(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5382736号
(P5382736)
(45)発行日　平成26年1月8日(2014.1.8)　(24)登録日　平成25年10月11日(2013.10.11)

(51)Int.Cl.　F I
G06F 15/173 (2006.01)　G06F 15/173 650B

請求項の数 44　(全 40 頁)

(21)出願番号	特願2010-515439 (P2010-515439)
(86)(22)出願日	平成20年5月27日 (2008.5.27)
(65)公表番号	特表2010-532540 (P2010-532540A)
(43)公表日	平成22年10月7日 (2010.10.7)
(86)国際出願番号	PCT/EP2008/056499
(87)国際公開番号	WO2009/007172
(87)国際公開日	平成21年1月15日 (2009.1.15)
審査請求日	平成23年5月26日 (2011.5.26)
(31)優先権主張番号	11/774,224
(32)優先日	平成19年7月6日 (2007.7.6)
(33)優先権主張国	米国 (US)
(31)優先権主張番号	12/027,567
(32)優先日	平成20年2月7日 (2008.2.7)
(33)優先権主張国	米国 (US)

(73)特許権者　509257307
エックスモス　リミテッド
ＸＭＯＳ　ＬＴＤ
イギリス，ビーエス１　４ピービー　ブリストル，キング　ストリート，ヴェンチャーズ　ハウス（番地なし）
(74)代理人　110000202
新樹グローバル・アイピー特許業務法人
(72)発明者　メイ，マイケル　デービット
イギリス，ビーエス８　２イージェイ　ブリストル，クリフトン，イートン　クレセント　９

審査官　清木　泰

最終頁に続く

(54)【発明の名称】トークンプロトコル

(57)【特許請求の範囲】
【請求項１】
プロセッサ間を結ぶリンクであって、ライン１およびライン０を含み、かつ前記ライン１上の論理遷移が論理１を示し、前記ライン０上の論理遷移が論理０を示すリンク上にトークンを送信する方法であって、
トークンの第１の部分を送信することと、
前記トークンの第２の部分を送信して、確実に前記トークンの論理１のビットの総数を偶数、かつ前記トークンの論理０のビット総数を偶数として、前記トークンの末尾で、前記リンクを静止状態に戻すこととを含む方法。
【請求項２】
データトークンを送信するか、制御トークンを送信するか判断することをさらに含み、
前記第１の部分が、データトークンを送信する場合にはデータの運搬、制御トークンを送信する場合には制御情報の運搬に用いられる情報部分と、前記トークンがデータトークンであるか制御トークンであるかを示す第１の追加ビットとを含む、請求項１に記載の方法。
【請求項３】
前記リンクが同一基板またはチップ上にあるプロセッサ間を結んでいる、
請求項１に記載の方法。
【請求項４】
前記第１の部分が、偶数個の情報ビットと前記第１の追加ビットとを含む奇数個のビッ

