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

特許第5352780号
(P5352780)

(45)発行日　平成25年11月27日(2013.11.27)　(24)登録日　平成25年9月6日(2013.9.6)

(51)Int.Cl.		F I		
G06F 15/82	(2006.01)	G06F	15/82	620A
G06F 15/173	(2006.01)	G06F	15/173	640C
G06F 13/36	(2006.01)	G06F	13/36	530A
		G06F	15/82	630E

請求項の数 19　(全 16 頁)

(21)出願番号	特願2010-509892(P2010-509892)
(86)(22)出願日	平成20年5月30日(2008.5.30)
(65)公表番号	特表2010-528387(P2010-528387A)
(43)公表日	平成22年8月19日(2010.8.19)
(86)国際出願番号	PCT/GB2008/001821
(87)国際公開番号	WO2008/145995
(87)国際公開日	平成20年12月4日(2008.12.4)
審査請求日	平成23年5月13日(2011.5.13)
(31)優先権主張番号	0710377.3
(32)優先日	平成19年5月31日(2007.5.31)
(33)優先権主張国	英国(GB)

(73)特許権者　513174092
アンダーソン、ジェームス、アーサー、ディーン、ワレス
ＡＮＤＥＲＳＯＮ　ＪＡＭＥＳ　ＡＲＴＨＵＲ　ＤＥＡＮ　ＷＡＬＬＡＣＥ
イギリス、レディング　アールジー４　５エルイー、カバシャム、ローアー　ヘンリー　ロード　８８番地
(74)代理人　100094983
弁理士　北澤　一浩
(74)代理人　100095946
弁理士　小泉　伸
(74)代理人　100099829
弁理士　市川　朗子

最終頁に続く

(54)【発明の名称】プロセッサ

(57)【特許請求の範囲】
【請求項１】
各々がインストラクションを実行するように配置された複数のプロセッサと、前記プロセッサ間でデータトークン及びコントロールトークンを搬送するように配置されたバスとを有し、
　各プロセッサは、バスを介してコントロールトークンを受け取る場合に、前記インストラクションを実行し、前記インストラクションを実行する際には、データに演算を行って結果を生成し、データ対象プロセッサとなるべきプロセッサを特定し、特定されたデータ対象プロセッサに出力データを伝送し、制御対象プロセッサとなるべきプロセッサを特定し、特定した制御対象プロセッサにコントロールトークンを伝送することを特徴とする処理装置。
【請求項２】
各プロセッサは、任意のデータ対象プロセッサのアドレスと一緒に前記バスに前記出力データを書き込むように配置されていることを特徴とする請求項１記載の処理装置。
【請求項３】
各プロセッサは、前記出力データが並列に送られる複数のデータ対象プロセッサを特定できることを特徴とする請求項１又は２記載の処理装置。
【請求項４】
前記バスは、前記特定されたデータ対象プロセッサに前記出力データを伝送するように配置され、前記出力データは、前記データ対象プロセッサに書き込まれることを特徴とする

請求項１乃至３のいずれか一に記載の処理装置。
【請求項５】
各プロセッサは、コントロールトークンが伝送される前記制御対象プロセッサのアドレスと共にバスに前記コントロールトークンを書き込むことによって前記コントロールトークンを伝送するように、配置されていることを特徴とする請求項１乃至４のいずれか一に記載の処理装置。
【請求項６】
各プロセッサは、前記インストラクションを実行する際、コントロールトークンを並列に伝送できる複数の制御対象プロセッサを特定できることを特徴とする請求項１乃至５のいずれか一に記載の処理装置。
【請求項７】
各プロセッサは、特定された対象プロセッサのいずれかに前記出力データ及びコントロールトークンを伝送するときに、他のコントロールトークンを受け取るまでは前記インストラクションを再度実行しないことを特徴とする請求項１乃至６のいずれか一に記載の処理装置。
【請求項８】
各プロセッサは、同一のインストラクションを実行するように配置されていることを特徴とする請求項１乃至７のいずれか一に記載の処理装置。
【請求項９】
各プロセッサは、唯一のインストラクションを実行するように配置されていることを特徴とする請求項１乃至８のいずれか一に記載の処理装置。
【請求項１０】
前記インストラクションは、
　ａ×ｂ＋ｃ　－＞　ｒ’
の乗算及び加算であることを特徴とする請求項１乃至９のいずれか一に記載の処理装置。
【請求項１１】
各プロセッサは、前記結果に基づいて対象プロセッサを選択するように配置されていることを特徴とする請求項１乃至１０のいずれか一に記載の処理装置。
【請求項１２】
各プロセッサは、前記結果が、ゼロ未満、ゼロ、ゼロよりも大、又は無効のいずれに該当するかを判別し、それに応じて、対象プロセッサを選択するように配置されていることを特徴とする請求項１１記載の処理装置
【請求項１３】
各プロセッサは、前記インストラクションへの入力が記憶される複数のメモリセルを有することを特徴とする請求項１乃至１２のいずれか一に記載の処理装置。
【請求項１４】
各プロセッサは、対象プロセッサのアドレスが記憶される複数のメモリセルを有することを特徴とする請求項１乃至１３のいずれか一に記載の処理装置。
【請求項１５】
各プロセッサは、前記オペレーションの結果が記憶される複数のメモリセルを有することを特徴とする請求項１乃至１４のいずれか一に記載の処理装置。
【請求項１６】
各プロセッサの全メモリは、電源投入時には固定値に設定されることを特徴とする請求項１乃至１５のいずれか一に記載の処理装置。
【請求項１７】
各々が複数のプロセッサからなる複数のチップを有し、各チップは、トークンが他のチップに転送される複数の出力装置を有し、
　各チップ上の各プロセッサは、関係するアドレスを有し、前記アドレスは範囲の内部にあり、
　前記範囲の外側にある対象アドレスを有するトークンを出力装置によって受け取ると、

