光インターフェース
１１８８	光インターフェース
１１８８	光インターフェース
１１９０	光インターフェース
１１９０	光インターフェース
１２０６	コントロール
１２１２	ユニット
１２２１	クロスバ交換機
１２３２	外部ポート
１２３４	コントロール
１２４０	ユニット
１２４２	コントロール
１２４４	バンク
１４０６	ブロック
１４１４	バンク
１４１６	バンク
１５０４	ノード
１６０７	バス
１６０８	バス
１７２２	制御回路
１７２４	制御論理回路
１７２６	ストレージ
１７２８	ロケーション
１７２９	セグメント

１７３１　ロケーション
１７３２　ロケーション
１７４２　制御論理回路
１７４６　ロケーション
１７５０　ロケーション
１７５２　セグメント
１７６０　セグメント
１７６２　セグメント
１８８０　エンプティ状態
１８８２　フル状態
１８８４　ブロッキング状態
１９０２　キー管理テーブル
１９０６　キー
１９０８　マスク
２００６　格納位置
２００８　セグメント
２０１０　セグメント
２０１２　キー
２１０２　アクセス管理テーブル
２１０６　アドレス
２１１０　キー
２１１０　キーマスク
２２３　バス
２２７　高帯域メモリ接続部
２３０２　セル
２３０８　ヘッダ
２３２０　ヘッダ
２３２２　インターフェース
２３３２　実行セクション
２３３４　リスト
２５２０　サンドボックス
２５２２　サンドボックス
２５２４　サンドボックス
２５２６　サンドボックス
２７０４　サンドボックス
２７０６　宛先サンドボックス
３０１　広帯域エンジン
３１１　バス
３１３　広帯域メモリ接続部
３１７　インターフェース
３１９　外部バス
４０６　メモリ
４０８　内部バス
４１０　レジスタ
４１２　浮動小数点演算ユニット
４１４　整数演算ユニット
４２０　バス
５０６　パッケージの中に光インターフェース
５０８　エンジン
５１０　画像用キャッシュ

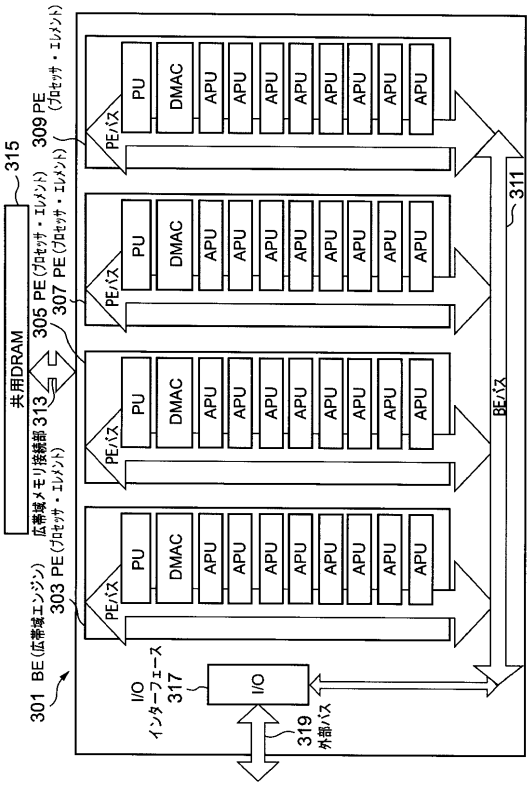
【図１】


101コンピュータ・システム
102ソフトウェア・セル
セルオブジェクト
(プログラム+データ)
ネットワーク
サーバー側コンピュータ
PDA
デジタルテレビ
(DTV)
クライアント側
コンピュータ
Visualizer
Cell

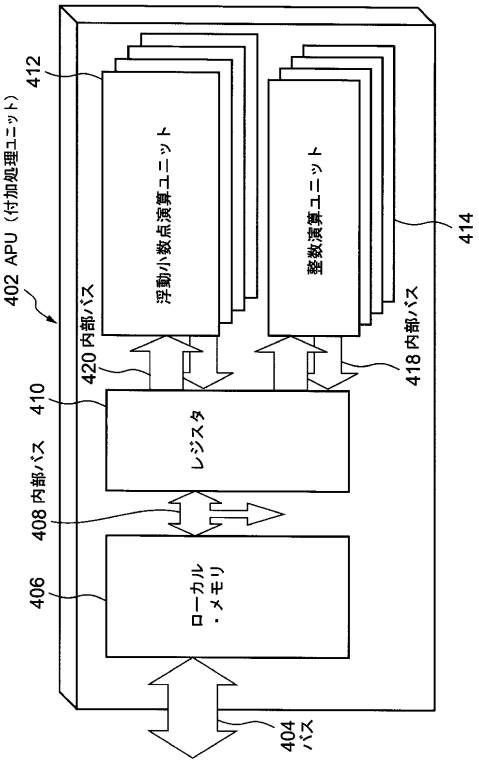
【図２】


プロセッサ・エレメント(PE)
201 PE(プロセッサ・エレメント)
PU
(処理ユニット)
DMAC
APU
(付加処理ユニット)
227 広帯域メモリ接続部
DRAM
223 ローカルPEバス

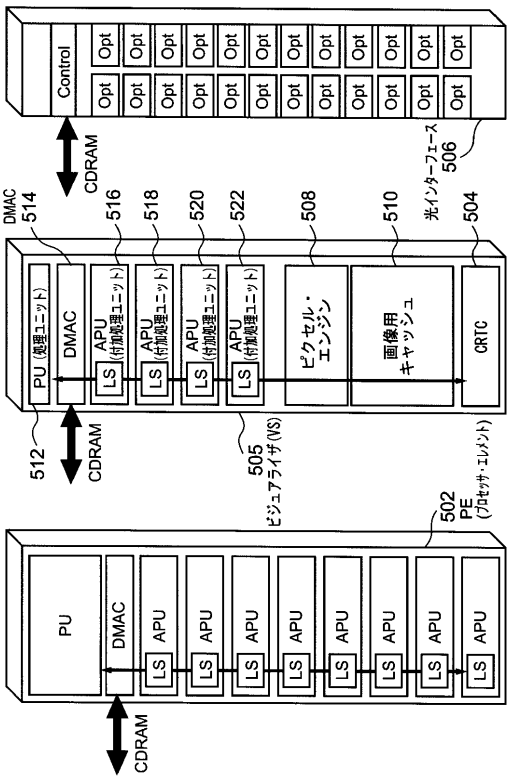
【図３】


共用DRAM
広帯域メモリ接続部
BE(広帯域エンジン)
PE(プロセッサ・エレメント)
PU
DMAC
APU
PEバス
BEバス
I/O
インターフェース
外部バス