トを有し、前記第２の部分が第２の追加ビットである、請求項２に記載の方法。
【請求項５】
前記第１の部分が論理１に偶数個のビットを、かつ論理０に奇数個のビットを含んでいるか、または前記第１の部分が論理１に奇数個のビットを、かつ論理０に偶数個のビットを含んでいるかを判断することをさらに含み、
もし前記第１の部分が偶数個の論理１および奇数個の論理０を含んでいれば、前記第２の部分は論理０のビットであり、
もし前記第１の部分が奇数個の論理１および偶数個の論理０を含んでいれば、前記第２の部分は論理１のビットである、請求項４に記載の方法。
【請求項６】
前記第１の部分についてビット毎ＸＯＲ演算を実行して、前記第２の部分を算出することさらに含む、請求項５に記載の方法。
【請求項７】
前記情報部分が８ビットである、請求項２に記載の方法。
【請求項８】
前記トークン内の送信が、前記第１の追加ビット、前記情報部分、そして前記第２の追加ビットの順に行われる、請求項４に記載の方法。
【請求項９】
前記第１の部分が、上記プロセッサのうちの１つで実行されるソフトウェアにより生成され、前記第２の部分が、前記リンク内ハードウェアにより生成される、請求項１に記載の方法。
【請求項１０】
前記第１の部分が、上記プロセッサのうちの１つで実行される命令のオペランドである、請求項９に記載の方法。
【請求項１１】
前記トークンが、アーキテクチャで定義される制御トークンであり、前記方法が、前記制御トークンを使用して前記リンクを含む回路を有するインターコネクト内で論理をトリガして、前記インターコネクトの構成要素をコントロールすることをさらに含む、請求項２に記載の方法。
【請求項１２】
ソフトウェアの機能を遂行するため、宛先プロセッサで実行されるソフトウェアを用いて、前記アーキテクチャで定義された制御トークンを受信することをさらに含む、請求項１１に記載の方法。
【請求項１３】
前記アーキテクチャで定義された制御トークンは、前記宛先プロセッサで実行される特権ソフトウェアにのみアクセス可能な特権制御トークンである、請求項１２に記載の方法。
【請求項１４】
前記トークンが、ソフトウェアに定義される制御トークンである、請求項２に記載の方法。
【請求項１５】
前記トークンが、宛先プロセッサを指定する１つまたは複数のヘッダトークンを含むメッセージで送信される、請求項１に記載の方法。
【請求項１６】
前記トークンが、１アレイ内の２つ以上のプロセッサ間をつなぐ、スイッチおよびリンクのシステムを含む回路を有するインターコネクト上に送信される、請求項１に記載の方法。
【請求項１７】
複数のプロセッサと、ライン１およびライン０を含んで前記プロセッサ間を結ぶリンクとを含み、前記ライン１上の論理遷移が論理１を示し、前記ライン０上の論理遷移が論理

０を示すデバイスであって、
前記プロセッサの少なくとも第１のプロセッサが、トークンの第１の部分を送信し、
前記トークンの第２の部分を送信して、確実に前記トークンの論理１のビット総数を偶数とし、かつ前記トークンの論理０のビット総数を偶数とし、これにより、前記トークンの末尾で、前記リンクを静止状態に戻すように構成されている、デバイス。
【請求項１８】
前記第１のプロセッサが、データトークンを送信するか、制御トークンを送信するかを判断するように構成されており、
前記第１の部分が、データトークンを送信する場合にはデータの運搬、制御トークンを送信する場合には制御情報の運搬に用いられる情報部分と、前記トークンがデータトークンであるか制御トークンであるかを示す第１の追加ビットとを含む、請求項１７に記載のデバイス。
【請求項１９】
前記デバイスが同一基板またはチップ内に位置している、請求項１７に記載のデバイス。
【請求項２０】
前記第１の部分が、偶数個の情報ビットと前記第１の追加ビットとを含む奇数個のビットを有し、
前記第２の部分が第２の追加ビットである、請求項１８に記載のデバイス。
【請求項２１】
前記第１のプロセッサが、前記第１の部分が論理１に偶数個のビットを、かつ論理０に奇数個のビットを含んでいるか、前記第１の部分が論理１に奇数個のビットを、かつ論理０に偶数個のビットを含んでいるかを判断するように構成されており、
もし前記第１の部分が偶数個の論理１および奇数個の論理０を含んでいれば、前記第２の部分は論理０のビットであり、
もし前記第１の部分が奇数個の論理１および偶数個の論理０を含んでいれば、前記第２の部分は論理１のビットである、請求項２０に記載のデバイス。
【請求項２２】
前記第１の部分についてビット毎ＸＯＲ演算を実行して、前記第２の部分を算出するように構成された論理回路をさらに含む、請求項２１に記載のデバイス。
【請求項２３】
前記情報部分が８ビットである、請求項１８に記載のデバイス。
【請求項２４】
前記トークン内の送信が、前記第１の追加ビット、前記情報部分、そして前記第２の追加ビットの順に行われる、請求項２０に記載のデバイス。
【請求項２５】
前記第１の部分が、前記第１のプロセッサで実行されるソフトウェアにより生成され、前記第２の部分が、前記リンク内ハードウェアにより生成される、請求項１７に記載のデバイス。
【請求項２６】
前記第１の部分が、前記第１のプロセッサで実行される命令のオペランドである、請求項２５に記載のデバイス。
【請求項２７】
前記デバイスが前記リンクを含む回路を有するインターコネクトをさらに備え、前記トークンが、アーキテクチャで定義される制御トークンであり、前記インターコネクトが、前記インターコネクトの構成要素をコントロールするために、前記制御トークンによりトリガされるように構成された論理を含む、請求項１８に記載のデバイス。
【請求項２８】
前記プロセッサのうち第２のプロセッサが、ソフトウェアの機能を遂行するため、前記前記第２のプロセッサで実行されるソフトウェアを用いて、前記アーキテクチャで定義さ

