(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4445708号
(P4445708)
(45)発行日　平成22年4月7日(2010.4.7)　(24)登録日　平成22年1月22日(2010.1.22)

(51)Int.Cl.　F I
*G06F 12/06 (2006.01)*　G06F 12/06 515M

請求項の数 3　(全 20 頁)

(21)出願番号　特願2003-43570(P2003-43570)
(22)出願日　平成15年2月21日(2003.2.21)
(65)公開番号　特開2003-280984(P2003-280984A)
(43)公開日　平成15年10月3日(2003.10.3)
審査請求日　平成18年2月3日(2006.2.3)
(31)優先権主張番号　10/080,739
(32)優先日　平成14年2月22日(2002.2.22)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100081721
弁理士　岡田　次生
(74)代理人　100105393
弁理士　伏見　直哉
(74)代理人　100111969
弁理士　平野　ゆかり

最終頁に続く

(54)【発明の名称】マップテーブルを使用して入出力モジュールにアクセスする方法

(57)【特許請求の範囲】
【請求項１】
連続した論理アドレス空間を使用して、複数のモジュールにアクセスする方法であって、該複数のモジュールは複数のメモリおよび複数の入出力モジュールを含み、
前記複数のモジュールを複数のインタリーブエントリに分け、該複数のインタリーブエントリに対応する複数のエントリを含む少なくとも１つのマップテーブルを設けることであって、前記複数のエントリのそれぞれは、メモリタイプエントリか入出力タイプエントリかを示すエントリタイプ識別子を含むと共に、該インタリーブエントリに含まれるモジュールのうちの１つを特定するためのモジュール識別子を含むエントリ項目を複数含み、さらに、前記入出力タイプエントリのそれぞれは、インデックス選択識別子を含んでおり、該インデックス選択識別子は、予め決められた複数の値のうちの１つを取り、該複数の値は、異なるアドレスサイズに対応づけられている、ことと、
複数のアドレスビットからなる論理アドレスを受信することと、
前記複数のアドレスビットのうちの第１アドレス部分に基づいて、前記少なくとも１つのマップテーブルにおけるエントリを識別することと、
前記識別されたエントリの前記エントリタイプ識別子に基づいて、前記識別されたエントリのタイプを決定することと、
前記エントリタイプ識別子が入出力タイプエントリを示す場合、前記複数のアドレスビットのうちの第２アドレス部分に基づいて、前記識別されたエントリにおけるエントリ項目を識別することであって、該第２アドレス部分に使用するアドレスビットは、前記識別

されたエントリの前記インデックス選択識別子の値に基づいて、前記論理アドレスの前記複数のアドレスビットから特定される、ことと、

前記エントリタイプ識別子がメモリタイプエントリを示す場合、前記複数のアドレスビットのうちの第３アドレス部分に基づいて、前記識別されたエントリにおけるエントリ項目を識別することと、

前記識別されたエントリ項目の前記モジュール識別子によって識別されたモジュールにアクセスすることと、

を含む方法。

【請求項２】

連続した論理アドレス空間を複数のモジュールにマッピングする方法であって、前記複数のモジュールは複数のメモリおよび複数の入出力モジュールを含み、前記論理アドレス空間における各論理アドレスは第１、第２、および第３のアドレス部分を含み、

前記複数のモジュールを複数のインタリーブエントリに分け、該複数のインタリーブエントリに対応する複数のエントリを含むマップテーブルを設けることであって、前記複数のエントリのそれぞれは、メモリタイプエントリか入出力タイプエントリかを示すエントリタイプ識別子を含むと共に、該インタリーブエントリに含まれるモジュールのうちの１つを特定するためのモジュール識別子を含むエントリ項目を複数含み、前記入出力タイプエントリは、さらに、インデックス選択識別子を含んでおり、該インデックス選択識別子は、予め決められた複数の値のうちの１つを取り、該複数の値は、異なるアドレスサイズに対応づけられている、ことと、

一組の前記論理アドレスを、対応する前記エントリに関連付けることであって、或るエントリに関連づけられる該一組の論理アドレスは、前記第１のアドレス部分において共通する値を含み、前記第１のアドレス部分についての前記共通の値は、エントリ間で異なっている、ことと、

前記関連づけられたエントリの前記エントリタイプ識別子が前記入出力タイプエントリを示す場合、前記一組の論理アドレスの前記第２のアドレス部分に基づいて、該エントリの各エントリ項目を特定することであって、該第２のアドレス部分に使用されるアドレスビットは、該エントリの前記インデックス選択識別子の値に基づいて、該論理アドレスを構成するアドレスビットから特定され、

前記関連づけられたエントリの前記エントリタイプ識別子が前記メモリタイプエントリを示す場合、前記一組の論理アドレスの前記第３のアドレス部分に基づいて、該エントリの各エントリ項目を特定することと、

を含む方法。

【請求項３】

連続した論理アドレス空間を使用して複数のモジュールへのアクセスを提供するシステムであって、前記複数のモジュールは複数のメモリおよび複数の入出力モジュールを含み、

前記複数のモジュールを複数のインタリーブエントリに分け、該複数のインタリーブエントリに対応する複数のエントリを含む少なくとも１つのマップテーブルであって、前記複数のエントリのそれぞれは、メモリタイプエントリか入出力タイプエントリかを示すエントリタイプ識別子を含むと共に、該インタリーブエントリに含まれるモジュールのうちの１つを特定するためのエントリ項目を複数含み、前記入出力タイプエントリは、さらに、インデックス選択識別子を含んでおり、該インデックス選択識別子は、予め決められた複数の値のうちの１つを取り、該複数の値は、異なるアドレスサイズに対応づけられている、少なくとも１つのマップテーブルと、

論理アドレスを受信するコントローラであって、受信した論理アドレスの第１の部分に基づいて、前記少なくとも１つのマップテーブルにおけるエントリを識別し、該識別されたエントリの前記エントリタイプ識別子が前記入出力エントリタイプを示す場合には、前記識別されたエントリの前記インデックス選択識別子の値に基づいて、前記論理アドレスの第２の部分に用いるアドレスビットを、該論理アドレスを構成するアドレスビットから

特定し、該第２の部分に基づいて、前記識別されたエントリにおけるエントリ項目を選択し、該識別されたエントリの前記エントリタイプ識別子がメモリエントリタイプを示す場合には、前記論理アドレスの第３の部分に基づいて、前記識別されたエントリにおけるエントリ項目を選択し、該選択されたエントリ項目によって識別されるモジュールに基づいてモジュール識別情報を出力するように構成されるコントローラと、
を備えるシステム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は概して、コンピュータシステムのセルマップに関する。本発明は特に、セルマップを使用して入出力モジュールを仮想化するとととともにメモリをインタリーブするシステムおよび方法に関する。
【０００２】
【従来の技術】
本願は、２０００年４月２９日付けで出願された「MEMORY ADDRESS INTERLEAVING AND OFFSET BITS FOR CELL INTERLEAVING OF MEMORY」と題する、下に示す特許文献１の米国特許出願に関連する。
【０００３】
歴史的に見て、主メモリは物理的に中央バス上に配置されてきた。この種のシステム内では、完全な物理アドレスからなるメモリ要求がメモリサブシステムに転送され、データが戻される。分散メモリシステムでは、主メモリは多くの異なるセルにわたって物理的に分散される。セルはいくつかのプロセッサ、入出力（Ｉ／Ｏ）装置、セルコントローラ、およびメモリからなり得る。
【０００４】
分散システムでは、メモリをインタリーブしてもしなくてもよい。従来技術によるメモリをインタリーブするシステムおよび方法は、たとえば、下に示す特許文献２のWilliams他に１９９６年６月２５日付けで発行された「METHODS AND APPARATUS FOR INTERLEAVING MEMORY TRANSACTIONS INTO AN ARBITRARY NUMBER OF BANKS」、および特許文献３のBrockmann他に１９９４年３月８日付けで発行された「FLEXIBLE N-WAY MEMORY INTERLEAVING」に説明および記載されており、これらは本発明の所有者に譲渡されている。メモリが複数の物理的なセルに分割される、または複数の物理的なセルにわたって分割される非インタリーブアクセス方法では、まず第１のセルのすべてのメモリに順次アクセスし、次いで第２のセル中の利用可能なすべてのメモリに順次アクセスし、以下同様を行うことにより、連続したメモリ空間が１つにまとめられたブロックとしてアドレス指定される。各セルが可能な限りの最大メモリ量で構成されていた場合、システムからそのメモリは１つの連続したメモリブロックのように見え、１つの連続したメモリブロックとしてアドレス指定される。しかし、すべてのセルが最大メモリ容量に構成されているわけではない場合、この非インタリーブ方式では、セル内のメモリブロックの損失に対応してメモリ空間内に穴が生じうる。また、非インタリーブメモリでは、命令およびデータの双方が順次使用される傾向があるため、特定のセルへの複数の順次アクセスが必要である。ローカルに格納される場合には恩恵を受けるが、別のセルにおけるリモートメモリに連続してもしくは頻繁にアクセスするプロセッサは、ローカルセルおよびリモートセルの双方、ならびに接続ネットワークにおける処理リソースおよび通信リソースを含む相当なオーバヘッドを消費する。かなり続くと、これらリソースが他の処理に利用できなくなり、システムパフォーマンスが劣化することになり得る。
【０００５】
代替として、分散メモリシステム内のメモリにインタリーブプロトコルを介してアクセスすることができる。いくつかのセルにわたりメモリをインタリーブすると、より均等にメモリにアクセスすることができる。たとえば、システムがバスシステムを介して共に接続された２個のセルを含み、各セルがメモリおよび４個の別個のプロセッサを含む場合、セ