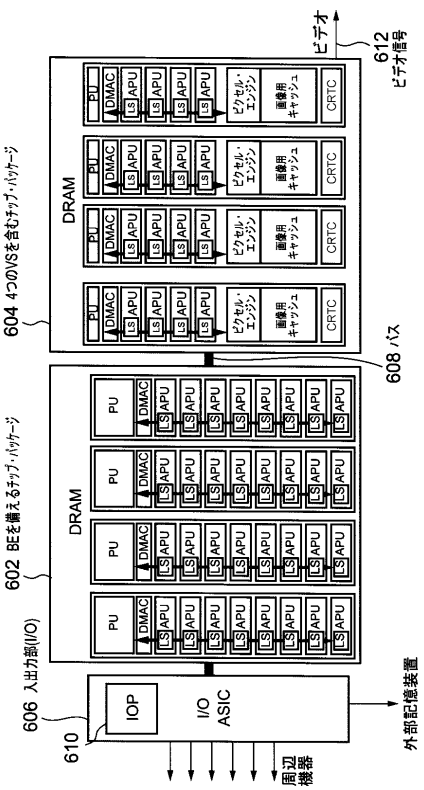
【図４】


APU(付加処理ユニット)
浮動小数点演算ユニット
整数演算ユニット
レジスタ
ローカル
・メモリ
内部バス
バス

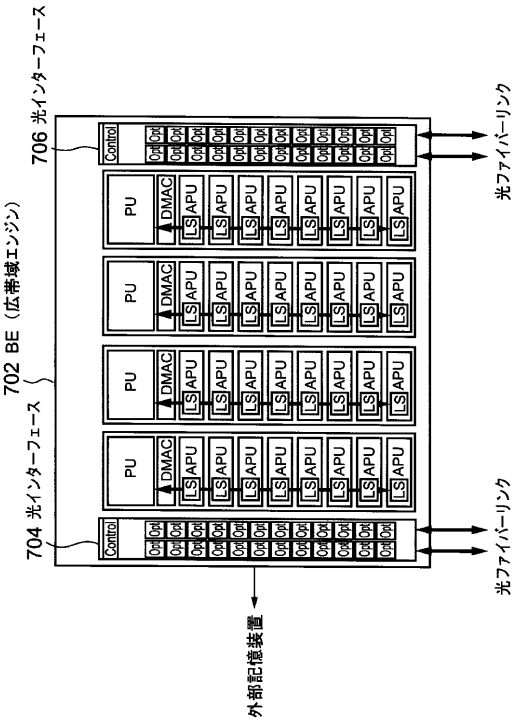
【図５】


Control
Opt
CDRAM
光インターフェース
506
DMAC
514
516
518
520
522
508
510
504
PU（処理ユニット）
DMAC
APU（付加処理ユニット）
LS
ピクセル・エンジン
画像用キャッシュ
CRTC
512
CDRAM
505
ビジュアライザ(VS)
502
PE
（プロセッサ・エレメント）
CDRAM

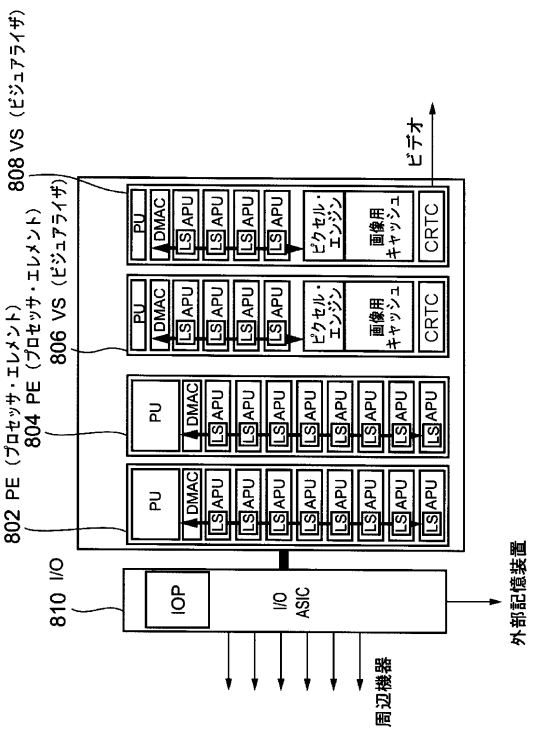
【図６】


ビデオ
612
ビデオ信号
604 4つのVSを含むチップ・パッケージ
DRAM
608 バス
602 BEを備えるチップ・パッケージ
606 入出力部(I/O)
610
IOP
I/O
ASIC
外部記憶装置
周辺機器