前記対象アドレスの変更を実行し、前記トークンを前記他のチップに転送することを特徴とする請求項１乃至１６のいずれか一に記載の処理装置。
【請求項１８】
前記出力装置は、前記変更を実行するように配置されていることを特徴とする請求項１７に記載の処理装置。
【請求項１９】
前記変更を実行するために配置された、さらなるオフチップ装置を有することを特徴とする請求項１７に記載の処理装置。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、プロセッサに関する。
【背景技術】
【０００２】
プロセッサ・チップは、概して多数の個別プロセッサを有し、個々のプロセッサは、それぞれインストラクションを実行する構成となっている。通常、多くの異なるインストラクションは、異なるプロセッサによって実行され、個々のプロセッサは、ホストメモリと通信する。
【発明の概要】
【発明が解決しようとする課題】
【０００３】
各プロセッサにおいて、数多くのインストラクションをコード化する必要があるため、プロセッサは大型化し、一つのチップ上に組みつけられるプロセッサの数は限られてしまう。また、各プロセッサは、ホストメモリと通信しなければならないため、処理が遅くなってしまう。
【課題を解決するための手段】
【０００４】
本発明は、複数のプロセッサからなる処理装置を提供する。各プロセッサは、単一のインストラクションを実行するよう構成され、このインストラクションは、各プロセッサで同じでもよい。この処理装置は、更に、プロセッサ間でデータトークンとコントロールトークンを伝送するためのバスを備える。各プロセッサは、バスを介してコントロールトークンを受信すると、インストラクションを実行する。インストラクションを実行する際、各プロセッサはデータに対して演算を行う。これは、データ対象プロセッサとなるべきプロセッサを特定してもよい。プロセッサは、その特定されたデータ対象プロセッサに出力データを送信することもできる。プロセッサは、また、制御対象プロセッサとなるべきプロセッサを特定することも可能であり、その特定された制御対象プロセッサにコントロールトークンを送信することもできる。
【０００５】
出力データは、インストラクションの結果、あるいは、例えばプロセッサ内に保存されたデータであってもよい。
【０００６】
バスは、プロセッサ間でデータトークンとコントロールトークンを伝達するが、その際、ホストメモリからデータを取ってくる必要がない。
【０００７】
バスは、複数のバスフレームを有しており、各フレーム間でデータトークンやコントロールトークンを移動させて、データトークンとコントロールトークンがバスに沿って伝達されるよう構成されてもよい。各プロセッサには、対応する１つ以上のバスフレームが設けられ、データは、該バスフレームからプロセッサに書き込まれる。
【０００８】

また、データは、データトークンという形でバスに送信されてもよい。

【０００９】

各プロセッサは、他のすべてのプロセッサと同じインストラクションを実行するように配置されてもよい。各プロセッサは、１つのインストラクションのみを実行するよう構成されてもよい。各プロセッサは、インストラクションを実行する度に、データ対象プロセッサを０、１、あるいは１つ以上特定することができ、また制御対象プロセッサも０、１、あるいは１つ以上特定することが可能である。これにより、各プロセッサは、複数のプロセッサに並行してデータを送信することができる。バスが特定されたデータ対象プロセッサに演算結果を送信するように構成される。そのデータ対象プロセッサに演算結果が書き込まれる。

【００１０】

好ましくは、各プロセッサは、コントロールトークンが送信されるべき制御対象プロセッサのアドレスと一緒にコントロールトークンをバスに書き込むことによって、コントロールトークンを送信するように構成されている。各プロセッサは、インストラクションを実行する時、コントロールトークンを並行して送信する制御対象プロセッサを複数特定することができる。

【００１１】

各プロセッサは、対象として特定されたプロセッサに演算結果やコントロールトークンを送信する際、そのコントロールトークンを手放すように構成されていることが望ましい。これにより、各プロセッサは、次のコントロールトークンを受け取るまで、そのインストラクションを再度実行することはない。

【００１２】

インストラクションは、ａ× ｂ+ ｃ －＞ ｒ’の形式の乗算加算であってもよい。

【００１３】

各プロセッサは、演算結果ｒ’を基に制御プロセッサを選択するように構成されてもよい。例えば、各プロセッサは、演算結果ｒ’が０より小さいか、０に等しいか、０より大きいか、あるいは無効であるかを判断し、それにしたがって、制御対象プロセッサあるいはデータ対象プロセッサを選択する。

【００１４】

各プロセッサは、インストラクションの入力が記憶される複数のメモリセルを備えてもよい。各プロセッサは、制御対象プロセッサのアドレスが記憶される複数のメモリセルを備えていてもよい。各プロセッサのメモリには、すべて電源投入時に固定値が設定されるような構成にしてもよい。これにより、電源投入時に任意の値が設定される場合にありがちな、プロセッサが恣意的なプログラムを実行してしまうことがなくなる。

【図面の簡単な説明】

【００１５】

【図１】本発明の１つの実施例にかかるプロセッサ・チップの回路図である。

【図２】図１のチップにおける、１つのプロセッサとバスの各セクションを示す概略図である。

【図３】図１のチップのバスのうちの１つの一部の回路図である。

【図４】図１のチップにおいて、バスに沿って、あるいはプロセッサ間を伝達されるデータフレームの略図である。

【図５】図１のチップのプロセッサの１つを示す略図である。

【図６】本発明の第２の実施の形態の一部を構成するバスにおけるアドレッシングを示す略図である。

【発明を実施するための形態】

【００１６】

以下、本発明の好適な実施の形態について、一例として、添付の図面を参照しつつ説明する。

【００１７】

図１に示すように、プロセッサ・チップ１０は、プロセッサ１２の２次元矩形アレイからなる。各プロセッサ、すなわちタプル１２は、直交座標系Ｘ，Ｙによって表されるアドレスを有する。メインアレイは有限であり、Ｙ座標は原点を中心として$-Y_{max}$から$+Y_{max}$まで延び、Ｘ座標もまた原点を中心として$-X_{max}$から$+X_{max}$まで延びている。プロセッサ１２のメインアレイの各行、各列の終端は、出入力プロセッサであり、Ｘ座標が＋∞若しくは－∞、あるいはＹ座標が＋∞若しくは－∞である出入力装置１４が設けられている。図１では、プロセッサ・チップ１０の４分の１だけが、すなわち、座標系における正象限にある部分のみが図示されている。多数のチップを組み合わせることで、出入力装置１４を介して各チップ間でデータが移動する単一の装置を形成する。

【００１８】

＜バス＞

プロセッサ１２の間には、一組のバス２０が矩形の格子状に配置される。各プロセッサ１２の列の間には、Ｙ軸方向に延びる一対のバスが設けられる。一方の＋Ｙは、Ｙ軸の正方向へのデータ転送を、他方の－Ｙは、Ｙ軸の負方向へのデータ転送を行う。プロセッサ１２の各行の間には、Ｘ軸方向に延びる一対のバスが設けられる。一方の＋Ｘは、Ｘ軸の正方向へのデータ転送、他方の－Ｘは、Ｘ軸の負方向へのデータ転送を担う。バス２０の各ペアは、図１では単一の線で表現されているが、図２にはプロセッサ１２の１つを囲む各バス２０の部分が示されている。すなわち、図２は、基本ユニットとなる１つのプロセッサ・タイル２２を示している。この基本ユニットは、チップ１０全体に渡って繰り返されてチップ全体を構成する。各プロセッサ１２は、その４辺において隣接する４つのバス２０のそれぞれと接続されており、したがって、各プロセッサは、データを４方向のうちのいずれかの方向に転送するために、データを適切なバスへと導く。