ル１におけるメモリをセル２におけるメモリでインタリーブすることにより、システム中の８個すべてのプロセッサがより均等に各メモリロケーションにアクセスする。２個のセルにメモリをインタリーブすることはまた、メモリロケーションにアクセスする際の各プロセッサの待ち時間遅延が一貫するように保証する。２つのメモリロケーションにメモリをインタリーブすると、プロセッサがメモリにアクセスしようと試みるか、またはメモリから情報を検索しようと試みる場合のボトルネックの可能性も減少する。

【０００６】

インタリーブの一例として、システムに包含されるメモリが０、１、２、および３とラベルの付いた４個のセルに分散するものと仮定する。さらに、セル０およびセル１はそれぞれ８ギガバイト（ＧＢ）のメモリを含み、セル２およびセル３はそれぞれ４ＧＢのメモリを含むものと仮定する。したがって、システム全体としては２４ＧＢのメモリを含む。分散メモリは以下のようにインタリーブすることができる。４個のセルはそれぞれ少なくとも４ＧＢのメモリを含むため、最初のインタリーブエントリであるエントリ０は、セル０、１、２、および３それぞれからの４ＧＢのメモリ、総計で１６ＧＢのメモリを含む。ここで、セル２およびセル３で利用可能なすべてのメモリがインタリーブエントリ０に使用されてしまった。セル０およびセル１それぞれは４ＧＢの未使用メモリを含む。インタリーブエントリ１は、セル０からの４ＧＢのメモリを含むとともにセル１からの４ＧＢのメモリを含む。したがって、インタリーブエントリ１は、セル０から４ＧＢかつセル１から４ＧＢで、８ＧＢのメモリを含む。ここで、４個のセルにおける２４ＧＢのメモリが２個のインタリーブエントリに分割された。ここで４個のセルからの２４ＧＢのメモリは、以下のような１つの連続ブロックとして見ることができる。ＧＢ０～１５はセル０、１、２、および３の下半分に配置され、ＧＢ１６～２３はセル０～１の上部分に配置される。このインタリーブはキャッシュラインレベルで行われる。プロセッサからは、２４ＧＢの情報は１つの連続したブロックに見える。２４ＧＢの情報は、１つの連続したブロックとして見えるが、物理的には４個の異なるセルに分散されている。

【０００７】

大型コンピュータシステムは、セルマップを使用してプロセッサアクセスを分散させてホットスポットを回避している。セルマップは、セル／ノードベースのシステムについて宛先モジュールを見つけるために使用されている。セルマップを使用してメモリをセルにインタリーブし、より均等なアクセスパターンをメモリに提供している。セルマップエントリは、１、２、４、８、１６、３２、および６４ウェイインタリーブを提供するために使用されてきた。セルマップエントリのサイズが、メモリがインタリーブされるウェイ数を決める。

【０００８】

上述したいくつかの従来のシステムは、セルマップを使用して仮想化およびメモリのインタリーブを実施していた。メモリのインタリーブは概して細粒であり、隣接するキャッシュラインへの２つのアドレスが異なるモジュールに行く。このような細粒度アクセスは通常、入出力（Ｉ／Ｏ）仮想化には必要ない。

【０００９】

従来のシステムでは、メモリのマッピングおよびＩ／Ｏのマッピングに別個のリソースが使用されていた。システムのトポロジおよびアーキテクチャに応じて、またシステムの現在のニーズに応じて、システムが“フリップフロップ”して所望のメモリまたはＩ／Ｏオペレーションを提供していた。また、インタリーブドメモリのプログラミングモデルとＩ／Ｏモジュールをマッピングするプログラミングモデルは全く異なっている。

【００１０】

【特許文献１】

米国特許出願第０９／５６３，０１８号

【特許文献２】

米国特許第５，５３０，８３７号

【特許文献３】

米国特許第５，２９３，６０７号
【００１１】
【発明が解決しようとする課題】
単一のセルマップ構造を使用してメモリのインタリーブおよびＩ／Ｏモジュールの仮想化を提供することが望ましい。
【００１２】
【課題を解決するための手段】
本発明の一形態は、連続した論理アドレス空間を使用してインタリーブされた複数のメモリおよび複数の入出力モジュールにアクセスする方法を提供する。少なくとも１つのマップテーブルが設けられる。少なくとも１つのマップテーブルは複数のエントリを含む。各エントリは、エントリタイプ識別子および複数のエントリ項目を含む。各エントリ項目はモジュール識別子を含む。各エントリは、メモリタイプエントリおよび入出力タイプエントリのうちの一方である。最初の論理アドレスを受信する。最初の論理アドレスは複数のアドレスビットを含む。アドレスビットの第１の組に基づいて少なくとも１つのマップテーブルにおけるエントリが識別される。識別されたエントリのエントリタイプ識別子に基づいて、識別されたエントリのタイプが決定される。エントリタイプ識別子が入出力タイプエントリを示す場合には、アドレスビットの第２の組に基づいて、識別されたエントリのエントリ項目が識別される。エントリタイプ識別子がメモリタイプエントリを示す場合には、アドレスビットの第３の組に基づいて、識別されたエントリのエントリ項目が識別される。識別されたエントリ項目のモジュール識別子によって識別されるモジュールがアクセスされる。
【００１３】
【発明の実施の形態】
以下の好ましい実施形態の詳細な説明では、本発明の一部をなし、本発明を実施し得る特定の実施形態を例として示す添付図面を参照する。本発明の範囲から逸脱することなく他の実施形態を利用することができるとともに、構造的または論理的な変更を行い得ることを理解されたい。したがって以下の詳細な説明は、限定を意味するものと解釈すべきではなく、本発明の範囲は併記の特許請求の範囲によって定義される。
【００１４】
図１は、クロスバー１２５を介して接続された４個のセル１０５、１１０、１１５、および１２０を含む処理システムまたはノード１００を示すブロック図である。各セルは対応するメモリブロックを有する。すなわち、セル１０５にはメモリ１３０、セル１１０にはメモリ１３５、セル１１５にはメモリ１４０、またセル１２０にはメモリ１４５がある。各セルはまた、４個のプロセッサ（セル１０５の場合、１５０、１５５、１６０、および１６５と符号が付与される）と、入出力（Ｉ／Ｏ）モジュール（セル１０５の場合では１７０と符号が付与される）と、セルコントローラ（セル１０５では１７５と符号が付与される）とを備える。
【００１５】
４個のプロセッサ１５０～１６５それぞれおよびＩ／Ｏモジュール１７０は、メモリ１３０へのアクセスを要求する。セルコントローラ１７５は、Ｉ／Ｏモジュール１７０およびプロセッサ１５０～１６５の双方とメモリ１３０との間のインタフェースである。メモリアクセスデバイス（たとえば、プロセッサ１５０～１６５のいずれか、またはＩ／Ｏモジュール１７０）がメモリの正確な部分にアクセスするために、セルコントローラ１７５が、メモリアクセスデバイスが知っている論理メモリアドレスを物理アドレスに変換する。物理アドレスは、セルコントローラ１７５にメモリ要求のルーティング方法がわかるようにする。セル１中の任意のメモリアクセスデバイスは、セル１１５のメモリ１４０、セル１２０のメモリ１４５、またはセル１１０のメモリ１３５にもアクセスすることができる。セルコントローラ１７５は、セルマップを使用して、メモリアクセスデバイスからの論理メモリアドレスを変換して、適切なメモリにアクセスするために使用することのできる適切な物理アドレスにする。一実施形態では、セルコントローラ１７５は、各メモリアク

セスデバイスごとに異なるセルマップを含む。図１に示す実施形態では、セルコントローラ１７５は、取り付けられたメモリアクセスデバイスごとに１つずつ、合わせて５つの異なるセルマップを含む。

【００１６】

セルマップの一実施形態は、複数の行および複数の列を有するテーブルであり、各行がインタリーブエントリに対応し、行中の各列がシステムにおけるセルの１つを識別するエントリ項目に対応する。

【００１７】

図２は、４個のセル：セル０、セル１、セル２、およびセル３にメモリが分散した分散メモリシステム２００を示す。セル０は総計で８ギガバイト（ＧＢ）のメモリを含み、セル１は６ＧＢのメモリを含み、セル２は４ＧＢのメモリを含み、セル３は２ＧＢのメモリを含む。一実施形態では、これら４個のセルへのインタリーブは、以下のようにして達成される。まず、インタリーブに利用可能なメモリ量が最小なのはどのセルかについて査定を行う。この場合、セル３は２ＧＢのメモリしか含まない。したがって、インタリーブエントリ０は、セル０、１、２、および３から２ＧＢのメモリ、総計で８ＧＢのメモリをインタリーブする。その結果得られるインタリーブエントリを図３の行３０５に示す。図３中、各行は最大８個のセルメモリを識別する。行３０５の上には、３ビットの８つの組み合わせがあり、３ビットの各組み合わせが行３０５（および行３１０）における８つのエントリ項目の１つに関連付けられる。３ビットのこれら組み合わせを使用して特定のエントリ項目を識別するが、これについてさらに詳細に後述する。