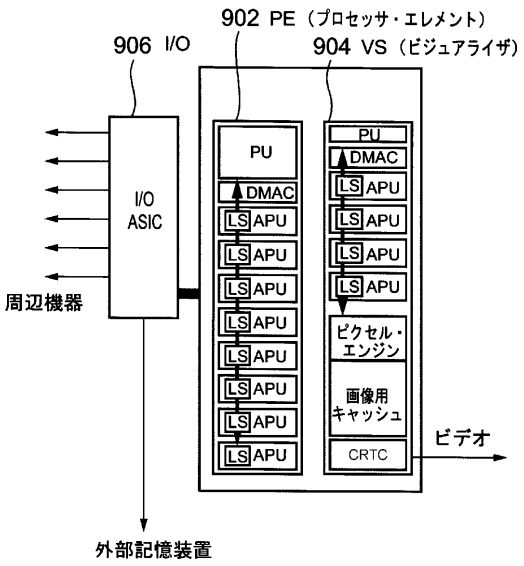
【図７】


706 光インターフェース
光ファイバーリンク
702 BE（広帯域エンジン）
704 光インターフェース
光ファイバーリンク
外部記憶装置

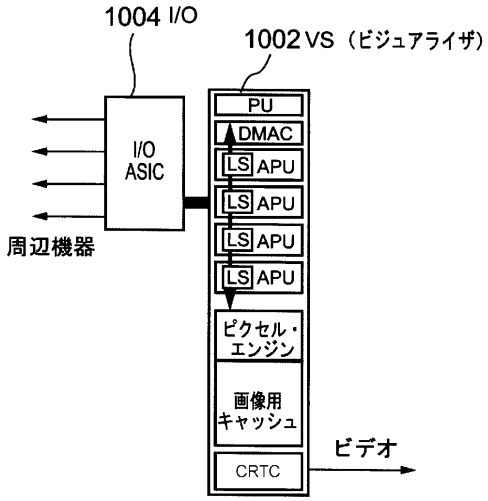
【図８】


808 VS（ビジュアライザ）
ビデオ
802 PE（プロセッサ・エレメント）
804 PE（プロセッサ・エレメント）
806 VS（ビジュアライザ）
810 I/O
IOP
I/O
ASIC
外部記憶装置
周辺機器

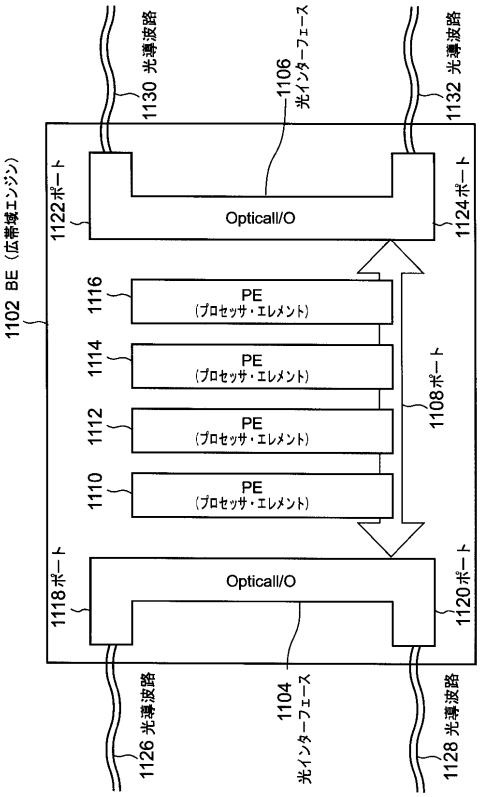
【図９】



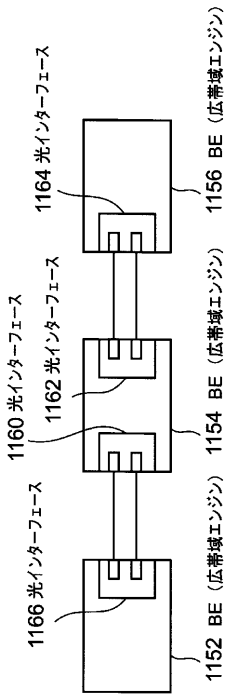
【図１０】



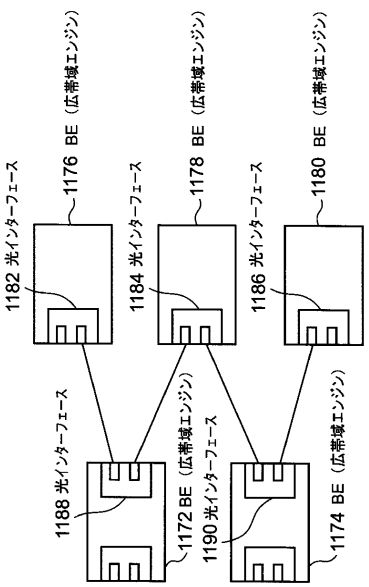
【図１１】