【００１９】

図３を参照すると、単方向バス２０の各々は、一連のバスフレーム２４と、それに平行な一連のテンポラリフレーム２６とから成る。各バスフレーム２４は、複数のメモリセルによって構成され、各テンポラリフレーム２６も、同数のメモリセルによって構成される。各テンポラリフレーム２６は、隣接する２つのバスフレーム２４に接続されており、一方のバスフレーム２４からデータを受け取り、他方のバスフレーム２４にデータを出力することができるようになっている。したがって、データは、１つのバスフレーム２４からバス２０の方向に沿って次のバスフレーム２４に、適切なテンポラリフレーム２６を介して転送されることによって伝達される。プロセッサ１２は、その近傍を通過するバス２０の各々のバスフレーム２４の１つと接続されており、その地点において、当該バスからのデータを受け取り、また、当該バスへのデータ書き込むことができるようになっている。

【００２０】

全てのプロセッサ１２、全てのバスフレーム２４およびテンポラリフレーム２６は、クロック信号が伝達される共通のクロックライン２８に接続されている。クロックライン２８は、バス２０に沿うと共に、バス２０とプロセッサ１２との間におけるデータ転送のタイミングを調整するために用いられる。クロックが刻まれる毎に、バスフレーム２４のデータは、テンポラリバスフレーム２６を介して隣接するバスフレーム２４にコピーされる。一般に、バスフレーム間のデータ移動は、各プロセッサがインストラクションを実行する頻度より高い頻度で発生する。そのため、プロセッサは、１つのプロセッサクロックサイクルにおいて、バスに沿って一つ以上のデータを伝達できる。製作誤差を防ぎながら、プロセッサは、両側に隣接タプルを有し、出入力タプルは、片側にのみ隣接タプルを有する。

【００２１】

＜演算＞

この装置は、修正２の補数エンコーディングにおいて実行される固定小数点数演算を用いる。標準的な２の補数演算は、０用の１つのビット列と、連続する正の整数をコード化する奇数個のビット列と、連続する負の整数をコード化する偶数個のビット列とを有する。負の整数のビット列は、正の整数のビット列よりも１ビット多い。標準演算では、オー

バーフロー時は、ステータスフラッグがセットされる。対照的に、本実施の形態では、無効を排除して、０のいずれか一方の側に位置する同数の奇数個のビット列を用いたコード化を発生させつつ、無効（nullity）Φを最上位負整数のビット列と同一であるとする修正２の補数演算が用いられる。符号付き無限大（±∞）は、無効及び符号付き無限大を排除し、０のいずれか一方の側に位置する同数の偶数個の連続（有限）整数を残しつつ、残りの最上位正整数及び最上位負整数用のビット列と同一視される。オーバーフロー時には、演算は、符号付き無限大を丸める。この整数の基礎コード化の下では、数は、固定小数点形式ｉ．ｆで表される。但し、iは、整数ビットであり、ｆは、小数点以下部分ビットである。今述べたように、iビットは、修正２の補数エンコーディングを用いた、符号、無限大、及び無効を表すビットパターンを含む。すなわち、これは、整数部分と小数点以下部分が同じビット数である場合、小数点以下部分は、整数部分より正確であることを意味する。通常、数は、符号が付いていることを明確にするために、±（ｉ．ｆ）の形式で記載される。本実施の形態で使われる修正２の補数演算の詳細は、ＧＢ０６２５７３５．６に開示されている。無効の定義は、以下の原理による。すなわち、無効は、無限大から無限大を減算した結果であり、無効は、無限大に０を乗算した結果であり、任意の数を無効に加算した結果は無効であり、任意の数に無効を乗算した結果は無効である。

【００２２】

＜データ・フォーマット＞

図４を参照すると、バスは、ビットグループにあるデータや情報をトークンの形式で伝達する。各トークンは、３つのフィールドから成る。すなわち、整数ビットｉと小数点以下部分ビットｆを含む第１グループのビットからなるデータフィールドと、整数ビットｉと小数点以下部分ビットｆを含む第２グループのビットからなるアドレスフィールドと、トークンのステータスを様々な方法で示すタグとして用いられるｃ、ｄ、a_1、a_2の４つのビットのグループからなるタグフィールドとである。各トークンは、以下に説明されるようにｃとｄのタグで特定される、コントロールトークンとデータトークンの２種類に分かれる。

【００２３】

データトークンは、バイナリ許容出力数に対するポテンシャルを与えつつ、１のゼネラルアドレス又は２つのアドレスを定義する±（ｉ．ｆ）の形式のアドレスフィールドを有する。また、データトークンは、さらに、書き込まれる予定のデータである１つの数±（ｉ．ｆ）を有する。コントロールトークンは、制御のバイナリ許容出力数に対するポテンシャルを与えつつ、１のゼネラルアドレス又は２つのアドレスを定義する±（ｉ．ｆ）の形式のアドレスフィールドを有する。

【００２４】

タグビットには、ｃ、ｄ、a_1、a_2の４つがある。ｃビットは、カレントバスフレームにコントロールトークンが含まれているか否かを示す。ｄビットは、カレントバスフレームにデータトークンが含まれているか否かを示す。本実施の形態において、２つのビットが特定され、故に、コントロールトークンとデータトークンとが同じプロセッサに送られることを単一のトークンで表示することができる。a_1ビットは、トークンがアドレスa_1に送られるか否か、あるいは送られているか否かを示す。同様に、a_2ビットは、当該トークンがアドレスa_2に送られるか否か、あるいは送られているか否かを示す。

【００２５】

図５は、簡素化のために３つのｉビットと３つのｆビットを示すが、本実施の形態では、整数部分に３２ビット、小数点以下部分に３２ビットの６４ビットが用いられる。

【００２６】

＜アドレス指定＞

データフィールドは、それ全体で１つの数として解釈される。アドレスフィールドが、±∞の数又はΦのいずれかである場合、アドレスフィールドは、単一の第１アドレスｉとして解釈される。それ以外の場合、アドレスフィールドは、ｉビットによって定義される第１アドレスa_1と、ｆビットによって定義される第２アドレスa_2という２つのアドレ