れた制御トークンを受信するように構成されている、請求項２７に記載のデバイス。
【請求項２９】
前記アーキテクチャで定義された制御トークンは、前記第２のプロセッサで実行される特権ソフトウェアにのみアクセス可能な特権制御トークンである、
請求項２８に記載のデバイス。
【請求項３０】
前記トークンが、ソフトウェアに定義される制御トークンである、請求項１８に記載のデバイス。
【請求項３１】
前記第１のプロセッサが、宛先プロセッサを指定する１つまたは複数のヘッダトークンを含むメッセージで前記トークンを送信するように構成されている、請求項１７に記載のデバイス。
【請求項３２】
１アレイ内の２つ以上のプロセッサ間をつなぐ、スイッチおよびリンクを含む回路を有するインターコネクトシステムをさらに備える、請求項１７に記載のデバイス。
【請求項３３】
プロセッサ間を結ぶリンクであって、ライン１およびライン０を含み、前記ライン１上の論理遷移が論理１を示し、前記ライン０上の論理遷移が論理０を示すリンク上にトークンを送信するコンピュータプログラムであって、
トークンの第１の部分を送信するステップと、
前記トークンの第２の部分を送信して、確実に前記トークンの論理１のビット総数を偶数、かつ前記トークン内の論理０のビット総数を偶数とし、これにより、前記トークンの末尾で、そのリンクを静止状態に戻すステップと、
をプロセッサに実行させる、プログラム。
【請求項３４】
複数の処理手段と、前記処理手段間をリンクし、かつ論理遷移により論理１を示す論理１送信手段と、論理遷移により論理０を示す論理０送信手段とを含むリンク手段とを含むデバイスであって、前記処理手段の少なくとも第１の処理手段が、トークンの第１の部分を送信する送信手段であって、前記トークンの第２の部分を送信して、確実に前記トークンの論理１のビット総数を偶数、かつそのトークン内の論理０のビット総数を偶数とし、これにより、前記トークンの末尾で、そのリンクを静止状態に戻す送信手段を含むデバイス。
【請求項３５】
プロセッサ間を結ぶリンク上に１つまたは複数のトークンを送信する方法であって、
前記リンクは少なくとも３本のラインを含み、
前記ラインは、複数のコードのうちの各コードを伝送するために用いられ、
前記複数のコードのうちの各コードは、前記ラインのうちの１つのラインに対応する論理遷移によって伝送されるか、または、前記ラインのうちの複数のラインに対応する論理遷移の組み合わせによって伝送され、
前記複数のコードから選択された一連のコードを前記ライン上に伝送することにより、１つまたは複数のトークンの各々を伝送することと、
各ライン上の論理遷移の総数を調整することにより、前記１つまたは複数のトークンおよび追加コードを伝送した後、前記リンクを確実に静止状態に戻すために、１つまたは複数の追加コードを前記ライン上に伝送することと、
を含む方法。
【請求項３６】
前記リンクが５本のラインを含み、前記ラインのうち４本のライン上それぞれにある論理遷移は各値を符号化し、前記ラインのうち残りの５本目のライン上にある論理遷移はエスケープコードを伝送するものであり、
前記方法がさらに、