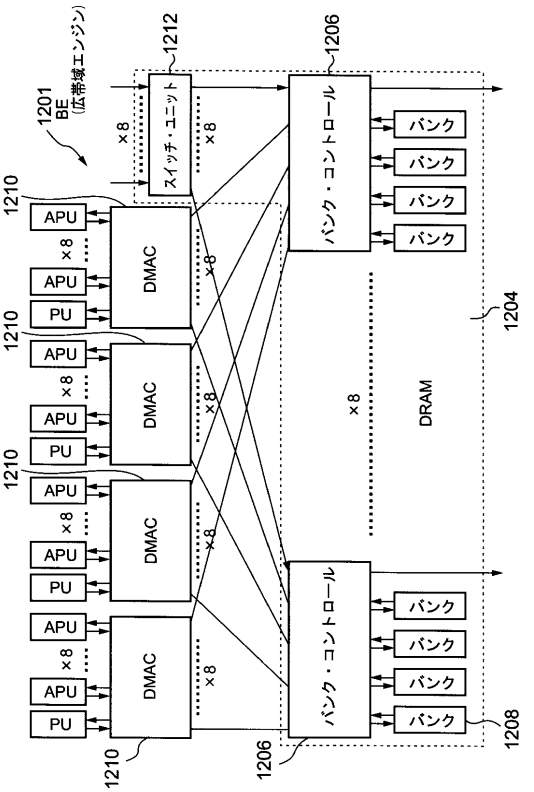
【図１２】



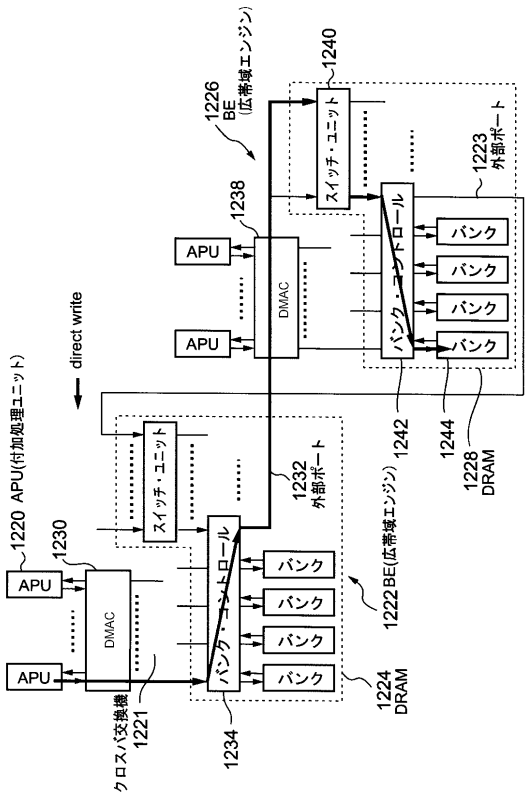
【図１３】



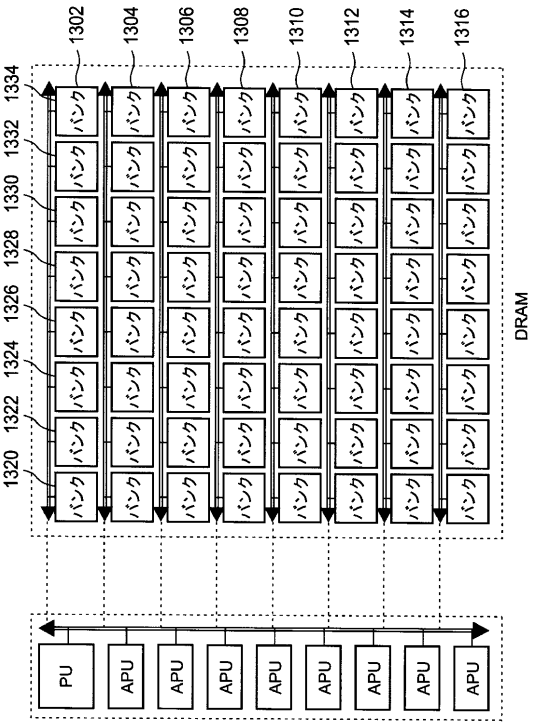
【図１４】



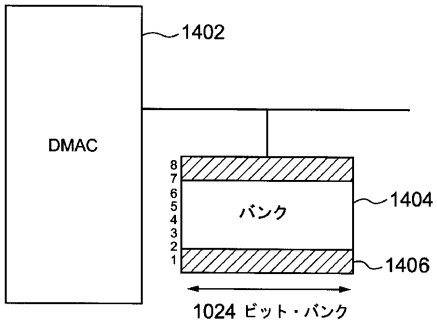
【図１５】



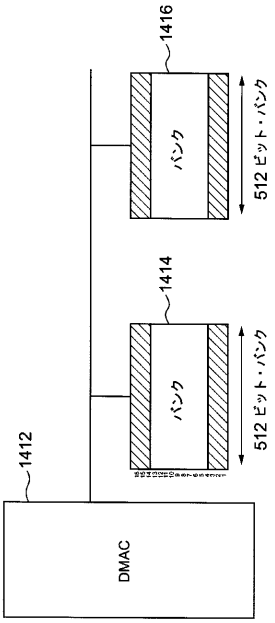
【図１６】



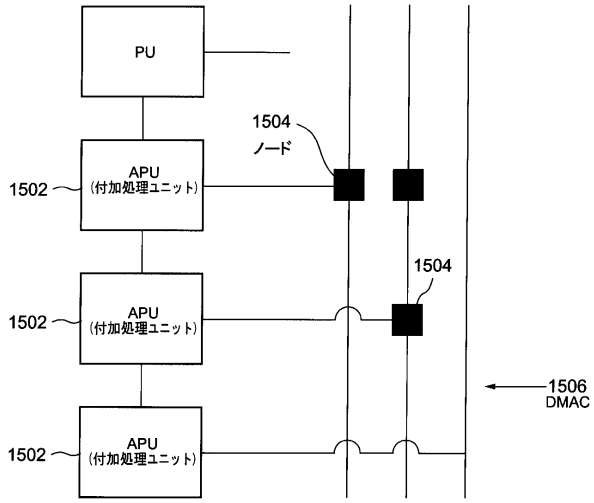
【図１７】



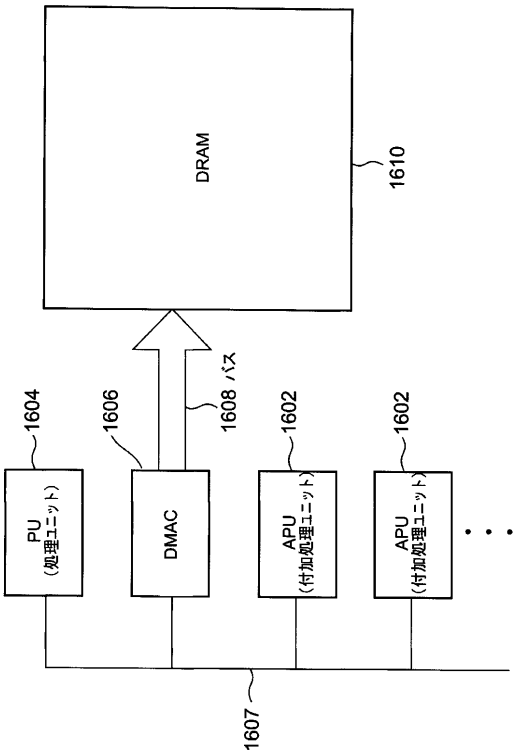
【図１８】



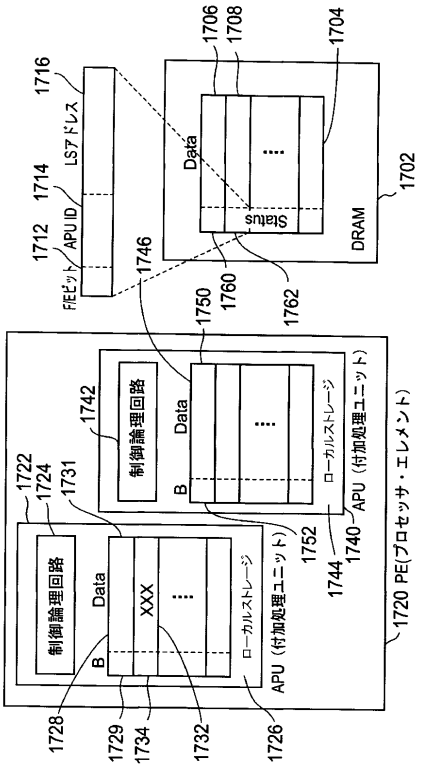
【図１９】



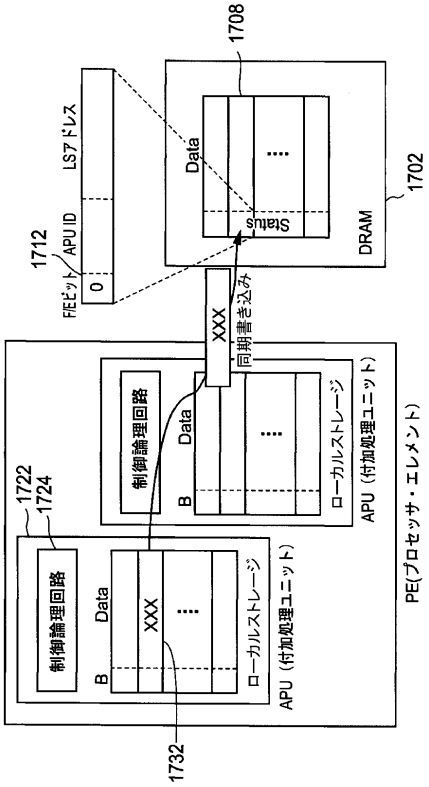