【００１８】

次に、査定を行って、利用可能な任意のセルに残っている最小のメモリ量を決定する。この場合、セル２における２ＧＢである。セル０、セル１、およびセル２にわたるインタリーブは、図４に示す３つのインタリーブエントリを使用することによって提供される。インタリーブエントリ１、２、および３（図４中の３ビット組み合わせの下にある最初の３行それぞれ）は、セル０の二番目の２ＧＢブロック、セル１の二番目の２ＧＢブロック、およびセル２の一番上の２ＧＢ部分に使用される。この時点において、インタリーブエントリ０は、総計８ＧＢの情報をアドレス指定するが、インタリーブエントリ１、２、および３はそれぞれ２ＧＢの情報を含む。セル３のメモリリソースは、インタリーブエントリ０で完全に使用されてしまった。セル２のメモリリソースは、インタリーブエントリ３の完成を通して使い尽くされた。

【００１９】

本プロセスにおける次のステップは、割り当てられずに残っている任意のセルのメモリを識別することである。この場合、セル１に２ＧＢのメモリが残っている。インタリーブエントリ４は、通常であれば、セル０からの２ＧＢのメモリおよびセル１からの２ＧＢのメモリを含むことになろう。この配列により、すでにマッピングされた１４ＧＢに４ＧＢが追加され、総計で１８ＧＢになる。しかし、一実施形態では、各テーブルエントリがグループサイズの整数倍で、すなわち１６ＧＢで始まることが好ましい。したがって、セル０およびセル１における４ＧＢが１つのインタリーブエントリを占めることができるようにするには、次に２ＧＢを追加する必要がある。このため、インタリーブエントリ４（図示せず）は、セル０にある一番上の２ＧＢを表し、マッピングする。ここで、今までに定義されたインタリーブエントリ（０、１、２、３、および４）は１６ＧＢのメモリを含むことになる。インタリーブエントリ５（図示せず）は、セル１にある残りの２ＧＢおよびセル０における残りの２ＧＢをマッピングする。これは、これまでにエントリ０、１、２、３、および４に割り当てられた１６ＧＢが４ＧＢの倍数であるため差し支えない。要約すると、セル０～セル３に含まれる２０ＧＢのメモリはここで、６つのインタリーブエントリに含められる。最初のインタリーブエントリは、４個のセルそれぞれから２ＧＢずつ、合わせて８ＧＢの情報を含む。インタリーブエントリ１、２、３、および４は２ＧＢを含み、インタリーブエントリ５は４ＧＢを有する。

【００２０】

インタリーブグループは、所与のメモリ範囲でのインタリーブに制限される同量のメモリを有する複数のセルのユニットとして定義される。インタリーブグループは１つまたは複数のインタリーブエントリで構成される。インタリーブグループ０は、インタリーブエントリ０内にある８ＧＢ、換言すれば４個のセルそれぞれからの２ＧＢで構成される。インタリーブグループ１は、インタリーブエントリ１、２、および３内にある６ＧＢ、換言すればセル０からの２ＧＢ、セル１からの２ＧＢ、およびセル２からの２ＧＢで構成される。インタリーブグループ２は、セル０の一番上の２ＧＢ（すなわち、インタリーブエントリ４）で構成される。インタリーブグループ３は、セル０およびセル１それぞれからの２ＧＢで構成される（すなわち、インタリーブエントリ５）。

【００２１】

所望の物理メモリが配置されている特定のセルを識別するには、入力メモリアドレスからの第１のアドレスビットの組を使用してセルマップの行を識別し、入力メモリアドレスからの第２のアドレスビットの組を使用して識別された行にインデックスを付け、エントリ項目を識別する。一実施形態では、４４ビットメモリアドレスが使用され、アドレスビット２９～４３が第１のアドレスビットの組に対応し、ビット６～８が第２のアドレスビットの組に対応する。３ビット（たとえば、ビット６～８）を使用して、８列すなわち８個のエントリ項目を有するエントリにインデックスを付ける。１６エントリ項目を有するエントリの場合には、４ビット（たとえば、ビット６～９）をインデックス付けに使用する。同様に、エントリサイズが倍になるごとに、追加のインデックス付けビットを１個使用する。

【００２２】

インタリーブは、アクセスされた連続メモリ量が少なく保たれる場合に最も効果的である。キャッシュラインインタリーブでは、セルマップはキャッシュラインが配置されるセルを示す。このため、インタリーブエントリ０（図３における行３０５）を再び参照すると、キャッシュラインがセル０、１、２、および３の間でインタリーブされた場合、行３０５はどのようにセルがテーブル内で表されるかを示す。一実施形態では、行３０５の上にある複数の３ビット組み合わせでの３ビットは、入力論理メモリアドレスのアドレスビット６～８に対応する。入力アドレスのこれら３ビットを使用して、行にインデックスを付け、エントリ項目の１つを識別する。行３０５における各エントリ項目は、４個のセルの１つを識別する。図３に示すように、メモリアドレスのビット６～８が０００である場合、これは、メモリの物理アドレスがセル０にあることをセルコントローラに示す。メモリアドレスのビット６～８の値が００１である場合、これはメモリの物理アドレスがセル１にあることをセルコントローラに示す。同様に、セルコントローラは、メモリアドレスのビット６～８について可能な他の６つの値それぞれを使用して、テーブルからのセルを識別する。

【００２３】

再び図１を参照して、４個のセルを有する単一ノードを示す。各セルはそれぞれのメモリデバイスを有する。しかし、システム全体が２個の４セルノードを有する場合には、それぞれメモリを有する８個のセルが利用可能である。これらセルにセル０～セル７とラベルを付ける場合、メモリを８個すべてのセルにわたってインタリーブすることができるであろう。図３の行３１０は、どのようにメモリを８個すべてのセルにインタリーブすることができるかを示す。この場合、３ビット指示子（メモリアドレスのビット６～８）は、８個のセルのうちいずれがメモリの物理アドレスを含むかを示す。したがって、行３１０は８ウェイインタリーブを示し、行３０５は４ウェイインタリーブを示す。

【００２４】

システムが３セル構成を有する場合、セルマップは、３ウェイインタリーブを示す図４に示すようになる。３個のセルは、セルマップにおいて３行にわたって効率的にマッピングされる。１行目の１列目は、セル０の値を含む。１行目の２列目は、セル１の値を含む。１行目の３列目は、セル２の値を含む。このシーケンスは、最後のセルが３行目の最後の列で終わるまで繰り返される。

【００２５】
セルマップの行は、マスク、コンパレータ、およびメモリアクセスデバイスからのメモリアドレスビットの組み合わせを介してセルコントローラによって識別される。マスクを使用して、適切な行の決定に関係のないアドレスのビットをマスクする。コンパレータは、マスクしたアドレス部分を整合値と比較して、セルマップの対応する行を識別する。
【００２６】
図５は、シングルキャッシュラインインタリーブの場合の６４ウェイインタリーブドセルマップエントリを示すブロック図である。図５におけるセルマップエントリは６４ウェイインタリーブであるため、メモリアドレスの６ビット（たとえば、ビット６～１１）をエントリへのインデックス付けに使用する。エントリイネーブルブロック５１０は、入力メモリアドレスのビット２９～ビット４３を含むアドレス部分５０５を使用してセルマップの適切な行を識別する。アドレス範囲は、行選択に使用するアドレス部分５０５から決定される。図５に示すように、エントリイネーブルブロック５１０は、以下の式Iを実行する。
【００２７】
【数１】
Ｅｎｔｒｙ＿ｅｎａｂｌｅ＝（（ＡＤＤＲ　ＡＮＤ　ＭＡＳＫ）＝＝ｃｍｐ）式I
【００２８】
式Iに示すように、エントリイネーブルブロック５１０は、アドレス部分５０５をマスクしてアドレスの最下位ビットを破棄し、マスクされたアドレスをその行の比較値または整合値である「ｃｍｐ」と比較する。比較における値が等しい場合、Ｅｎｔｒｙ＿ｅｎａｂｌｅに、マッチする行を示す論理的に真の値が割り当てられる。値が等しくない場合、Ｅｎｔｒｙ＿ｅｎａｂｌｅに、マッチしない行であることを示す偽の論理値が割り当てられる。
【００２９】
行が選択されると、メモリアドレスのビット６～１１を使用して、セルマップのその行内の６４個のエントリ項目の１つを識別し、「ＣＥＬＬ」および「Ｃｅｌｌ＿Ａｄｄｒ」が決定され、出力される。「ＣＥＬＬ」は、セルを一意に識別するセルＩＤを表し、「Ｃｅｌｌ＿Ａｄｄｒ」はセルアドレスを表し、これについてさらに詳細に後述する。図５に示すように、アドレス部分５１５は、アドレスビット６～１１および２９～４２を含み、アドレスビット６～１１および２９～４２を使用して適切なセルＩＤおよびセルアドレスを決定する。アドレス入力５２０に示すように、さらなる入力によってさらなるテーブルエントリ項目を選択することができる。図５に示すように、以下の式IIに示すような比較を行うことによって列が選択される。
【００３０】
【数２】
Ｃｏｌｕｍｎ＿Ｓｅｌｅｃｔ＝（ＡＤＤＲ[１１：６]＝＝０ｘ０）式II
【００３１】
式IIからわかるように、メモリアドレスのビット６～１１を、列に対応する１６進数の値（たとえば、この場合では０ｘ０であり、これは一列目に対応する）と比較する。比較している値が等しい場合、Ｃｏｌｕｍｎ＿Ｓｅｌｅｃｔは、マッチングを示す真の論理値を含む。比較している値が等しくない場合、Ｃｏｌｕｍｎ＿Ｓｅｌｅｃｔは、マッチングがないことを示す偽の論理値を含む。
【００３２】
エントリイネーブルブロック５１０でのコンパレータおよびマスクの使用について、図６を参照してさらに詳細に述べる。図６は、十進数の０～１６を表す二進数を示す。０～１６の範囲の１０進数は４つの異なるグループ：０～３を含むグループ６０５、４～７を含むグループ６１０、８～１１を含むグループ６１５、および１２～１５を含むグループ６２０に分けられている。マスクを効果的に使用するために、種々のグループを識別する方法が必要である。この場合、最初の２ビット、すなわち最上位２ビット（最左端の２ビッ