スとして解釈される。タグフィールドは、４つのビットを表し、４ビットの各々は、セットされたり、あるいはクリアである。ｃタグが設定される場合、データフレームは、制御を運ぶ。それ以外は、ｃタグは制御を運んでいない。制御を運んでいるデータフレームは、コントロールトークンと呼ばれる。ｄタグが設定される場合、データフレームは、データを運ぶ。それ以外は、ｄタグは、データを運んでいない。データを運んでいるデータフレームは、データトークンと呼ばれる。a_1タグが設定される場合、データフレームは、±∞の第１アドレスa_1、あるいはゼネラルアドレスｉに依然として伝送される。それ以外の場合には、このアドレスに伝送されない。a_2タグが設定される場合、データフレームは、第２アドレスa_2に依然として伝送される。そうでない場合は、もはやこのアドレスには伝送されない。アドレスフィールドが無効Φであれば、データフレームは、バスには乗っていない。アドレスa_1、a_2は、同じプロセッサをターゲットとしても、あるいは、別々のプロセッサをターゲットとしてもよい。２つの別々のアドレスを用いることで、単一のスレッドから２つの平行スレッドに許容出力するよう制御することが可能になる。対象となるプロセッサが異なる場合、データフレームは、第２アドレスa_2の前に第１アドレスa_1に送られることになる。a_1タグとa_2タグとが両方クリアである場合、データフレームは、空であり、プロセッサによる書き込まれる。

【００２７】

＜バスとのタプルの接続＞

上述したように、各タプル１２は、左右上下の４つの線条バス２０に重ね合わせられるように接続されている。図３は、タプル１２とゼネラル、アップアドレス、又はダウンアドレスの線条バスとの接続を示す。本実施の形態のチップでは、４つの異なるバスが各タプルに接続されているが、他の実施の形態では、共有のために適切な空間的・時間的トレードオフがある場合には、隣接するタプル同士の間でバスを共有してもよい。チップのすべてのバスは、集合的に「バス」と称される。

【００２８】

＜出入力タプル＞

上述したように、タプル１２は、見かけ上矩形の行列のアレイに配置される。各タプルは、左右上下４つの線条バスに重ね合わせられている。１の線条バスにおける最初と最後のタプルは、アドレス±∞とされて出入力を司る。一方、中間のタプルは、プロセッサとなる。オンチップ出入力装置は、トークンを関連するバスに伝達したり、出入力タプルにおいて関連するバスからトークンを外したりする。出入力タプルが、周辺機器からチップへと入力される対象のトークンである場合には、タプルは、反対側の線状バス上にある出力装置にトークンを書き込む。これにより、出入力タプルの接続テストが可能となる。トークンが外側境界バス上の出入力タプルに到着した場合、そのトークンは、出力装置に書き込まれる。もし、トークンがその出入力タプルをターゲットとしていない場合、そのトークンが先のターゲットによって捕捉されなかったために、トークンは到着する。すなわち、ハードウェアやコンパイラのエラーである。オフチッププロセッサは、このエラーを検証してもよい。出入力タプルがコントロールトークンを捕捉する動きは、アーキテクチャに依存するので、出入力処理の条件付けに用いることもできる。本実施の形態のチップにおいて、出入力タプルでコントロールトークンは用いられないが、出入力タプルは、チップ内のある場所にトークンを書き込んで、何らかの条件を報告してもよい。

【００２９】

無限大は、最も極端な数であり、無限大のプロセッサの０番目、ｕのセルを超えた任意のメモリセルにアドレスすることは可能ではない。無限大タプルは、常に実際に番号が付されたプロセッサを超えたところに位置する。したがって、線条バスにおけるプロセッサの個数、それ故にチップの形状は、チップ内のトークンの伝播に影響を及ぼすかもしれないが、チップの出入力には影響を与えない。これは、チップが、もともと製造形状が非矩形であったり、チップ内での製造誤差により、チップが矩形でない場合に有効である。

【００３０】

＜プロセッサ＞

図５を参照すると、各プロセッサ１２は、８つの物理メモリセルｕ、ｖ、ｗ、ｒ、ｌ、ｚ、ｇ、ｎからなる８－タプルから成る。セルは、０から７まで番号が付いている。したがって、ｕは、タプルの０番目のエレメントとして認められ、ｎは、７番目のエレメントとして認められる。０から番号付けすることは、目標のターゲットを検出するためのアドレスをマスキングする際のハードウェアにおいて行われているように、モジュール演算を行う上で有益である。物理８－タプルは、マニピュレータ又は出入力装置１２でもあるプロセッサ１２によって操作されるデータをホールドする。いずれの種類の装置も、隣接する４つの線条バスのいずれかに書き込みすることができる。物理タプルは、ラベル－ｘ、＋ｘ、－ｙ、+ｙのバーチャルセルで書き込みアドレスを受け取ことによって、ラベルされたバスに書き込みをするように調整されている。このように、タプルの各物理セルに対応する４つのバーチャルメモリセルが存在し、全体で以下の３２のバーチャルセルを提供する。

（u_{-x}、u_{+x}、u_{-y}、u_{+y}、v_{-x}、v_{+x}、v_{-y}、v_{+y}、w_{-x}、w_{+x}、w_{-y}、w_{+y}、r_{-x}、r_{+x}、r_{-y}、r_{+y}、l_{-x}、l_{+x}、l_{-y}、l_{+y}、z_{-x}、z_{+x}、z_{-y}、z_{+y}、g_{-x}、g_{+x}、g_{-y}、g_{+y}、n_{-x}、n_{+x}、n_{-y}、n_{+y}）

【００３１】

確認するが、エレメントは、０から番号付けされているため、u_{-x}は、バーチャル３２－タプルの０番目のエレメントであり、n_{+y}は、３１番目のエレメントである。本実施の形態においては、これらのセルのアドレスは、アドレスの最下位５ビットによって表され、プロセッサのアドレスは、上位ビットによって表される。セルu_{-x}は、ゼロであり、（００００００）とラベル付けされ、残りのセルは、１から３１まで（０００００１）から（１１１１１）まで順に増えるようにラベル付けされている。各プロセッサのバーチャルメモリセルの各々は、それ自身のアドレスを有し、バスにある他のプロセッサから、あるいは同じプロセッサの内部からのデータによってターゲットとされる。

【００３２】

各プロセッサは、下記のインストラクションを実行するように配置されている。

ｕ×ｖ+ｗ→ｒ’
ｗｒｉｔｅ（ｒ’，ｒ）
ｊｕｍｐ（ｒ’，ｌ，ｚ，ｇ，ｎ）

【００３３】

一行目のインストラクションは、トランスリアル乗算及び加算である。したがって、プロセッサは、セルｕ、ｖにおける数の乗算を実行する乗算器５０と、乗算器の出力をセルｗの数に加算する加算器５２とを有する。なお、乗算器の出力は、プロセッサ内に一時的な変数としてホールドされる。この行では、加算、減算、乗算いずれの組み合わせも計算することができる。除算は、逆数を作るインストラクションを用い、その逆数を乗算することによって実行される。同様に、数学の関数や一般的な計算は、多くのインストラクションの中で実行される。