前記５本目のラインにエスケープコードを伝送することなく、前記４本のライン上に伝送された一連のコードを伝送することにより、データトークンを伝送することと、
前記５本目のラインを使って前記エスケープコードを伝送することを含め、前記複数のコードから選択された一連のコードを伝送することにより、制御トークンを伝送することと、
を含む、請求項３５に記載の方法。
【請求項３７】
前記１つまたは複数の追加コードを伝送することが、上記ラインそれぞれにある論理遷移の総数を確実に偶数にするように、前記追加コードの少なくとも１つを伝送することを含む、請求項３５に記載の方法。
【請求項３８】
前記リンクの空き容量を示すために前記リンクが生成するクレジットトークンであって、１対または複数対の前記コードを含み、各対に含まれる前記コードが同一コードであるため、前記ラインそれぞれにある遷移総数が偶数または０となり、前記クレジットトークンの伝送後も、前記リンクの状態に何ら影響を及ぼさないクレジットトークンを伝送することをさらに含む、請求項３５に記載の方法。
【請求項３９】
複数のプロセッサと、３本のラインを含んでそのプロセッサ間を結ぶリンクとを含むデバイスであって、
前記リンクは少なくとも３本のラインを含み、
前記ラインは、複数のコードのうちの各コードを伝送するために用いられ、
前記複数のコードのうちの各コードは、前記ラインのうちの１つのラインに対応する論理遷移によって伝送されるか、または、前記ラインのうちの複数のラインに対応する論理遷移の組み合わせによって伝送され、
前記プロセッサの少なくとも第１のプロセッサが、
前記複数のコードから選択された一連のコードを前記ライン上に伝送することにより、１つまたは複数のトークンの各々を伝送し、
各ライン上の論理遷移の総数を調整することにより、前記１つまたは複数のトークンおよび追加コードを伝送した後、前記リンクを確実に静止状態に戻すために、前記ライン上にある１つまたは複数の追加コードを伝送するように構成されている、デバイス。
【請求項４０】
前記リンクが５本のラインを含み、前記ラインのうち４本のライン上それぞれにある論理遷移は各値を符号化し、前記ラインのうち残りの５本目のライン上にある論理遷移はエスケープコードを伝送するものであり、
前記第１のプロセッサが、
前記５本目のラインにエスケープコードを伝送することなく、前記４本のライン上に伝送された一連のコードを伝送することにより、データトークンを伝送し、
前記５本目のラインを使って前記エスケープコードを伝送することを含め、前記複数のコードから選択された一連のコードを伝送することにより、制御トークンを伝送するように構成されている、請求項３９に記載のデバイス。
【請求項４１】
前記第１のプロセッサが、前記追加コードの少なくとも１つを伝送する際に、前記ラインそれぞれにある論理遷移の総数を確実に偶数にするように、前記１つまたは複数の追加コードを伝送するように構成されている、請求項３９に記載のデバイス。
【請求項４２】
前記リンクが、前記リンクの空き容量を示すクレジットトークンであって、１対または複数対の前記コードのみを含み、各対に含まれる前記２つのコードが同一コードであるため、前記ラインそれぞれにある遷移総数が偶数または０となり、前記クレジットトークンの伝送後も、前記リンクの状態に何ら影響を及ぼさないクレジットトークンを生成するように構成されている、請求項３９に記載のデバイス。