ト）を使用して各種グループを区別することができることがわかる。グループ６０５に表示される４つの数はそれぞれ００で始まり、グループ６１０の中の数はそれぞれ０１で始まり、グループ６１５の中の数はそれぞれ１０で始まり、グループ６２０の中の数はそれぞれ１１で始まる。したがって、１１００からなるマスクが確立される。重要な各ビットには「１」を配置し、重要ではない、すなわち「無関係な」各ビットには「０」を配置する。コンパレータは、メモリブロックの開始アドレスに等しくセットされる。マスクを使用して、４つのグループ６０５～６２０のいずれに所望のビットシーケンスが存在するかを判定する。２つの最上位ビットの比較により、セルマップ内の特定の行が決定される。

【００３３】

コンパレータおよびマスクの使用をさらに説明するため、いくつかの例を考察する。図７を参照すると、ノードは４個のセルを含み、セル０は５ＧＢのメモリを有し、セル１は３ＧＢ、セル２も３ＧＢ、またセル３も３ＧＢのメモリを有する。１６進数表記では、１ＧＢは０Ｘ０００＿４０００００００である。図７のインタリーブエントリ０は、４個のセルそれぞれから２ＧＢ、総計８ＧＢのメモリを含む。このメモリブロックの範囲は０ＧＢから最大８ＧＢである。コンパレータは、０Ｘ０００＿０００００００に等しい。マスクの値は、メモリロケーションが８ＧＢよりも大きいかどうかを判定するために調べる必要のあるビットを識別することによって決定される。値が８ＧＢを越えるかどうかを判定するために調べなければならない各ビット位置に、「１」を配置する。このため、１６進数表記では、マスクは０Ｘｆｆｅ＿０００００００に等しい。１６進数表記での最下位の８つの数で表されるビットは、メモリロケーションの値が８ＧＢを越えているかどうかの判定に必要ないため、調べる必要はない。したがってマスクは、メモリアクセスデバイスの１つからのメモリ値が８ＧＢを越えるか否かを判定するために調べる必要のあるビット位置のみを含む。確立されたこのマスクおよびコンパレータを使用すると、０～８ＧＢの範囲にあるいずれのアクセスもインタリーブエントリ０しか起動しない。対応するセルマップの適切な行内において、セル番号は０、１、２、３、０、１、２、３として識別され得る。これは、図３における行３０５と同様である。この構成は、セル０～３の間での４ウェイインタリーブを達成する。インタリーブエントリ１の場合、ここでも４個すべてのセルが使用されるが、今度は各セルから１ＧＢのメモリのみが使用される。この場合、コンパレータ値は、８ＧＢに等しい０Ｘ００２＿０００００００に等しい。これは、このインタリーブエントリの開始値である。この場合のマスク値は、０ＸｆｆｆＸ０００００００００である。インタリーブエントリ１が確立されると、セル０に２ＧＢが残る。

【００３４】

概して、インタリーブは、まず最大のブロックにわたって行われるため、続いて次のインタリーブブロックのサイズはより小さい。セル０に残っている２ＧＢは、通常であれば、図７に示す１ＧＢチャンクに分けられる。しかし、この場合、インタリーブエントリ０およびインタリーブエントリ１に含まれる１２ＧＢは、セル０に残っている２ＧＢで割り切れる。したがって、インタリーブエントリ２は、セル０に残っている２ＧＢのメモリを含むことになり、コンパレータ値は１２ＧＢである０Ｘ００３＿０００００００になり、マスク値は０ＸｆｆｆＸ８００００００になる。

【００３５】

最後の例として、図８に示すように、ノードが３個のセルを含み、各セルが２ＧＢのメモリを含む場合、インタリーブは以下のように行われる。インタリーブエントリ０の場合、コンパレータは０Ｘ０００＿０００００００、すなわち０ＧＢである。マスクは０ＸｆｆＸ８００００００である。これにより、セル０の最初のＧＢおよびセル１の最初のＧＢの間でインタリーブすることができるようになる。この場合の行エントリは、０の後に１が続き、行の全長にわたりこれが交互になったものからなる。これは、ここでもセル０から１ＧＢおよびセル１から１ＧＢというメモリの最初の２ＧＢにわたり２ウェイでインタリーブする。

【００３６】

インタリーブエントリ１の場合、コンパレータ値は０Ｘ０００＿８００００００であり

、これはすなわち２ＧＢに等しい。この場合のマスク値は、０Ｘｆｆｆ＿８０００００００である。対応するセルマップ行は、行の全長にわたって１と２が交互になったものであり、これにより、セル１からの２番目のＧＢおよびセル２からの最初のＧＢにわたり２ウェイインタリーブすることができる。

【００３７】

インタリーブエントリ２は、０Ｘ００１＿０００００００００のコンパレータ値を有し、これは４ＧＢに等しい。マスク値は、０Ｘｆｆｆ＿８０００００００である。このインタリーブエントリのセルマップ内の対応する行は、行の全長にわたって２と０が交互になったものである。このインタリーブエントリは、セル２およびセル０にわたり２ウェイでインタリーブし、最後の２ＧＢのメモリを含む。

【００３８】

一実施形態では、セルコントローラは、セルＩＤだけの場合よりも、より多くの情報をセルマップエントリ項目から取得する。たとえば、セルコントローラをセル０に送る（direct）いくつかのセルマップエントリ項目がありうる。セル０に含まれるメモリは、様々なインタリーブエントリに分けられている場合がある。一実施形態では、セルマップエントリ項目は、セル０のメモリ内のどこに要求される情報が格納されているかの指示もセルコントローラに供給する。本発明の一形態では、ＣＥＬＬ＿ＡＤＤＲすなわちセルアドレスを使用して、セルコントローラを特定のメモリアクセスのためにセルメモリ内の特定の５１２メガバイト（ＭＢ）領域に送る。セルアドレス式の一実施形態を以下の式IIIで提供する。

【００３９】

【数３】

ＣＥＬＬ＿ＡＤＤＲ＝（（ＡＤＤＲ[４２：２９]＆ＣＨＵＮＫ＿ＭＡＳＫ）＞＞インタリーブ＋ＣＨＵＮＫ＿ＳＴＡＲＴ式III

【００４０】

アドレスビット２９～４２は、特定のセル内のどこからデータの読み出しを開始するかを決定する際に使用される。式IIIに含まれるＣＨＵＮＫ＿ＭＡＳＫは、先に考察したマスクの逆（inverse：インバース）である。ＣＨＵＮＫ＿ＭＡＳＫの目的は、セルコントローラをメモリブロックの正確な部分に導くことである。マスクしたアドレス部分は右シフト（記号「＞＞」で表される）され、インタリーブに使用されていたビットを削除する（式IIIでは「インタリーブ」と表される）。セルマップテーブルが８列を含む場合、３つのインタリーブビットが列の識別に使用される。セルマップが１６列を有する場合、列の識別に４ビットが使用される。