【図２０】



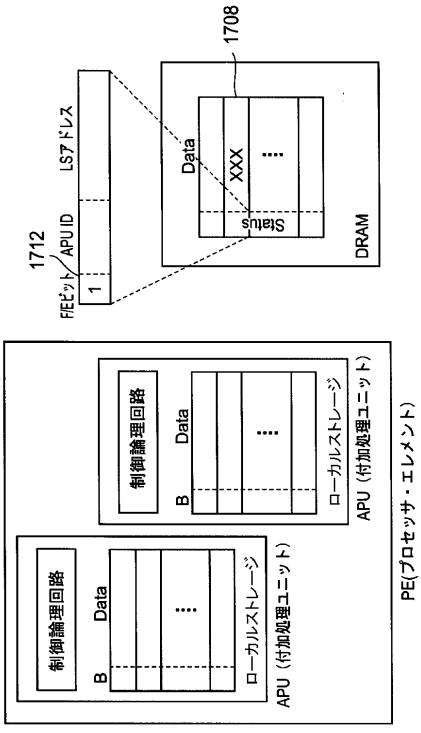
【図２１】



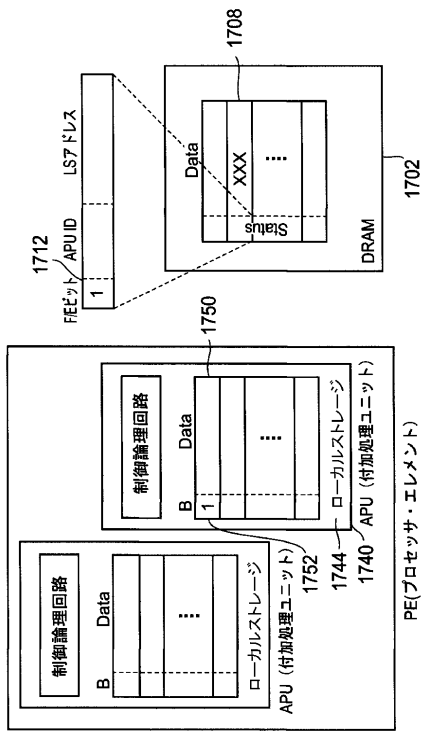
【図２２】



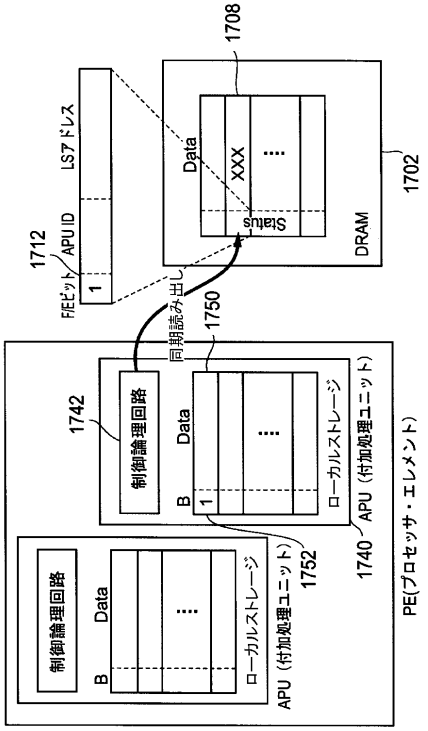
【図２３】



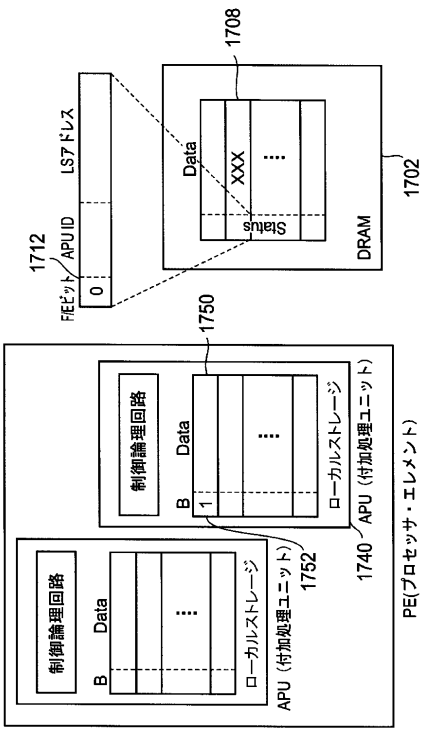
【図２４】



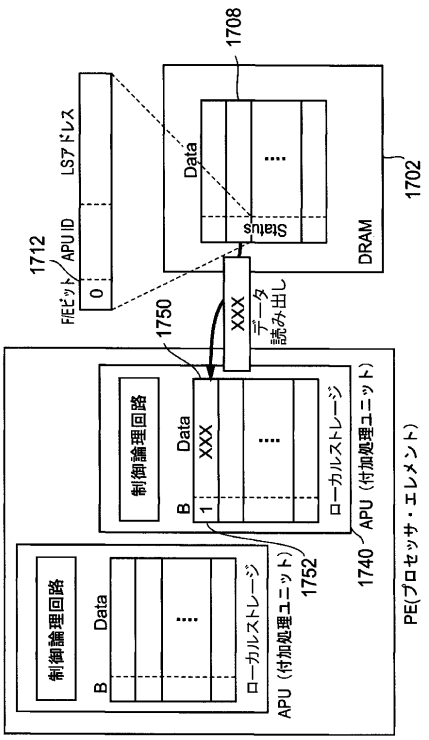
【図２５】



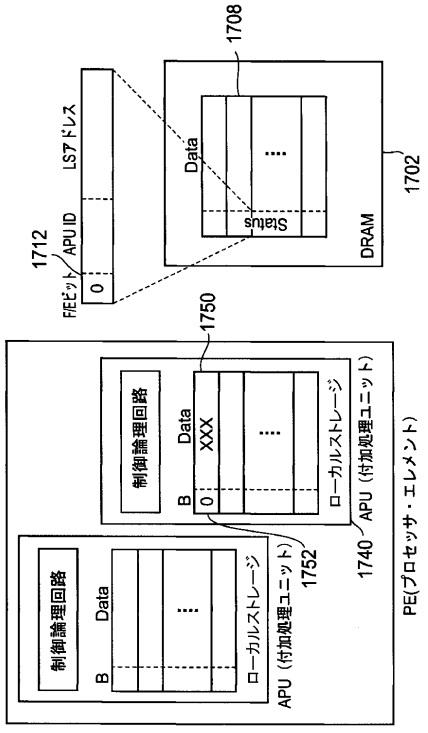
【図２６】



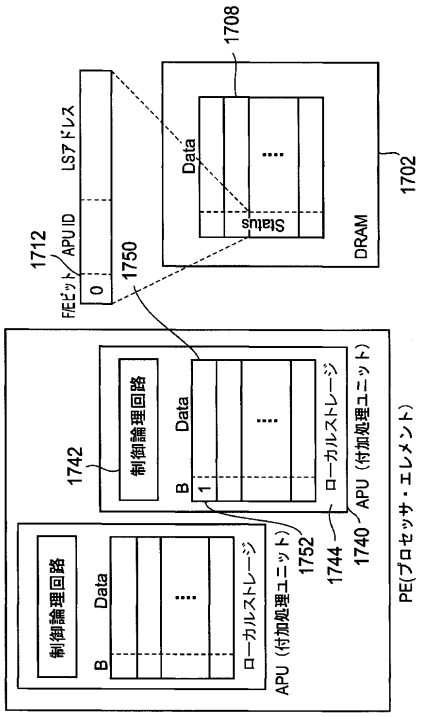
【図２７】



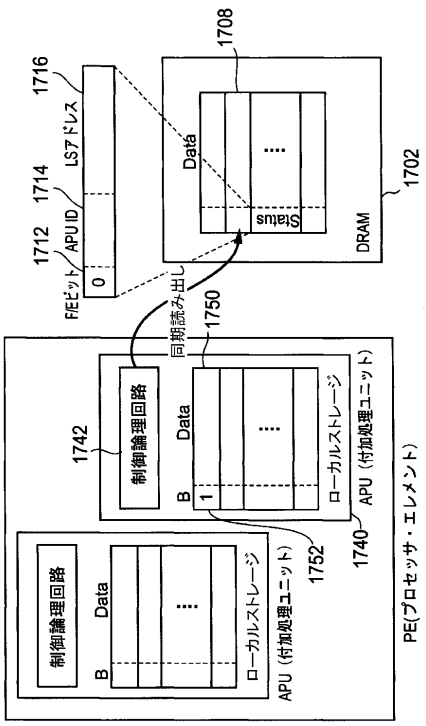
【図２８】