【００３４】

２行目のインストラクションは、データトークンを正しい線条バスに置くことで、その演算結果ｒ’を、１つまたは２つのタプルの１つまたは２つのメモリセルへの書き込む。アドレスｒが、バーチャルレジスタr_{-x}に書き込まれた場合、バス－Ｘにｒ’が書き込まれる。そして、これに対応して、r_{+x}、r_{-y}、r_{+y}でｒが受け取られると、バス+Ｘ、－Ｙ、+Ｙにｒ’へと書き込まれる。これは、加算器５２から出力を受け取り、その出力をデータトークンにするルータ５３によって実現される。ルータ５３は、また、バーチャルセルr_iからアドレスｒを受け取り、アドレスｒをデータトークンのアドレスフィールドに置き、そして、データトークンを適切なバスに乗せる。物理メモリセルｌ、ｚ、ｇ、ｗは、同様に扱われる。しかし、物理セルｕ、ｖ、ｗは、演算用のデータであり、到達したバーチャルメモリセルによってその行動を変えることはない。しかしながら、これらは、今後の使用のためにバーチャルアドレスを維持するu_{-x}、v_{-x}、w_{-x}としてアドレス指定されるべきである。

【００３５】
ジャンプ（ｊｕｍｐ）インストラクションは、バス上にコントロールトークンを乗せる。加算器５２からの演算結果ｒ’は、４つのセレクタ５５、５７、５９、６１に入力される。ｒ’が０以下の場合、最初のセレクタが反応し、ｒ’が０の場合は、２つ目のセレクタが、ｒ’が０以上の場合は、３つ目のセレクタが、ｒ’が無効の場合は、４つ目のセレクタが反応する。トリガーされたセレクタは、それぞれ対応するルータ５４、５６、５８、６０の動作をトリガする。ルータは、バーチャルセルl_i、z_i、g_i、n_iからアドレスを取り出し、取り出したアドレスをコントロールトークンへのアドレスとし、適切なバスにコントロールトークンを乗せる。
【００３６】
ジャンプインストラクションがトークンをバス上に置く前に、書き込みインストラクションがバスにトークンを置く。したがって、データとコントロールトークンが同じタプルに渡されるとき、データトークンは、コントロールトークンの前に到着する。この時空間トポロジにより、インストラクションを用いてメモリロックキングアルゴリズムを実行することが可能となる。ハードウェアにおいて、タプルをバスにリンクさせる以外に、タイミング制御を行う必要はない。このリンクの正確な性質は、チップの性能に重要である。
【００３７】
ジャンプインストラクションは、中止したり、シリアルスレッドを継続したり、あるいは２つの並行するスレッドに分岐することも可能である。スレッドは、Φプロセッサにジャンプしたとき、中止される。これは、無効プロセッサは決して実行されないことになる。さらに、タプルアーキテクチャは、フェッチレス（すなわち読み込まない）なので、無効プロセッサに書き込みをすることはできず、したがって、無効プロセッサはメモリを必要としないことになる。無効プロセッサは、プロセッシングを行わず、メモリも有していないため、タプルとして実行される必要がない。無効プロセッサは、コントロールジャンプや書き込み先のアドレスに指定することはできるが、ジャンプや書き込みのソースにおいては、無演算命令として実行される。
【００３８】
無効プロセッサを排除することによる有用な予期せぬ結果は、トランスリアル面から無効で行を排除することである。これにより、トポロジは簡素化され、無限大に方向が向けられたラインと共に延在する実数平面になる。無限大でのラインは、全プロセッサが実数平面上に存在しながら、出入力に用いられる。
【００３９】
ジャンプインストラクションは、次のように実行される。
Ｊｕｍｐ ｔｏ（ｌ）ｉｆ ｒ’＜０
Ｊｕｍｐ ｔｏ（ｚ）ｉｆ ｒ’＝０
Ｊｕｍｐ ｔｏ（ｇ）ｉｆ ｒ’＞０
Ｊｕｍｐ ｔｏ（ｎ）ｉｆ ｒ’＝Φ
【００４０】
プロセッサは、コントロールトークンを適切な線状バスに置くことによって、ジャンプインストラクションを実行する。従って、コントロールトークンは、ｌ、ｚ、ｇ、ｎ＝±（a_1・a_2）のうち、アドレスa_1とa_2に運ばれる。
【００４１】
プロセッサは、バスからトークンを受け取るバッファを有する。プロセッサは、演算をするとき、バッファを内部レジスタにコピーして、内部レジスタ上で動作する。
【００４２】
＜バスでのトークンの取り扱い＞
上述したように、各プロセッサは、プロセッサへの３２のアドレスを示すために確保された５アドレスビットを有するアドレスＰを有る。プロセッサにデータフレームが到着すると、データフレームは検査される。最初に、Ｐがｉと合致し、a_1がセットされ、ｄもセットされる場合、データフィールドは、バスからプロセッサに書き込まれ、a_1がクリ

アされ、このアドレスへの伝達は、もはや必要なくなったことを示す。次に、第二に、Ｐがｆと合致し、a_1がクリアで、a_2がセットされ、ｄがセットされている場合、データフィールドは、バスからプロセッサへとに書き込まれ、a_2とｄがクリアされる。これは、伝達がどこに対しても不要となったことを示す。第三に、Ｐがｉと合致し、a_1がセットされ、ｃがセットされている場合には、単一サイクルのプロセッサの実行が開始され、a_1がクリアされる。これは、このアドレスへの伝達はもはや必要なくなったことを示す。次に、第四に、Ｐがｆと合致し、a_1がクリアでa_2がセットされ、ｃがセットされている場合、単一サイクルのプロセッサの実行が開始されてa_2がクリアされ、ｃもクリアされる。これは、どこへも伝達が不要となったことを示す。注意すべきは、ｉとｆが同じプロセッサで実行を開始する度に、単一サイクルのみのプロセッサの実行が開始される。第五に、Ｐがｆと合致し、a_1がセットされている場合、第１のアドレスへの伝達が失敗する。これは、エラーである。データは、プロセッサへ書き込まれず、実行は開始されない。データフレームは、データをどこにも伝達させずに、バスに沿って終点まで通過する。

【００４３】

バス上のすべてのプロセッサが、バス上の対応するデータフレームに書き込む機会を持ってしまうと、データフレームは、バス上で位置をひとつ移動させられる。好ましい実施の形態においては、これは、データフレームをテンポラリデータフレームにコピーし、それを隣のデータフレームにコピーすることによって実現される。

【００４４】

＜出入力装置の動作＞

バスの終点にあるデータフレームに、ｃあるいはｄのいずれか一方がセットされていた場合、出入力装置によって、データフレームは、チップの外部に書き込まれる。単一のアドレスがアップアドレスバスで∞、あるいは、ダウンアドレスバスで－∞の場合、そのバスフレームは、出入力装置を正しく目標とし、オフチップデバイスによって、有効データフレームとして扱われる。他のアドレスは、伝達エラーを示し、オフチップデバイスによって適切なエラー処理が施される。