【請求項４３】
プロセッサ間を結ぶリンク上に１つまたは複数のトークンを送信するコンピュータプログラムであって、
前記リンクが少なくとも３本のラインを含み、
前記ラインは、複数のコードのうちの各コードを伝送するために用いられ、
前記複数のコードのうちの各コードは、前記ラインのうちの１つのラインに対応する論理遷移によって伝送されるか、または、前記ラインのうちの複数のラインに対応する論理遷移の組み合わせによって伝送され、
前記プログラムが、
前記複数のコードから選択された一連のコードを前記ライン上に伝送することにより、１つまたは複数のトークンの各々を伝送するステップと、
各ライン上の論理遷移の総数を調整することにより、前記１つまたは複数のトークンおよび追加コードを伝送した後、前記リンクを確実に静止状態に戻すために、前記ライン上にある１つまたは複数の追加コードを伝送するステップとをプロセッサに実行させる、プログラム。
【請求項４４】
複数の処理手段と、前記複数の処理手段間をリンクするリンク手段とを含み、
前記リンク手段が少なくとも３つの送信手段を含み、
前記送信手段は、複数のコードのうちの各コードを伝送するためのものであって、
前記複数のコードのうちの各コードは、前記送信手段のうちの１つの送信手段に対応する論理遷移によって伝送されるか、または、前記送信手段のうちの複数の送信手段に対応する論理遷移の組み合わせによって伝送され、
前記処理手段の少なくとも第１の処理手段が、
前記複数のコードから選択された一連のコードを前記送信手段上に伝送することにより、１つまたは複数のトークンの各々を伝送し、
各送信手段上の論理遷移の総数を調整することにより、前記１つまたは複数のトークンおよび追加コードを伝送した後、前記リンク手段を確実に静止状態に戻すために、前記送信手段上にある１つまたは複数の追加コードを伝送する、デバイス。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、プロセッサ間を結ぶリンク上にトークンを送信することに関する。
【背景技術】
【０００２】
プロセッサ間、特に、同じ回路基板やチップ上などに配列、すなわち多数配置されたプロセッサアレイにメッセージを送るスイッチやリンクを含む回路を有するインターコネクトシステム上でやり取りされる大量の通信をどう扱うかは、プロセッサ設計者にとって１つの課題である。
【０００３】
具体的には、こうした通信ではインターコネクトで電力が相当に消耗することがまず問題である。通信での電力消費を低減することが望まれる。
【０００４】
次いで、制御情報の通信に問題がある。インターコネクトで送られるメッセージは通常、バイト単位のデータの集合である。ただし、インターコネクトそのものを制御するための制御情報を送信する機構も必要である。この制御情報はたとえば、スイッチまたはスイッチまたはリンクのいずれかに対する制御レジスタへの読取りまたは書取り要求などにより設定されるチャネルを閉じるための「メッセージ終了」信号である。ところが、データ送信機構とうまく両立する制御機構を見つけることは問題を含んでいる。
【発明の概要】
【発明が解決しようとする課題】