【００４１】

ＣＨＵＮＫ＿ＳＴＡＲＴは、他のインタリーブエントリ専用の、または他のインタリーブエントリによって以前使用されていたメモリの量である。たとえば、再び図２のインタリーブエントリ１を参照すると、インタリーブエントリ１は、インタリーブエントリ０が４個のセルそれぞれから２ＧＢを占めた後に来た。したがって、セル０の２番目の２ＧＢ部分がアドレス指定されている場合、ＣＨＵＮＫ＿ＳＴＡＲＴはインタリーブエントリ０に含まれた２ＧＢである。あるいは、引き続き図２を参照すると、セル０における２ＧＢおよびセル１における２ＧＢを含むインタリーブエントリ５は、セル０およびセル１の双方における４ＧＢのＣＨＵＮＫ＿ＳＴＡＲＴを有する。要約すると、ＣＥＬＬ＿ＡＤＤＲ式は、メモリアクセスデバイスからのアドレスビットを使用し、所与のブロック内の特定のメモリロケーションに到達するようにするオフセットに等しいメモリ量を追加し、右シフトしてインタリーブにすでに使用しておりブロック内のロケーションを決定するためにもはや必要のないビットを削除し、他のインタリーブエントリ内のインタリーブにあてられていたセルメモリを追加する。

【００４２】

図９は、セルマップ９００の一実施形態の図である。セルマップ９００は、複数のエントリ９０４をそれぞれ含むモジュールＩＤテーブル９１２およびチャンク開始テーブル９１

４を含む。各エントリ９０４は複数のエントリ項目９１０を含む。代替の実施形態では、モジュールＩＤ情報およびチャンク開始情報を単一テーブルに組み込むことができる。別の代替実施形態では、セルマップ９００はチャンク開始テーブル９１４を含まない。各エントリ９０４におけるエントリ項目９１０中の数は、インタリーブのウェイ数に基づく。たとえば、１６ウェイインタリーブの場合、モジュールＩＤテーブル９１２およびチャンク開始テーブル９１４はそれぞれ、エントリ９０４毎に少なくとも１６個のエントリ項目９１０をそれぞれ含むことになる（図９に示すように）。また、６４ウェイインタリーブの場合、モジュールＩＤテーブル９１２およびチャンク開始テーブル９１４はそれぞれ、エントリ９０４毎に少なくとも６４個のエントリ項目９１０を含むことになる。一実施形態では、モジュールＩＤテーブル９１２における各エントリ項目９１０は、宛先モジュール識別のために８ビットモジュールＩＤ値を含み、チャンク開始テーブル９１４における各エントリ項目９１０は、メモリロケーションの識別に使用されるチャンク開始値を含む。受信した各入力アドレス９０２に基づき、後述するように、適切なモジュールＩＤ値９１６およびチャンク開始値９１８が識別され、セルマップ９００から出力される。

【００４３】

以下の表１は、セルマップベースシステムの一実施形態を実施する擬似コードを含む。

【００４４】

【表１】

1. Incoming_Address[49:0]
2. Address Match[49:29]
3. Address Mask[49:29]
4. Module_id_table_in_entry[Max_ways of interleave in entry * Bits_per_module_id]
5. Chunk_start_table_in_entry[Max_ways of interleave in entry * Number of bits required to address all 0.5GB chunks]
6. Hit_for_that_entry = ((Incoming_Address[49:29]] & Address Mask[49:29]) == Address Match[49:29])
7. Physical_destination_module = Module_id_table_in_entry [Incoming_Address[9:6]]
8. Chunk_start = Chunk_start_table_in_entry [Incoming_Address[9:6]]

【００４５】

上記表１の場合、以下の仮定を行う。すなわち、アドレスサイズの実施は５０ビット（すなわち、ビット０～４９）である；各エントリ９０４は１６ウェイインタリーブされる（したがって、エントリ９０４内でのインデックス付けのために４ビットが必要である）；システムはシングルキャッシュラインインタリーブを実行している；キャッシュラインサイズは６４バイトである（よって、アドレスビット６～９がエントリ９０４内のインデックス付けに使用される）；おおよそ３２～６４のエントリ９０４が各プロセッサインタフェースに設けられる；かつ、最小アドレス指定可能メモリは１／２ＧＢである（よって、アドレスマスク（Ａｄｄｒｅｓｓ　Ｍａｓｋ）はビット２９～４９を使用する）。

【００４６】

上記仮定は一例のシステムに基づき、かつ特定の実施に基づいて変更を行いうることが理解されよう。たとえば、キャッシュラインサイズが６４バイトではなく１２８バイトである場合には、入力アドレス９０２のビット７～１０がエントリ９０４内のインデックス付けに使用される。キャッシュラインサイズが２５６バイトである場合は、入力アドレス９０２のビット８～１１がエントリ９０４内のインデックス付けに使用される。キャッシュラインサイズが倍になるごとに、４つのインデックス付けビットがそれぞれ左に１ビット位置分シフトする。別の例では、最小アドレス指定可能メモリが１／２ＧＢではなく６４メガバイト（ＭＢ）である場合、アドレスマスク（Ａｄｄｒｅｓｓ　Ｍａｓｋ）は、ビッ

ト２９～４９ではなくビット２４～４９を使用する。種々のキャッシュラインサイズおよび異なる最小アドレス指定可能メモリサイズの他に、種々のアドレスサイズ、シングルキャッシュラインインタリーブではなくマルチキャッシュラインインタリーブ、およびインタリーブの種々のウェイ数を含むがこれらに限定されない他の変更を行うこともできる。

【００４７】

表１に列挙される最初の項目はＩｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［４９：０］であり、これは入力アドレス９０２として図９に表される。一実施形態では、入力アドレス９０２はプロセッサアドレスであり、０～４９の番号が付けられた５０ビットを含む。入力アドレス９０２は、第１のアドレス部分９０２Ａ、第２のアドレス部分９０２Ｂ、および第３のアドレス部分９０２Ｃを含む。

【００４８】

表１に列挙される２番目の項目は、Ａｄｄｒｅｓｓ　Ｍａｔｃｈ［４９：２９］である。一実施形態では、セルマップ９００における各エントリ９０４は、Ａｄｄｒｅｓｓ　Ｍａｔｃｈ［４９：２９］の特定の値に関連付けられる。

【００４９】

表１に列挙する３番目の項目は、Ａｄｄｒｅｓｓ　Ｍａｓｋ［４９：２９］である。Ａｄｄｒｅｓｓ　Ｍａｓｋ［４９：２９］は、入力アドレス９０２から関連するビットを抽出するためにセルコントローラ（たとえば、セルコントローラ１７５）により使用される。一実施形態では、入力アドレス９０２のビット２９～４９は第１のアドレス部分９０２Ａと呼ばれる。セルコントローラ１７５は、マスクされた第１のアドレス部分９０２Ａを、Ａｄｄｒｅｓｓ　Ｍａｔｃｈ値と比較してマッチするエントリ９０４を識別する。

【００５０】

モジュールＩＤテーブル９１２におけるエントリ９０４は、表１に列挙される４番目の項目で表され、この項目はＭｏｄｕｌｅ＿ｉｄ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙ［エントリにおけるインタリーブのＭａｘ＿ｗａｙｓ＊Ｂｉｔｓ＿ｐｅｒ＿ｍｏｄｕｌｅ＿ｉｄ］である。Ｍｏｄｕｌｅ＿ｉｄ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙの括弧内の値は、モジュールＩＤテーブル９１２におけるエントリ９０４のビットサイズを表す。各モジュールＩＤに８ビットを使用し（すなわち、Ｂｉｔｓ＿ｐｅｒ＿ｍｏｄｕｌｅ＿ｉｄ＝８）、かつエントリ９０４が１６ウェイインタリーブされる（すなわち、エントリにおけるインタリーブのＭａｘ＿Ｗａｙｓ＝１６）と仮定すると、モジュールＩＤテーブル９１２の各エントリ９０４に必要なビット数は１２８（すなわち、８×１６）になる。６４ウェイインタリーブエントリ９０４の場合、モジュールＩＤテーブルエントリ９０４のサイズは５１２ビット（すなわち、６４×８）になる。モジュールＩＤにつき８ビットを使用すると、２５６個のモジュールを一意に識別することができる。特定の実施に応じて、モジュールＩＤにつき他のビット数を使用してもよい。

【００５１】

チャンク開始テーブル９１４のエントリ９０４は、表１に列挙される５番目の項目で表され、この項目はＣｈｕｎｋ＿ｓｔａｒｔ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙ［エントリにおけるインタリーブのＭａｘ＿Ｗａｙｓ＊０．５ＧＢのチャンクすべてをアドレス指定するために必要なビット数］である。Ｃｈｕｎｋ＿ｓｔａｒｔ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙの括弧内の値は、チャンク開始テーブル９１４におけるエントリ９０４のビットサイズを表す。すべての１／２ＧＢチャンクをアドレス指定するために８ビットが使用され、かつエントリ９０４が１６ウェイインタリーブされる（すなわち、エントリにおけるインタリーブのＭａｘ＿ｗａｙｓ＝１６）と仮定すると、チャンク開始テーブル９１４の各エントリ９０４に必要なビット数は１２８（すなわち、８×１６）になる。６４ウェイインタリーブエントリ９０４の場合、チャンク開始テーブルエントリ９０４のサイズは５１２ビット（すなわち、６４×８）になる。

【００５２】

表１に列挙される６番目の項目は、Ｈｉｔ＿ｆｏｒ＿ｔｈａｔ＿ｅｎｔｒｙ＝（（Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［４９：２９］）＆Ａｄｄｒｅｓｓ　Ｍａｓｋ［４９：２９