【００４５】

＜プロセッサのバスへの接続動作＞

タプルは、その位置においてバスフレームからトークンを受け取ったり、バスフレームへトークンを書き込んだりする。タプルは、そのバスフレームに書き込みをする前にバスフレームからトークンを受け取る。そのため、バスフレームを再利用することができる。これにより、バスの帯域幅を効率的に利用できることになる。また、孤立したチップ内における通信が隣接するタプル間での移動に限定されているとき、バスは、常にトークンを受け渡し可能な状態にあることになる。バスのこの準備は、たとえば、右と下向きの線条バスを隣接するタプルへの短い書き込み用とし、左と上向きのバスの領域においてのみ長い書き込みや隣接していないタプルへのジャンプを行うことによって、広範囲での応用が可能になる。長いジャンプは、バスの容量を越えないような密度で維持される必要がある。チップ上のどこにおいても迅速な通信を可能とするように、チップ内の領域で長短の配置の間での切替は可能である。

【００４６】

各プロセッサタプルは、バスを使わずに自身の内部に書き込みしたり、ジャンプしたりできる。この場合、書き込みやジャンプの時間は、標準インストラクション時間に含まれており、プロセッサは、トークンを読むよりも速くトークンを書き込みをすることはできない。

【００４７】

トークンは、バスからタプルのバッファへ任意の順番で伝達される。これにより、プロセッサ・バスコミュニケーションの任意のマルチプレクシングが可能となる。しかし、プロセッサがビジーでトークンを受け取れないとき、トークンは、出入力タプルによってチップを外れて書き込まれるまで、バス上に置かれたままになる。このようにして、バスの

競合エラーは、自己報告される。ここでのバスの競合とは、コンパイラあるいはハードウェアのエラーであり、発生してならないものである。同様に、トークンがa_2に到着してもa_1に伝達されない場合、トークンは、バスに沿って伝達され、エラーが自動的に報告される。このため、トークンがa_1に届けられる前にa_2に伝達されず、a_2におけるデータとコントロールは、a_1への伝達を確認するために使用される。正しく実行するためにタイミングルールを利用することは、コンパイラの責任である。これは、バスのローカルエリアでのタイミングを判別することによるコンパイル時間で、あるいは、メモリロッキングアルゴリズムを実施するランタイムで、そのように行ってもよい。

【００４８】

＜バスパワーマネジメント＞

コントロールタグｃとデータタグｄがクリアならば、バスフレームは、コピーされていない。コントロールタグがセットされているが、データタグがクリアな場合は、タグ及びコントロールナンバ全体がコピーされる。データタグがセットされている場合、バスフレーム全体がコピーされる。このように、有効なデータのみを移動させるために、実質的なパワーが使用される。

【００４９】

＜プロセッサによるデータの取り扱い＞

プロセッサのメモリセルu_iのいずれかをターゲットとするバスフレームに、タグｄがセットされている場合、フレームのデータフィールドが、乗算器のメモリセルｕに書き込まれる。同様に、メモリセルv_iのいずれかをターゲットとするデータフィールドは、乗算器のメモリセルｖに書き込まれ、メモリセルw_iのいずれかをターゲットとするデータフィールドは、加算器のフィールドｗに書き込まれる。同様に、メモリセルr_iのいずれかをターゲットとするデータフィールドは、ルータのデータフレームアドレスフィールドに書き込まれ、該ルータが付加的な動作を実行する。データフィールドがr_{-x}をターゲットとする場合、バス－Ｘが出力先として選択される。同様に、データフィールドがr_{+x}、r_{-y}、r_{+y}をターゲットとしている場合には、対応するバス－Ｘ、－Ｙ、＋Ｙが出力先として選択される。

【００５０】

メモリセルl_iのすべてがルータ５４に入力し、メモリセルz_i、g_i、n_iは、それぞれのルータ５６、５８、６０に入力する。すべてのルータは、同じ様に動作する。例えば、プロセッサのメモリセルl_iのいずれかをターゲットとするデータフレームのタグｄがセットされている場合は、フレームのデータフィールドは、ルータのアドレスフィールドに書き込まれる。データフィールドがl_{-x}、をターゲットにしている場合は、－Ｘバスが出力先として選択される。同様に、データフィールドがl_{+x}、l_{-y}、l_{+y}をターゲットにしている場合には、対応するバス－Ｘ、－Ｙ、＋Ｙが出力先として選択される。データフレームのタグフィールドは、第１と第２のアドレスへの制御の伝達を示すように設定される。

【００５１】

プロセッサでの実行は、タグｃがセットされたデータフレーム、すなわち、コントロールトークンによって開始され、プロセッサの任意のバーチャルメモリセルをターゲットとする。バーチャルメモリセルに関連するアドレスビットを無視することは、単にプロセッサのアドレスＰが用いられることを意味するが、しかし、これは、プロセッサのメモリセルのアドレスu_{-x}に等しい。図５は、メモリセルu_{-x}にコントロールトークンが到着してトリガされる実行を示す。このコントロールトークンが到着すると、乗算器は、自身のメモリセルｕとｖとを乗算し、その積を加算器に書き込む。加算器は、その積にセルｗのコンテンツを加算する。そして、その結果である和は、データフレームのデータフィールドに書き込まれる。データフレームのタグフィールドは、第１と第２のアドレスへデータへの伝達を示すように設定される。アドレスが無効でなければ、選択された出力バスにデータフレームが書き込まれる。アドレスが無効であれば、データフレームは、バスに置かれない。また、加算器からの結果としての和は、４つのセレクタのそれぞれにも書き込まれ

、和が０より小さいか、０に等しいか、０より大きいか、あるいは無効であるかによって、いずれか１つのルータがトリガされる。トリガされたルータは、アドレスが無効でなければ、選択された出力バスにデータフレームを書き込む。アドレスが無効の場合には、データフレームはバスには置かれない。この書き込みは、メモリセルをターゲットとするルータからのデータの書き込みの後に行われるように、タイミングが設定されている。

【００５２】

すべてのプロセッサのタイミングは、共通のクロック信号によって制御されている。このクロック信号は、バスの制御に使用されるものと同じでよい。プロセッサは、クロック信号に応答し、ワンサイクル毎に一度インストラクションを実行するよう構成されている。そして、全プロセッサは、同じタイミング動作するので、各プロセッサは、同時にデータをバスに置く。各サイクルにおいて、データがバスからプロセッサに書き込まれるタイミングは、プロセッサをアドレスとするデータがそのプロセッサに隣接するバスにある時間に依存する。データは、インストラクションが実行されるより頻繁にバスに沿って移動するため、プロセッサにデータが書き込まれる時間は、プロセッサ毎に異なる。