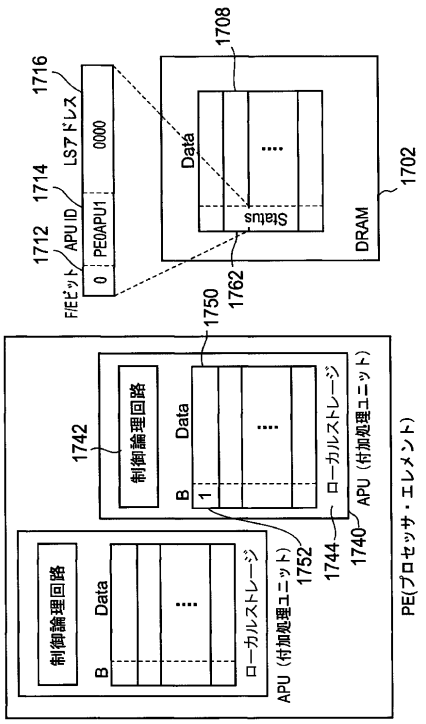
【図２９】



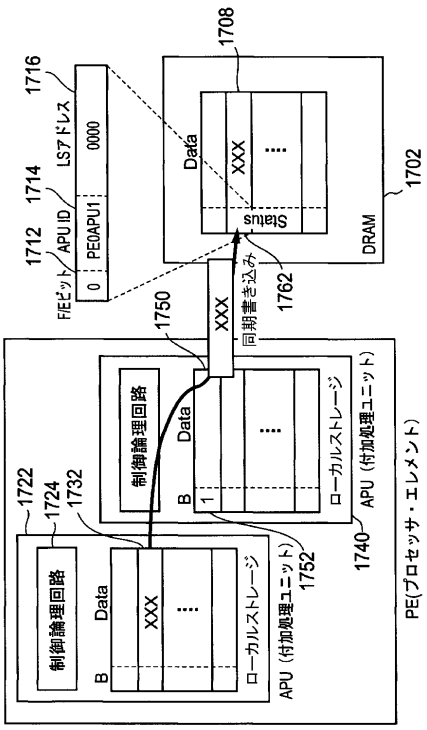
【図３０】



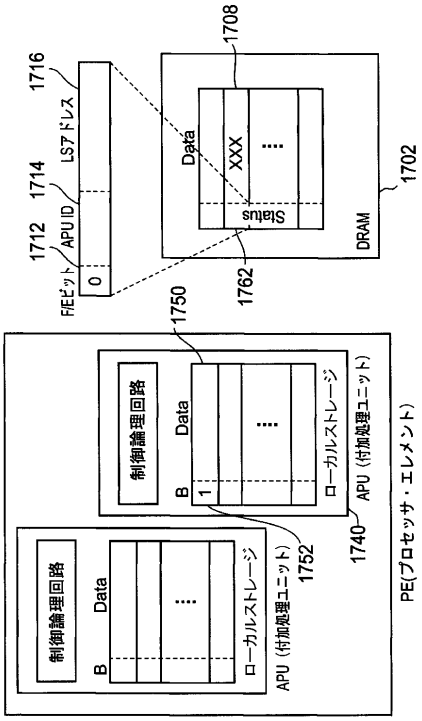
【図３１】



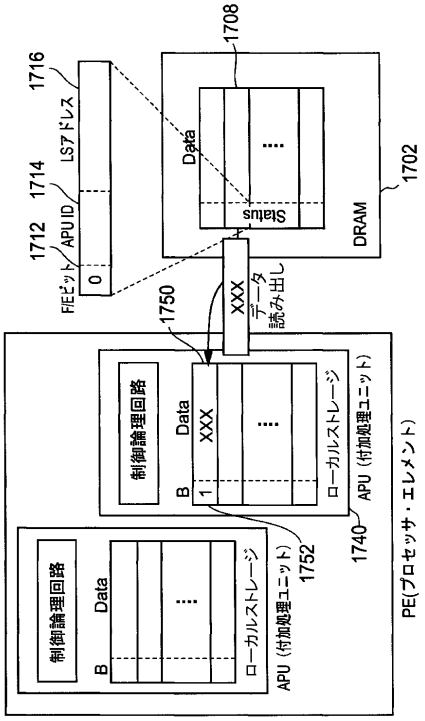
【図３２】



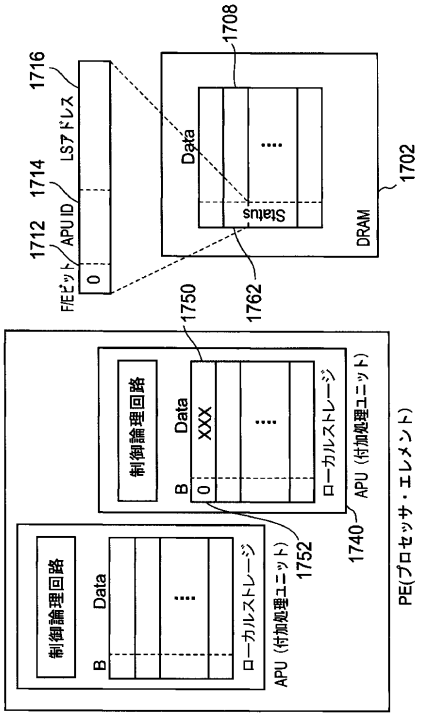
【図３３】



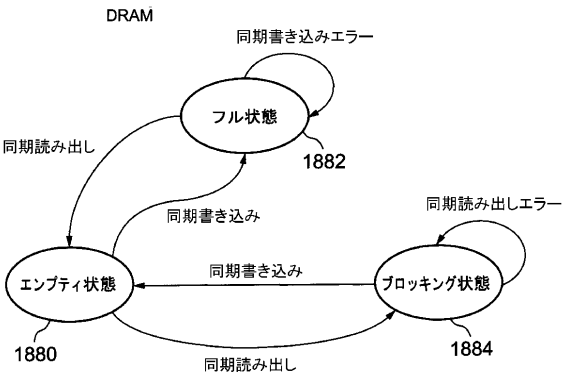
【図３４】



【図３５】



【図３６】



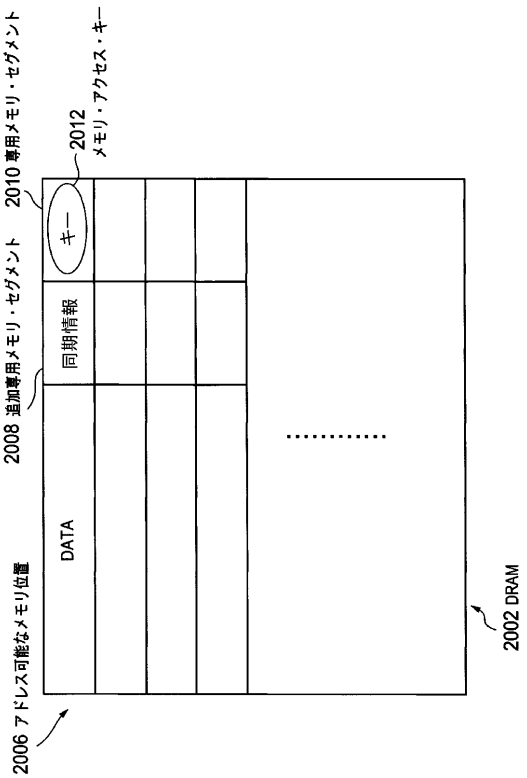
【図３７】

1902 キー管理テーブル

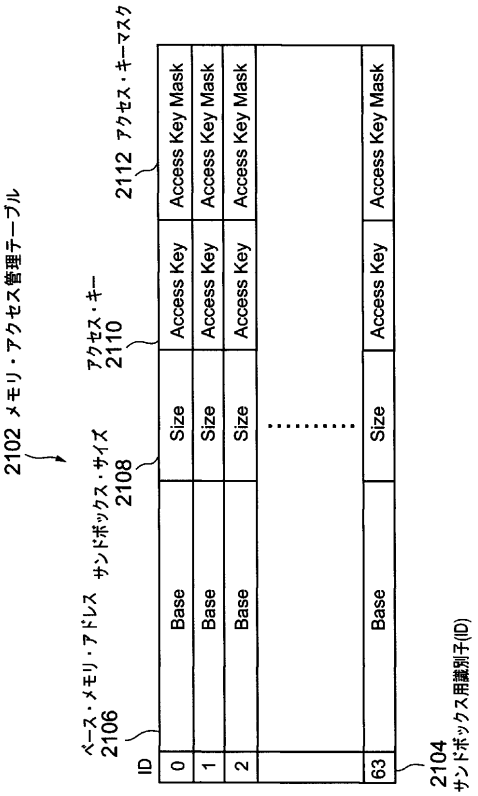
APU用の識別子(ID)