]）＝＝Ａｄｄｒｅｓｓ Ｍａｔｃｈ［４９：２９］）である。一実施形態では、セルマップ９００における各エントリ９０４は、Ａｄｄｒｅｓｓ Ｍａｔｃｈ［４９：２９］の特定の値に関連付けられ、表１に列挙される６番目の項目のようなヒット式が、ヒットが識別されるまで、セルマップ９００における各エントリ９０４について実行される。変数Ｈｉｔ＿ｆｏｒ＿ｔｈａｔ＿ｅｎｔｒｙは、（Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［４９：２９］］＆Ａｄｄｒｅｓｓ Ｍａｓｋ［４９：２９］）とＡｄｄｒｅｓｓ Ｍａｔｃｈ［４９：２９］の比較に応じて、真の論理値かあるいは偽の論理値のいずれかを含むことになる。（Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［４９：２９］］＆Ａｄｄｒｅｓｓ Ｍａｓｋ［４９：２９］）の値とＡｄｄｒｅｓｓ Ｍａｔｃｈ［４９：２９］の値が等しくない場合、Ｈｉｔ＿ｆｏｒ＿ｔｈａｔ＿ｅｎｔｒｙは、偽の論理値になり、マッチしないエントリであることを示す。（Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［４９：２９］］＆Ａｄｄｒｅｓｓ Ｍａｓｋ［４９：２９］）の値とＡｄｄｒｅｓｓ Ｍａｔｃｈ［４９：２９］の値が等しい場合、Ｈｉｔ＿ｆｏｒ＿ｔｈａｔ＿ｅｎｔｒｙは、真の論理値になり、マッチするエントリであることを示す。

【００５３】

モジュールＩＤテーブル９１２の各エントリ９０４の場合、チャンク開始テーブル９１４に対応するエントリ９０４がある。ヒットがエントリ９０４に対して生成される場合、モジュールＩＤテーブルエントリからのモジュールＩＤ値９１６が出力され、対応するチャンク開始テーブルエントリからチャンク開始番号９１８が出力される。一実施形態では、チャンク開始番号９１８は、モジュールＩＤ値９１６によって識別されるメモリ内のメモリロケーションを識別するための１／２ＧＢの倍数である。エントリ９０４内の適切なモジュールＩＤ値９１６およびチャンク開始番号９１８を識別するための式は、表１の７番目および８番目の項目に列挙される。

【００５４】

表１に列挙される７番目の項目は、Ｐｈｙｓｉｃａｌ＿ｄｅｓｔｉｎａｔｉｏｎ＿ｍｏｄｕｌｅ＝Ｍｏｄｕｌｅ＿ｉｄ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙ［Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［９：６］］である。この項目が示すように、入力アドレス９０２のビット６～９を使用して、モジュールＩＤテーブル９１２でマッチするエントリ９０４における１６個のエントリ項目９１０の１つを識別する。一実施形態では、入力アドレス９０２のビット６～９は第３のアドレス部分９０２Ｃと呼ばれる。識別されたエントリ項目９１０からのモジュールＩＤ値は、変数Ｐｈｙｓｉｃａｌ＿ｄｅｓｔｉｎａｔｉｏｎ＿ｍｏｄｕｌｅに格納される。６４ウェイインタリーブエントリ９０４を有するモジュールＩＤテーブル９１２の場合、モジュールＩＤテーブルエントリ９０４にインデックスを付ける式は、Ｐｈｙｓｉｃａｌ＿ｄｅｓｔｉｎａｔｉｏｎ＿ｍｏｄｕｌｅ＝Ｍｏｄｕｌｅ＿ｉｄ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙ［Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［１１：６］］である。この場合、第３のアドレス部分９０２Ｃはアドレスビット６～１１を含み、これらのアドレスビットを使用してモジュールＩＤテーブルエントリ９０４にインデックスを付ける。

【００５５】

表１に列挙される８番目の項目は、Ｃｈｕｎｋ＿ｓｔａｒｔ＝Ｃｈｕｎｋ＿ｓｔａｒｔ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙ［Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［９：６］］である。この項目が示すように、入力アドレス９０２の第３のアドレス部分９０２Ｃを使用して、チャンク開始テーブル９１４でマッチするエントリ９０４における１６個のエントリ項目９１０の１つを識別する。識別されたエントリ項目９１０からのチャンク開始番号が変数ｃｈｕｎｋ＿ｓｔａｒｔに格納される。６４ウェイインタリーブエントリ９０４を有するチャンク開始テーブル９１４の場合には、チャンク開始テーブルエントリ９０４にインデックスを付けるための式は、Ｃｈｕｎｋ＿ｓｔａｒｔ＝Ｃｈｕｎｋ＿ｓｔａｒｔ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙ［Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［１１：６］］である。この場合、第３のアドレス部分９０２Ｃはアドレスビット６～１１を含み、これらのアドレスビットをチャンク開始テーブルエントリ９０４へのインデックス付けに使用する。上述したように、一実施形態では、特定のメモリロケーションを識別するために、チャンク

開始値をセルアドレス式（たとえば、式III）で使用する。

【００５６】

セルマップ９００等のセルマップを使用してＩ／Ｏ仮想化を達成するため、メモリに使用されるエントリ９０４とＩ／Ｏに使用されるエントリ９０４とを区別する機構が設けられる。また、Ｉ／Ｏタイプエントリ９０４の場合、モジュールＩＤテーブル９１２へのインデックス付けに使用する種々のアドレスビットの組の選択に複数のオプションを提供することができる。したがって、メモリタイプエントリとＩ／Ｏタイプエントリとを区別するために、かつＩ／Ｏに追加のインデックスオプションを提供するために、追加の状態がセルマップ９００に設けられる。一実施形態では、セルマップ９００がチャンク開始テーブル９１４を含む場合、メモリタイプエントリ９０４のみが関連するエントリをチャンク開始テーブル９１４に含み、チャンク番号はＩ／Ｏタイプエントリ９０４で使用されない。

【００５７】

一実施形態では、セルマップ９００におけるエントリ９０４は、１ビットのエントリタイプ識別子９０６を含む。本発明の一形態では、エントリタイプ識別子９０６の値が０である場合、エントリ９０４はメモリタイプエントリであり、エントリタイプ識別子９０６の値が１の場合、エントリ９０４はＩ／Ｏタイプエントリである。また、一実施形態では、エントリ９０４は２ビットのインデックス選択識別子９０８を含む。インデックス選択識別子９０８の２ビットは、可能な４つの異なる値を提供し、これらの値を使用して、Ｉ／Ｏテーブルエントリ９０４についてモジュールＩＤテーブル９１２にインデックスを付けるためのアドレスビットの組を選択する。

【００５８】

セルマップ９００におけるメモリタイプエントリ９０４の場合、ヒットロジック、モジュールＩＤ識別、およびチャンク開始番号識別は、上述したものと同じである。具体的には、表１の６番目の項目に示すように、Ｈｉｔ＿ｆｏｒ＿ｔｈａｔ＿ｅｎｔｒｙ＝（（Ｉｎｃｏｍｉｎｇ＿Ａｄｄｒｅｓｓ［４９：２９］］＆Ａｄｄｒｅｓｓ　Ｍａｓｋ［４９：２９］）＝＝Ａｄｄｒｅｓｓ　Ｍａｔｃｈ［４９：２９］）である。よって、入力アドレス９０２のビット２９～４９（すなわち、第１のアドレス部分９０２Ａ）を使用して、セルマップ９００におけるマッチするエントリ９０４を識別する。また、表１の７番目および８番目の項目に示すように、入力アドレス９０２のビット６～９（すなわち、第３のアドレス部分９０２Ｃ）を使用して、モジュールＩＤテーブル９１２におけるメモリタイプエントリ９０４およびチャンク開始テーブル９１４におけるチャンク開始エントリ９０４にインデックスを付ける。

【００５９】

一実施形態では、Ｉ／Ｏタイプエントリ９０４の場合、ヒットロジックはメモリタイプエントリ９０４の場合に使用したものと同じであるが、より少ない数のアドレスビットが比較される。本発明の一形態では、これは、Ａｄｄｒｅｓｓ　Ｍａｔｃｈレジスタおよび Ａｄｄｒｅｓｓ　Ｍａｓｋレジスタにおいてより多くのビットをマスクすることによって達成される。また、Ｉ／Ｏタイプエントリ９０４の場合、モジュールＩＤテーブル９１２へのインデックスは、メモリタイプエントリ９０４の場合とは別様に選択される。第２のアドレス部分９０２Ｂと呼ぶ、Ｉ／Ｏタイプエントリ９０４にインデックスを付けるために使用される入力アドレスビットの組は、そのエントリ９０４のインデックス選択識別子９０８に基づいて選択される。以下の表２は、本発明の一実施形態によるインデックス選択識別子９０８の値およびインデックス付けに使用される対応する入力アドレスビットを示す。