【００５３】

＜記号＞

好ましい実施の形態の記載において用いられる記号を以下にまとめる。

-x：デカルト座標系の原点から負のＸ軸を示す下付き添字。

-x：デカルト座標系の原点から正のＸ軸を示す下付き添字。

-y：デカルト座標系の原点から負のＹ軸を示す下付き添字。

+y：デカルト座標系の原点から正のＹ軸を示す下付き添字。

a_1：±（a_1・a_2）の形式で最初に現れるアドレス。

a_1：トークンがアドレスa_1に伝達されるべきなのか、あるいはすでに伝達されたのかを示すバスフレームのタグビット。

a_2：±（a_1・a_2）の形式で２番目に現れるアドレス。

a_2：トークンがアドレスa_2に伝達されるべきなのか、あるいはすでに伝達されたのかを示すバスフレームのタグビット。

ｃ：フレームが制御を含むか否かを示すバスフレームのタグビット。

ｄ：フレームがデータを含むか否かを示すバスフレームのタグビット。

ｆ：固定小数点の小数点以下部分ビット。

ｇ：物理８－タプルの６番目のセル、０より大きい結果の場合にジャンプするアドレス。

ｉ：固定小数点の整数ビット、符号、無限大及び無効を示すビットパターンを含む。

ｌ：物理８－タプルの４番目のセル、結果が０より小さい場合にジャンプするアドレス。

ｎ：物理８－タプルの７番目のセル、結果が無効の場合にジャンプするアドレス。

Ｐ：プロセッサのアドレス。これは、物理８－タプルの０番目のセルｕのアドレスである。

ｒ、ｒ’：物理８－タプルの３番目のセル。インストラクション・フラグメントｕ×ｖ+ｗ→ｒ’の演算結果のアドレス。演算結果は、一時変数ｒ’にホールドされる。

ｕ：物理８－タプルの０番目のセル。インストラクション・フラグメントｕ×ｖ+ｗ→ｒ’の第１の引数。

ｖ：物理８－タプルの１番目のセル。インストラクション・フラグメントｕ×ｖ+ｗ→ｒ’の第２の引数。

ｗ：物理８－タプルの２番目のセル。インストラクション・フラグメントｕ×ｖ+ｗ→ｒ’の第３の引数。

ｚ：物理８－タプルの５番目のセル。解が０の場合にジャンプするアドレス。

【００５４】

＜効果＞

上述した実施の形態には、数多くの効果がある。

【００５５】
チップの周辺に、どこでも出入力及び電源供給ができる。したがって、出入力及び電源供給のいずれにおいても、膨大な帯域幅及び冗長性がある。しかし、冗長電源供給は、不要なチャージフローや電気的ノイズを防ぐよう注意して操作する必要がある。それでもなお、この帯域幅及び冗長性は、ある程度の将来の保証をもたらす。
【００５６】
周辺のどこででも出入力が可能という取り組みは、トークンが捉まらなかった場合、未捕捉トークンを調べるようプログラムされた出力装置に書き込まれたトークンによって、このエラーが自動的に報告されることを意味する。
【００５７】
プロセッサ・インストラクションは、２のべき乗の任意の長さのタプルにも拡張可能であり、したがって任意の複雑なインストラクションのセットも実行できる。このことは、ある程度の将来の保証をもたらす。
【００５８】
物理アドレスｕ、ｖ、ｗのバーチャルバージョンに関連する冗長ビットが６つあり、プロセッサに異なった動作をさせるよう条件づけるために使用される。プロセッサのアーキテクチャやコンパイラのモジュールを変えるだけで柔軟性が生まれ、繰り返しになるが、ある程度の将来の保証がもたらされる。
【００５９】
上述した実施の形態においては、トランスナンバは、ビット列によって表される。厳密には、トランスリアル、すなわち、±∞又はΦは、ビット列全体を使用するが、実数は、ｉ．ｆという２つの部分で表現される。ここで、ｉは、数の整数部分であり、ｆは、小数点以下部分である。アドレス指定のスキームによって、０、１、または、２つの対象への指定が可能である。アドレスが無効Φである場合、データフレームは、バスに置かれず、したがってターゲットとして指定されるアドレスもない。アドレスが符号付き無限大±∞のうちの一方である場合には、１つの出入力装置がアドレス指定される。アドレスが実数の場合、ｉは第１のアドレス、ｆは第２のアドレスとして解釈される。一般に、ｉとｆは、異なるプロセッサのメモリセルをターゲットとしており、２つのターゲットがアドレスとして指定される。しかし、単一のプロセッサ内の同一のメモリセル、又は異なるメモリセルにアドレス指定してもよい。この場合、１つのプロセッサ、あるいは１つのメモリセルが、アドレス指定される。このように、ターゲットなし、１つの出入力装置、１つのプロセッサ内の１つまたは２つのメモリセル、あるいは、２つの異なるプロセッサの２つのメモリセルをアドレスとして指定できる。これで全体的には十分であるが、ｉは符号ビットを含むが、ｆには符号ビットがないことが課題である。したがって、すべての正のアドレスと、負の第１アドレスについては自然な表現があるが、負の第２アドレスには自然な表現がない。
【００６０】
負のアドレスに対して自然な表現がないという問題は、正のアドレスのみを用いる適宜のスキームにおいて解決される。このようなスキームのうち最も簡単なものは、２次元のデカルト座標系の第一象限に配置された線や格子を使用することである。図１に示される座標系の一部がこれに相当する。しかし、この場合、各チップの結合が限定的になってしまう。この場合、チップは、正の軸方向にそって足すことはできるが、負の軸方向に沿って足すことはできない。これは、装置が利用できるスペースを制限する。
【００６１】
したがって、図６を参照すると、本発明の第２の実施の形態では、別の解決策が用いられている。ここで提案される解決策では、各バスを、入力装置に隣接するプロセッサのメモリセルu_{-x}で０からの連続自然数、チップの端部では－∞が付され、出力プロセッサに近接するメモリセルn_{+y}で正整数ｎとなり、チップの端部では＋∞が付されるように、連続した自然数で番号付けする。このように、各メモリセルは、各バスにおいて異なるアドレスを有し、アドレスは、簡単な方法で互いに関連付けられている。さらには、アド

レスは、チップ何個分も離れた場所にあるプロセッサ内の１つのメモリセルをターゲットしてアドレス指定できる。もっとも重要なことは、すべてのアドレスの計算は、トランス算術において自然な方法で実行される。