1904 ID	1906	1908
0	APUキー	キー・マスク
1	APUキー	キー・マスク
2	APUキー	キー・マスク
	.....	
7	APUキー	キー・マスク

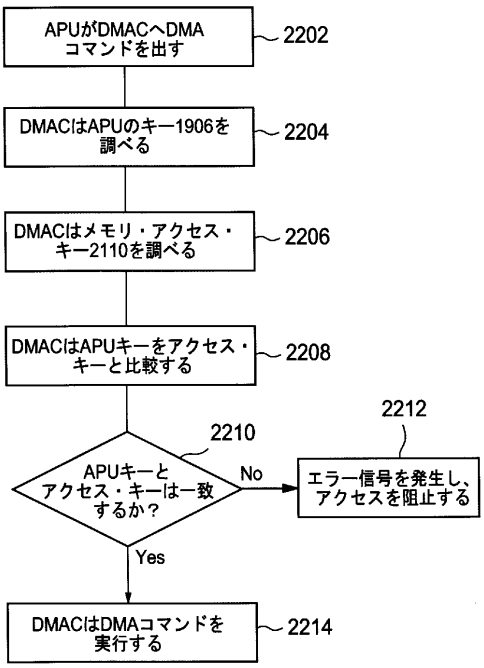
【図３８】



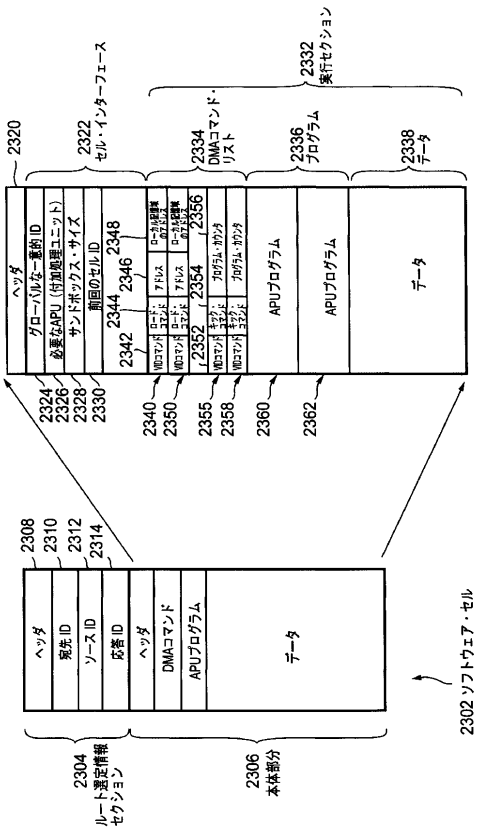
【図３９】



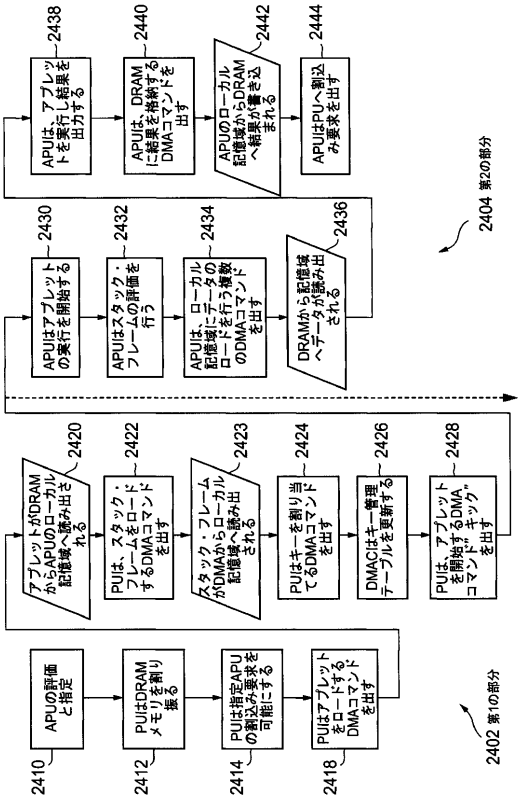
【図４０】



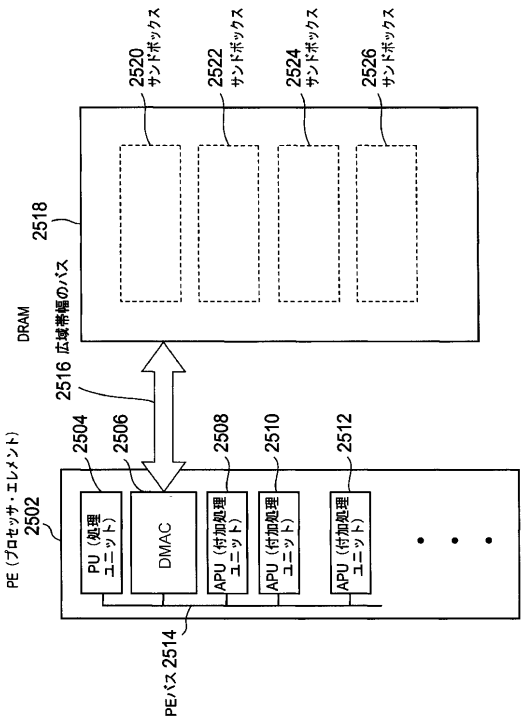
【図４１】



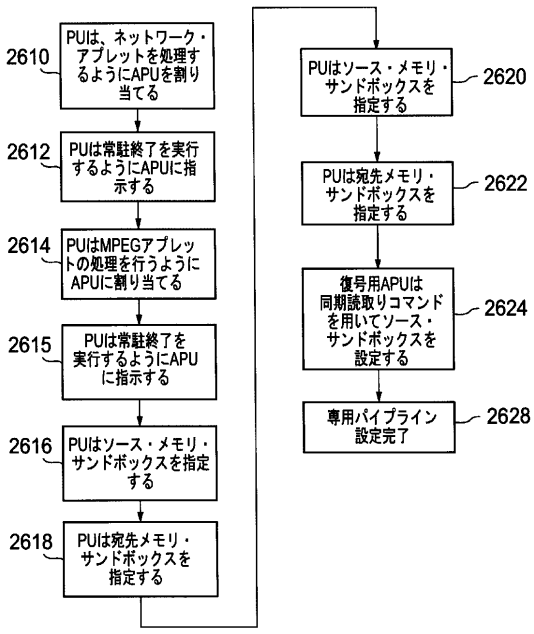
【図４２】



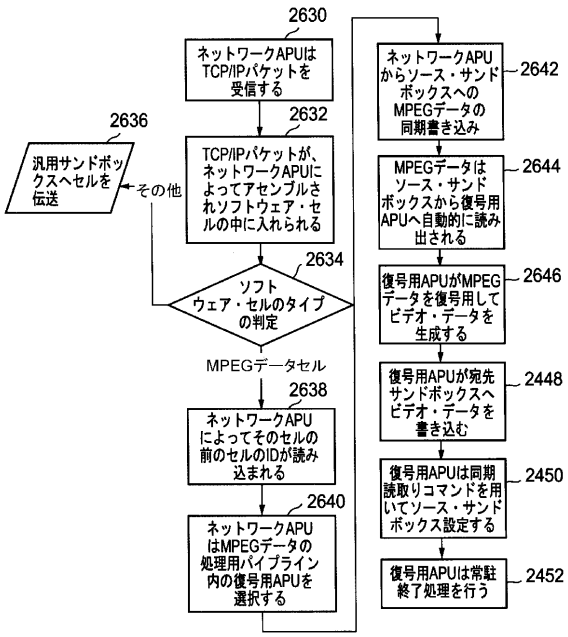
【図４３】



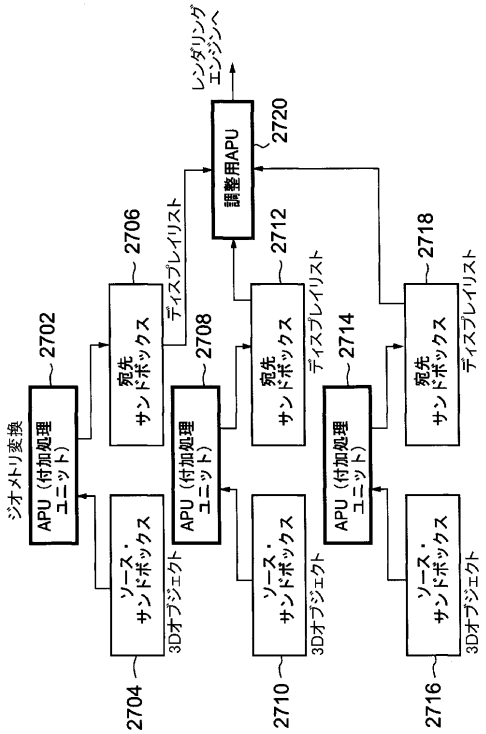
【図４４】



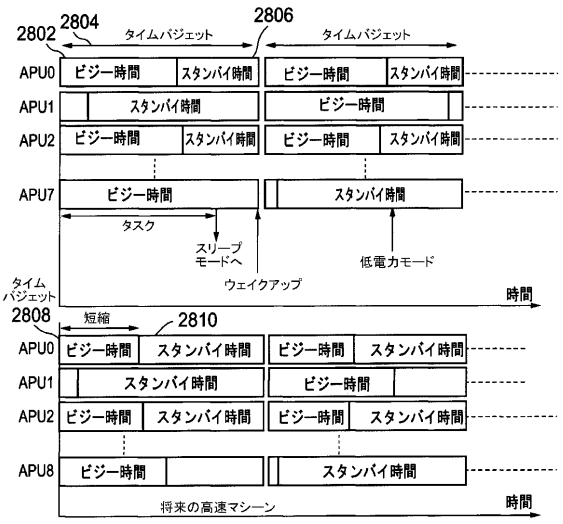
【図４５】



【図４６】


レンダリングエンジンへ
調整用APU
2720
2706
ディスプレイリスト
2712
2718
ディスプレイリスト
ディスプレイリスト
2702
ジオメトリ変換
APU（付加処理ユニット）
宛先サンドボックス
2708
宛先サンドボックス
2714
宛先サンドボックス
APU（付加処理ユニット）
APU（付加処理ユニット）
ソース・サンドボックス