【００６０】

【表２】

| Index Select bit values | Address bits for indexing |
|---|---|
| 00 | Incoming_address[31:28]// Allows 256MB per IO module |
| 01 | Incoming_address[35:32]// Allows 4GB per IO module |
| 10 | Incoming address[37:34// Allows 16GB per IO module |
| 11 | Incoming Address[39:36]// Allows 64GB per IO module |

【００６１】
ヒットが決定された（すなわち、セルマップ９００におけるエントリ９０４が第１のアドレス部分９０２に基づいて識別された）後、識別されたエントリ９０４内の特定のエントリ項目９１０がインデックス値に基づいて選択される。以下の表３は、本発明の一実施形態によるモジュールＩＤテーブル９１２におけるエントリ９０４にインデックスを付けるための擬似コードを含む。
【００６２】
【表３】

```
Switch(Index_select)
Case 00: index_module_table_io = Incoming_address[31:28];
        break
Case 01: index_module_table_io = Incoming_address[35:32];
        break
Case 10: index_module_table_io = Incoming_address[37:34];
        break
Case 11: index_module_table_io = Incoming_address[39:36];
        break
endofswitch
index_module_table_mem = Incoming Address[9:6]
switch(Entry_type)
Case MEM: index_module_table = index_module_table_mem
Case IO: index_module_table = index_module_table_io
endofswitch
Physical Destination Module(type) =
        Module_id_table_in_entry[index_module_table]
```

【００６３】
最初の「ｓｗｉｔｃｈ」コードセグメントは、マッチするエントリ９０４についてインデックス選択識別子９０８の値に基づいてｉｎｄｅｘ＿ｍｏｄｕｌｅ＿ｔａｂｌｅ＿ｉｏに値を割り当てる。次に、入力アドレスのビット６～９（すなわち、第３のアドレス部分９０２Ｃ）に基づいてｉｎｄｅｘ＿ｍｏｄｕｌｅ＿ｔａｂｌｅ＿ｍｅｍに値を割り当てる。２番目の「ｓｗｉｔｃｈ」コードセグメントは、マッチするエントリ９０４のエントリタイプ９０６に基づいてｉｎｄｅｘ＿ｍｏｄｕｌｅ＿ｔａｂｌｅに値を割り当てる。エントリ９０４がメモリ（ＭＥＭ）タイプエントリである場合、ｉｎｄｅｘ＿ｍｏｄｕｌｅ＿ｔａｂｌｅにはｉｎｄｅｘ＿ｍｏｄｕｌｅ＿ｔａｂｌｅ＿ｍｅｍの値が割り当てられる。エントリがＩ／Ｏタイプエントリである場合、ｉｎｄｅｘ＿ｍｏｄｕｌｅ＿ｔａｂｌｅには

ｉｎｄｅｘ＿ｍｏｄｕｌｅ＿ｔａｂｌｅ＿ｉｏの値が割り当てられる。最後に、ｉｎｄｅｘ＿ｍｏｄｕｌｅ＿ｔａｂｌｅに割り当てられた値を、マッチするエントリ９０４（すなわち、Ｍｏｄｕｌｅ＿ｉｄ＿ｔａｂｌｅ＿ｉｎ＿ｅｎｔｒｙ）のインデックスとして使用して、エントリ９０４における１６個のエントリ項目９１０の１つを識別する。識別されたエントリ項目９０４は、変数「Ｐｈｙｓｉｃａｌ　Ｄｅｓｔｉｎａｔｉｏｎ　Ｍｏｄｕｌｅ」に割り当てられるモジュールを識別するための識別情報を含む。メモリタイプエントリ９０４の場合、対応するエントリ９０４からのチャンク開始値およびチャンク開始テーブル９１４におけるエントリ項目９１０もまた上述のように識別される。

【００６４】

よって、セルマップエントリ９０４のタイプに応じて、適切なモジュール情報が抽出および使用される。代替の実施形態では、複数のエントリ９０４を連結して、より多数のモジュールを仮想化することができる。

【００６５】

本発明の一実施形態は、セルマップ９００を使用してＩ／Ｏモジュール１７０の仮想化およびメモリインタリーブを提供する。本発明の一形態では、Ｉ／Ｏモジュール１７０は、周辺コンポーネント相互接続（ＰＣＩ）デバイス等の複数のＩ／Ｏカードまたはデバイスを制御するようにそれぞれ構成されたＩ／Ｏコントローラである。したがって、入力アドレス９０２が第１のＩ／Ｏモジュールに向けられ、かつセルマップ９００を使用してアクセスを第２のＩ／Ｏモジュールにリダイレクトされる場合、第２のＩ／Ｏモジュール下のＩ／Ｏデバイスは、ソフトウェアからは第１のＩ／Ｏモジュール下のＩ／Ｏデバイスと同じように見えるはずである。代替の実施形態では、仮想化をＩ／Ｏカードまたはデバイスのレベルにまで拡張することができる。

【００６６】

本発明の一形態は、同じセルマップインフラストラクチャをメモリインタリーブおよびＩ／Ｏ仮想化の双方に使用することにより、メモリインタリーブ化およびＩ／Ｏ仮想化に別個の構造を設計する必要性をなくす。本発明の一実施形態では、仮想化することのできるモジュールの数に制限がない。本発明の一形態は、ソフトウェア介入なしで、または最小限のサポートでＩ／Ｏモジュール１７０を変更可能なシステムを構築できるようにする。一実施形態では、本発明は、Ｉ／Ｏの仮想化において、従来提供されているシステムよりも高い柔軟性を提供する。また、本発明の一形態では、単一のセルマップエントリ９０４を使用して複数のＩ／Ｏモジュール１７０を仮想化することができ、またセルマップエントリ９０４をメモリ目的とＩ／Ｏ目的とで交換して使用することができる。この発明は、例として次のような実施形態を含む。

【００６７】

（１）　連続した論理アドレス空間を使用して、複数のメモリ（１３０、１３５、１４０、１４５）にインタリーブ方式でアクセスし、複数の入出力モジュール（１７０）にアクセスする方法であって、

複数のエントリ（９０４）を含む少なくとも１つのマップテーブル（９００）を設けることであって、前記エントリ（９０４）はそれぞれエントリタイプ識別子（９０６）および複数のエントリ項目（９１０）を含み、前記エントリ項目（９１０）はそれぞれモジュール識別子を含み、前記エントリ（９０４）はそれぞれメモリタイプエントリおよび入出力タイプエントリの一方であり、

複数のアドレスビット（９０２Ａ～９０２Ｃ）を含む第１の論理アドレス（９０２）を受信することと、

前記アドレスビットの第１の組に基づいて、前記少なくとも１つのマップテーブルにおけるエントリを識別することと、

前記識別されたエントリの前記エントリタイプ識別子に基づいて、前記識別されたエントリのタイプを決定することと、

前記エントリタイプ識別子が入出力タイプエントリを示す場合、前記アドレスビットの第２の組に基づいて前記識別されたエントリにおけるエントリ項目を識別することと、

前記エントリタイプ識別子がメモリタイプエントリを示す場合、前記アドレスビットの第３の組に基づいて前記識別されたエントリにおけるエントリ項目を識別することと、
前記識別されたエントリ項目の前記モジュール識別子によって識別されたモジュールにアクセスすることと
を含む方法。

【００６８】

（２）　前記少なくとも１つのマップテーブルにおける各入出力タイプエントリは、インデックス選択識別子（９０８）をさらに含み、
前記識別されたエントリの前記インデックス選択識別子に基づいて、前記第２のアドレスビットの組に使用するアドレスビットを識別することをさらに含む（１）記載の方法。

【００６９】

（３）　前記インデックス選択識別子はそれぞれ、複数のインデックス値の１つであり、前記インデックス値はそれぞれアドレスブロックサイズに対応する（２）記載の方法。

【００７０】

（４）　前記第１のアドレスビットの組は、前記第２のアドレスビットの組よりも上位のビットを含み、前記第２のアドレスビットの組は、前記第３のアドレスビットの組よりも上位のビットを含む（１）記載の方法。

【００７１】

（５）　前記少なくとも１つのマップテーブルに複数のメモリオフセット値（９１４）を格納することと、
前記第１の論理アドレスに基づいて前記メモリオフセット値（９１４）の１つを識別することとをさらに含み、
前記エントリタイプ識別子がメモリタイプエントリを示す場合、前記識別されたエントリ項目の前記モジュール識別子によって識別される前記モジュールは、前記識別されたメモリオフセット値に少なくとも部分的に基づいてメモリロケーションにおいてアクセスされる（１）記載の方法。

【００７２】

（６）　マルチビットマスク値を設けることと、
複数のマルチビット整合値を設けることと、
前記マルチビットマスク値を使用して前記第１の論理アドレスから前記第１のアドレスビットの組を抽出することと、
該抽出された第１のアドレスビットの組を前記複数のマルチビット整合値と比較してマッチングを識別することと
をさらに含む（１）記載の方法。

【００７３】

（７）　前記メモリはそれぞれ少なくとも１つのメモリセグメントを含み、前記メモリセグメントはグループに編成され、各グループの前記メモリセグメントは均等なサイズを有し、また前記少なくとも１つのマップテーブルにおける各メモリタイプエントリは、前記メモリセグメントグループの１つに対応する（１）記載の方法。