【０００５】
この問題を「メッセージ終了」の例を取って説明すると、回路設計者は通常、メッセージ終了を指示し、その後スイッチ類により２つの通信プロセッサ間のチャネルを閉じるように、１メッセージ内の２５５バイトを制御値に割り当てる。ところが、ソフトウェア開発者がその実数値２５５を、チャネル閉鎖要求として誤解されることなく宛先ソフトウェアに送信させたい場合、従来、必要に応じてそのインターコネクトをこのようにトリガさせないためには、複雑なエスケープシーケンスを適宜、その転送機構内に構築しなければならなかった。
【０００６】
さらに、種々のアプリケーションの特定ニーズ全般に有用となる、融通性を高めた制御機構の提供も必要である。
【課題を解決するための手段】
【０００７】
そこで、本発明は、プロセッサ間の通信による電力消費の低減を目的とする。実施形態の一部では、この低減した電力消費を、制御情報送信機構の改良型に効率よく組み合わせることをさらなる目的とする。
【０００８】
本発明の一態様によると、プロセッサ間を結ぶリンクであって、ライン１およびライン０を含み、ライン１上の論理遷移が論理１を示し、ライン０上の論理遷移が論理０を示すリンク上にトークンを送信する方法が得られ、この方法は、トークンの第１の部分を送信することと、トークンの第２の部分を送信して、確実にそのトークンの論理１のビット総数を偶数、かつそのトークン内の論理０のビット総数を偶数とし、これにより、そのトークンの送信後に、そのリンクが静止状態に戻ることとを含む。
【０００９】
各トークンが常に、偶数個１と偶数個の０を含むため、ライン１およびライン０はそれぞれ、常に偶数の論理遷移を行う。すなわち、遷移の立ち上がりがあれば、必ず遷移の立ち下がりがある。リンクが、トークンの送信を終えるごとに確実に低電力状態に戻るということである。
【００１０】
一部の実施形態において、この方法はさらに、データトークンを送信するか、制御トークンを送信するかを判断することを含み、前記第１の部分は、データトークンが送信された場合のデータの運搬に用いられ、また制御トークンが送信された場合には制御情報の搬送に用いられる情報部分と、前記トークンがデータトークンであるか制御トークンであるかを示す第１の追加ビットとを含む。
【００１１】
前記リンクを、同一基板またはチップ内のプロセッサ間に設けることができる。
【００１２】
前記第１の部分は、偶数の情報ビットおよび前記第１の追加ビットを含む奇数個のビットを有し、前記第２の部分は第２の追加ビットである。前記情報部分を８ビットとすることができる。
【００１３】
この方法はさらに、前記第１の部分が論理１である偶数個のビットを含み、かつ論理０である奇数個のビットを含んでいるか、前記第１の部分が論理１である奇数個のビットを含み、かつ論理０である偶数個のビットを含んでいるかを判断することを含み、もし第１の部分が偶数個の論理１および奇数個の論理０を含んでいれば、第２の部分は論理０のビットであり、もし第１の部分が奇数個の論理１および偶数個の論理０を含んでいれば、第２の部分は論理１のビットである。
【００１４】
この方法はさらに、第１の部分についてビット毎のＸＯＲ演算を実行して、第２の部分を算出することを含んでもよい。

【００１５】
第１の部分が９ビットを有する実施形態では、前記プロトコルが特に有効である。というのは、情報部分を、そのトークンが制御トークンかデータトークンかを示す１つの追加ビットを含む便利な１バイト（８ビット）と、そのリンクを静止状態に戻す１つの追加ビットとで構成できるからである。この第２の追加ビットの算出には、ビット毎のＸＯＲ演算が特に有効である。
【００１６】
また、バイト単位ではないがさらに長いトークンを使うことにより、複雑なエスケープシーケンスを回避することができる上、データ転送機構に影響を及ぼすことなく、さまざまな制御トークン全般が利用可能となる。
【００１７】
トークン内の送信順序は、第１の追加ビット、情報部分、第２の追加ビットとすることができる。
【００１８】
前記第１の部分を、前記プロセッサのうちの１つで実行されるソフトウェアで生成し、また前記第２の部分を、リンク内ハードウェアにより生成することができる。第１の部分を、前記プロセッサの１つで実行される命令のオペランドとすることができる。
【００１９】
前記トークンを、アーキテクチャで定義される制御トークンとすることができ、本方法はさらに、この制御トークンを使用して前記リンクを含む回路を有するインターコネクト内で論理をトリガして、そのインターコネクトの構成要素をコントロールすることを含むことができる。本方法はさらに、ソフトウェアの機能を遂行するため、宛先プロセッサで実行されるソフトウェアを用いた、アーキテクチャで定義される制御トークンの受信を含むことができる。前記アーキテクチャで定義される制御トークンを、その宛先プロセッサで実行される特権ソフトウェアにのみアクセス可能な特権制御トークンとすることができる。
【００２０】
前記トークンを、ソフトウェアに定義される制御トークンとすることができる。
【００２１】
前記トークンを、宛先プロセッサを指定する１つまたは複数のヘッダトークンを含むメッセージで送信することができる。
【００２２】
前記トークンを、１アレイ内の２つ以上のプロセッサ間をつなぐ、スイッチおよびリンクのシステムを含む回路を有するインターコネクト上に送信することができる。
【００２３】
本発明の他の態様によれば、複数のプロセッサと、ライン１およびライン０を含んで前記プロセッサ間を結ぶリンクとを含み、前記ライン１上の論理遷移が論理１を示し、前記ライン０上の論理遷移が論理０を示すデバイスが提供される。このデバイスでは、前記プロセッサの少なくとも第１のプロセッサは、トークンの第１の部分を送信し、そのトークンの第２の部分を送信して、確実にそのトークン内における論理１のビット総数を偶数とし、かつそのトークンの論理０のビット総数を偶数とし、これにより、そのトークンの末尾で、そのリンクを静止状態に戻すように構成されている。
【００２４】
本発明の他の態様によれば、プロセッサ間を結ぶリンクであって、ライン１およびライン０を含むリンク上にトークンを送信するコンピュータプログラムであって、ライン１での論理遷移が論理１を示し、およびライン０での論理遷移が論理０を示すプログラムが得られる。このプログラムは、トークンの第１の部分を送信するステップと、トークンの第２の部分を送信して、確実にそのトークンの論理１のビット総数を偶数、かつそのトークン内の論理０のビット総数を偶数とし、これにより、確実にそのトークンの末尾で、そのリンクを静止状態に戻すステップとをプロセッサに実行させる。