3Dオブジェクト
ソース・サンドボックス
3Dオブジェクト
ソース・サンドボックス
3Dオブジェクト
2704
2710
2716

【図４７】


2802
2804
タイムバジェット
2806
タイムバジェット
APU0
ビジー時間
スタンバイ時間
ビジー時間
スタンバイ時間
APU1
スタンバイ時間
ビジー時間
APU2
ビジー時間
スタンバイ時間
ビジー時間
スタンバイ時間
APU7
ビジー時間
スタンバイ時間
タスク
スリープモードへ
ウェイクアップ
低電力モード
タイムバジェット
時間
2808
短縮
2810
APU0
ビジー時間
スタンバイ時間
ビジー時間
スタンバイ時間
APU1
スタンバイ時間
ビジー時間
APU2
ビジー時間
スタンバイ時間
ビジー時間
スタンバイ時間
APU8
ビジー時間
スタンバイ時間
将来の高速マシーン
時間

フロントページの続き

(72)発明者　山崎　剛
アメリカ合衆国、カリフォルニア州　９４４０４－２１７５、フォスター　シティー、セカンド　フロア、イースト　ヒルスデイル　ブルバード　９１９　ソニー　コンピュータエンタテインメント　アメリカ、インク．内

審査官　鈴木　修治

(56)参考文献　特開２０００－１３２４１１（ＪＰ，Ａ）
特開平０２－２１１５７５（ＪＰ，Ａ）
特開２０００－２８４９７４（ＪＰ，Ａ）
特開平０７－３２５７２５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/46-9/54