【００７４】

（８）　連続した論理アドレス空間を複数のモジュールにマッピングする方法であって、
前記複数のモジュールはメモリ（１３０、１３５、１４０、１４５）および入出力モジュール（１７０）を含み、前記論理アドレス空間における各論理アドレス（９０２）は第１、第２、および第３のアドレス部分（９０２Ａ～９０２Ｃ）を含み、
複数のエントリ（９０４）を含むマップテーブル（９００）を設けることであって、前記エントリ（９０４）はそれぞれエントリタイプ識別子（９０６）および複数のエントリ項目（９１０）を含み、前記エントリ（９０４）はそれぞれメモリタイプエントリおよび入出力タイプエントリの一方であり、前記各エントリ項目（９１０）は前記複数のモジュールの１つを識別するモジュール識別子を含み、
論理アドレスの組を前記エントリのそれぞれに関連付けることであって、個々のエントリ

それぞれに関連付けられる前記論理アドレスの組は、前記第１のアドレス部分について共通する値を含み、前記第１のアドレス部分についての前記共通の値は前記エントリそれぞれごとに異なり、
前記エントリに関連する前記論理アドレスの組の前記第２および前記第３のアドレス部分に基づいて各エントリ内のエントリ項目にインデックスを付けることであって、インデックス付けに使用される前記アドレス部分の選択は、前記エントリの前記エントリタイプ識別子に基づく、
を含む方法。

【００７５】

（９）　連続した論理アドレス空間を使用して複数のモジュールへのアクセスを提供するシステム（１００）であって、前記モジュールはメモリ（１３０、１３５、１４０、１４５）および入出力モジュール（１７０）を含み、
複数のエントリ（９０４）を含む少なくとも１つのマップテーブル（９００）であって、前記エントリ（９０４）はそれぞれエントリタイプ識別子（９０６）および複数のエントリ項目（９１０）を含み、前記エントリ（９０４）はそれぞれメモリタイプエントリおよび入出力タイプエントリの一方であり、前記エントリ項目（９１０）はそれぞれ複数のモジュールの１つを識別する、少なくとも１つのマップテーブル（９００）と、
論理アドレスを受信するコントローラ（１７５）であって、受信した論理アドレス（９０２）の第１の部分に基づいて前記少なくとも１つのマップテーブルにおけるエントリを識別し、該識別されたエントリの前記エントリタイプ識別子および前記受信した論理アドレスの第２の部分に基づいて前記識別されたエントリにおけるエントリ項目を選択し、該選択されたエントリ項目によって識別されるモジュールに基づいてモジュール識別情報を出力するように構成されるコントローラ（１７５）とを備えるシステム。

【００７６】

（１０）　前記第２の部分のロケーションは、前記識別されたエントリのタイプに基づいて、前記受信した論理アドレス内で変化する（９）記載のシステム。

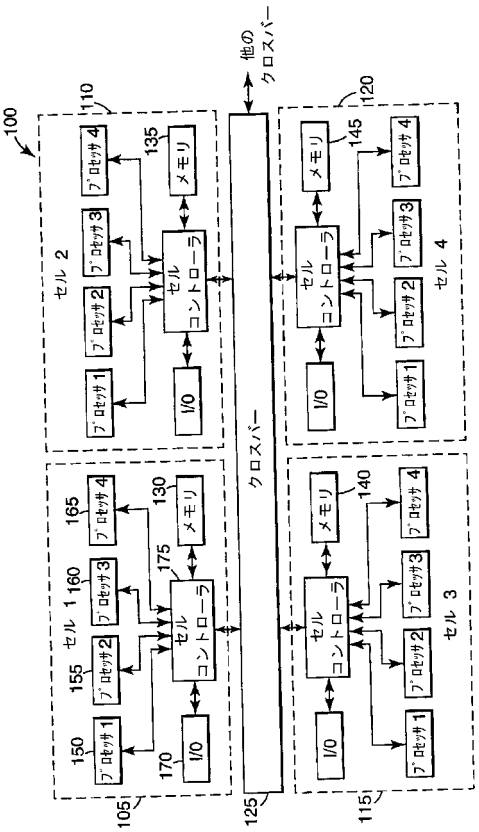
【００７７】

好ましい実施形態の説明を目的として、本発明の特定の実施形態を本明細書に図示し説明したが、本発明の範囲から逸脱することなく図示し説明した特定の実施形態を多種多様な代替および／または同等の実施で置き換えてもよいことが当業者には理解されよう。化学、機械、電気機械、電気、およびコンピュータの分野における当業者は、本発明が非常に広範囲の実施形態において実施されうることをたやすく理解しよう。本願は、本明細書に述べた好ましい実施形態のあらゆる改変または変形を網羅するものである。したがって、本発明は特許請求の範囲およびその等価物によってのみ制限されるものと明らかに意図される。

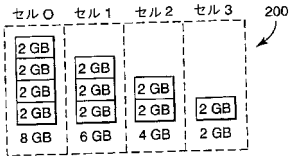
【図面の簡単な説明】

【図１】４個のセルおよび１個のクロスバーを含むノードを示すブロック図である。
【図２】メモリが４個のセルに分散する分散メモリシステムを示すブロック図である。
【図３】４ウェイインタリーブおよび８ウェイインタリーブを示すテーブルである。
【図４】３ウェイインタリーブを示すテーブルである。
【図５】シングルキャッシュラインインタリーブの６４ウェイインタリーブドセルマップエントリを示すブロック図である。
【図６】インタリーブグループの間での顕著な特徴を識別するチャートである。
【図７】２のべき乗ではないメモリ量を含むセルにわたるインタリーブのブロック図である。
【図８】３個のセルにわたるインタリーブのブロック図である。
【図９】本発明の一実施形態によるメモリインタリーブおよびＩ／Ｏ仮想化を提供するセルマップの図である。

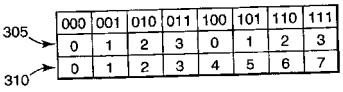
【図１】



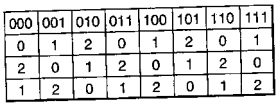

【図２】



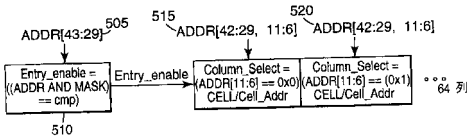

【図３】



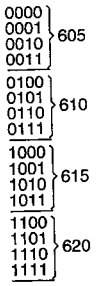

【図４】



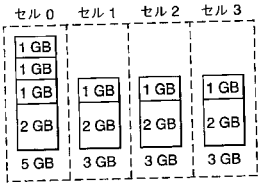

【図５】



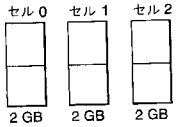

【図６】



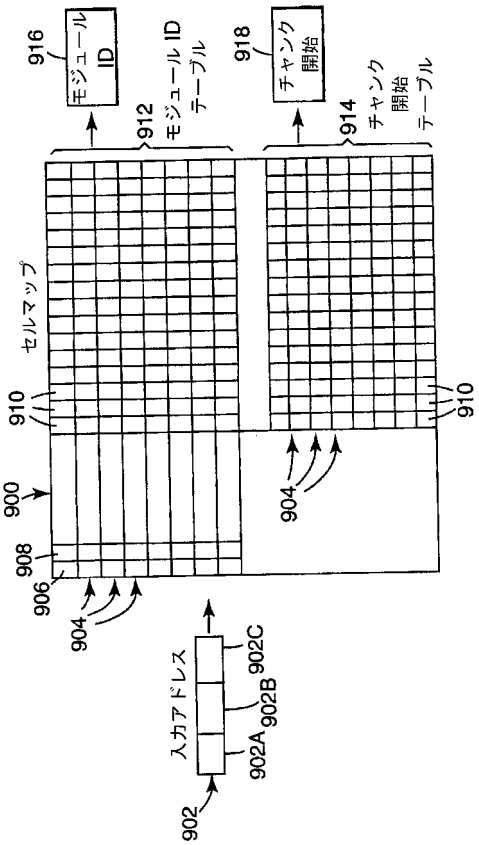

【図７】




【図８】




【図９】



フロントページの続き

(72)発明者　アシシュ・ガプタ
アメリカ合衆国９５１２９カリフォルニア州サン・ノゼ、オラ・ストリート　５６３７
(72)発明者　デベンドラ・ダス・シャーマ
アメリカ合衆国９５０５０カリフォルニア州サンタ・クララ、アカシア・コート　２０４３

審査官　▲高▼橋　正▲徳▼

(56)参考文献　特開平０８－１０１７９２（ＪＰ，Ａ）
特開昭６２－１３１３５１（ＪＰ，Ａ）
国際公開第９９／０２４９０６（WO，Ａ１）
特開平０８－１０１８０２（ＪＰ，Ａ）
特開平０５－１１３９３０（ＪＰ，Ａ）
特開平１０－３０１８４２（ＪＰ，Ａ）
特開平０４－１９１９４３（ＪＰ，Ａ）
特開２０００－３３０８６５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　12/00-12/06,
G06F　13/14,
G06F　15/167