【００６２】

各バスは、－∞と番号付けされた入力装置で入り口を、＋∞と番号付けされた出力装置で出口を有する。プロセッサの内部に介在するメモリセルには、図６に示されるように、０から正の数ｎまで順番に番号がふられている。このように、単一のメモリセルは、一般に、各バスに異なるアドレスを有する。

【００６３】

１つのバスのメモリセルのリアルアドレスｃを、対向するバスのアドレスｃ’に変換するために、シンプルなアルゴリズム、

ｎ－ｃ→ｃ’

が実行される。この演算は、冪等であり、故に

ｎ－ｃ’→ｃ

である。

【００６４】

厳密には、トランスリアルアドレスは、そのままで正しい。トランスリアルアドレスは、反対側のバスのアドレスに写像する適宜の手段によって変換される必要はない。もし、リアルアドレスｃがｎより大きい場合、ターゲットとなるメモリセルは、別のチップの上にある。そこへアドレス指定されたトークンは、現在のチップからバスに沿って、＋∞が付けられた出力装置に運ばれる。出力装置は、現在のチップの幅分、すなわちｎ＋１分をそのアドレスからディクリメントして、トークンを出力する。従って、このトークンは、次のチップに置かれる。次のチップでは、アドレスは、そのチップでのアドレスに相当する十分に小さい数であるか、あるいは、そこへ伝送される場合は、大きすぎる数であるか、のいずれかである。この場合、トークンは、次のチップを横切ってとなりのチップの出力装置に向けて送られる。ここで、再び、アドレスはディクリメントされ、トークンは、更に次の隣接チップに置かれる。このプロセスは、トークンが適切なチップのプロセッサに伝達されるまで、何度でも繰り返えされる。この構成においては、各チップは、全プロセッサ用のアドレスを有し、また同じチップについては、対応するプロセッサのアドレスは同じであることが望ましい。しかし、トークンによって運ばれるようなターゲットのアドレスは、大きくてもよい。実際、このアドレスは「相対的」なものであり、トークンの現在の位置に関係した位置によって、ターゲットであるプロセッサを特定する。

【００６５】

本システムにおいては、各チップは、±∞が付けられたそれぞれの出入力装置を有している。これは、±∞までに多くのルートが存在することを示している。実数がつけられた負のアドレスは存在しない。そのようなアドレスは、特定のシステムの特定のデザインに依存する何らかのものをコード化するために用いられることになる。このようなアドレスは、対応する正のアドレスの２の補数になる。

【００６６】

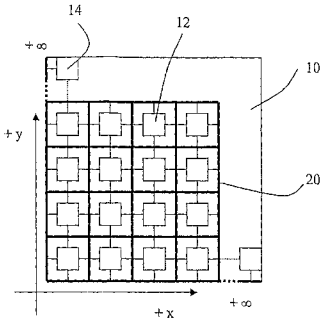
チップに不具合があったり、形状が矩形ではない場合、異なるチップのバス上には異なる個数の有効なプロセッサが存在してもよい。したがって、各チップは、それ自身のデクリメントを行う必要がある。このデクリメントは、オフチップ装置にさせてもよい。チップが垂直方向に積層している場合、オフチップ装置は、ターゲットのチップまで迅速に信号を送ればよい。こうしたショートカットは、コンパイラが利用するタイミングルールに影響を及ぼすことがある。

【符号の説明】

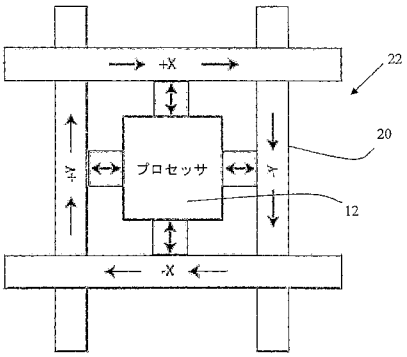
【００６７】

１０　プロセッサ・チップ
１２　プロセッサ
２０　バス

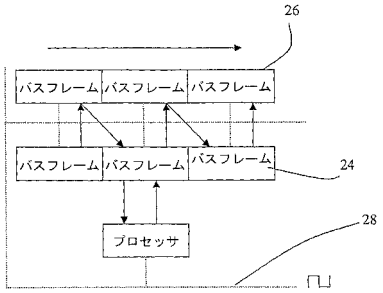
【図１】



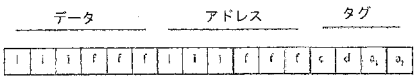
【図２】



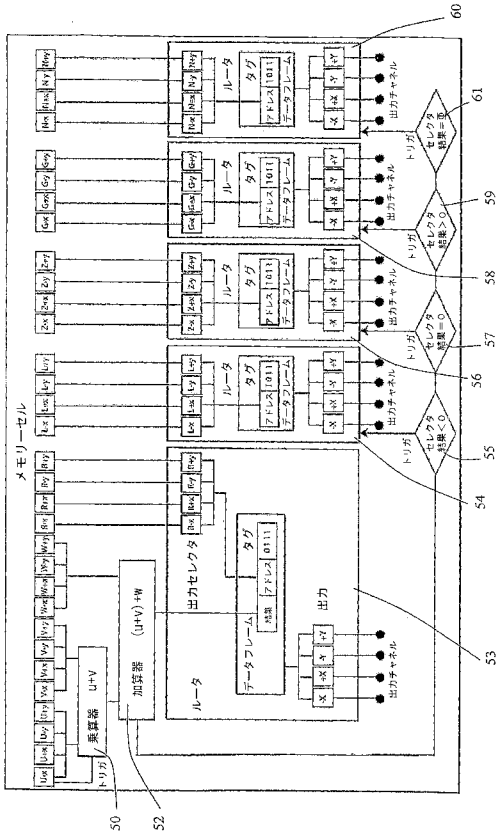
【図３】



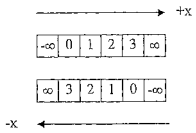
【図４】



【図５】



【図６】



フロントページの続き

(74)代理人 100158023
弁理士 牛田 竜太
(72)発明者 アンダーソン、ジェームス、アーサー、ディーン、ワレス
イギリス、レディング アールジー４ ５エルイー、カバシャム、ローアー ヘンリー ロード ８８番地

審査官 田中 幸雄

(56)参考文献 Shlomit WEISS et al., Architectural Improvements for a Data-Driven VLSI Processing Array, Journal of Parallel and Distributed Computing, 米国, Academic Press, １９９３年, vol. 19, no. 4, pages 308-322
V. G. GRAFE et al., Implementation of the εpsilon Dataflow Processor, Proceedings of the 17th Annual International Symposium on Computer Architecture, 米国, IEEE, １９９０年, pages 19-29

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ １５／８２
Ｇ０６Ｆ １３／３６
Ｇ０６Ｆ １５／１７３