【００２５】
本発明の他の態様によれば、複数の処理手段と、前記処理手段間をリンクするリンク手段とを含むデバイスが得られる。このリンク手段は、論理遷移により論理１を示す論理１送信手段と、論理遷移により論理０を示す論理０送信手段とを含み、前記処理手段の少なくとも第１の処理手段が、トークンの第１の部分を送信する送信手段を含み、その送信手段はさらに、同じトークンの第２の部分を送信して、確実にそのトークンの論理１のビット総数を偶数、かつそのトークン内の論理０のビット総数を偶数とし、これにより、そのトークンの末尾で、そのリンクを静止状態に戻す。
【００２６】
本発明の他の態様によれば、プロセッサ間のリンクを介して１または複数のトークンを送信する方法であって、前記リンクは少なくとも３本のラインを含み、前記ラインは、複数のコードのうちの各コードを伝送するために用いられ、前記複数のコードのうちの各コードは、前記ラインのうちの１つのラインに対応する論理遷移によって伝送されるか、または、前記ラインのうちの複数のラインに対応する論理遷移の組み合わせによって伝送される。この方法は、前記複数のコードから選択された一連のコードを前記ライン上に伝送することにより、１つまたは複数のトークンの各々を伝送することと、各ライン上の論理遷移の総数を調整することにより、前記１または複数のトークンおよび追加コードの伝送後、前記１または複数の追加コードを前記ライン上に伝送することとを含む。
【００２７】
リンクに５本のラインを設けてもよい。このうち４本のライン上それぞれにある論理遷移は各値を符号化し、残りの５本目のライン上にある論理遷移はエスケープコードを伝送する。前記方法はさらに、前記５本目のラインにエスケープコードを伝送せず、前記４本のライン上に伝送された一連のコードを伝送することにより、データトークンを伝送することと、前記５本目のラインを使って前記エスケープコードを伝送することを含め、前記コードから選択された一連のコードを伝送することにより、制御トークンを伝送することを含むことができる。
【００２８】
前記１または複数の追加コードを伝送することが、前記ラインそれぞれにある論理遷移の総数を確実に偶数にするように、前記追加コードの少なくとも１つを伝送することを含んでもよい。
【００２９】
前記方法はさらに、前記リンクの空き容量を示すため、前記リンクが生成するクレジットトークンを伝送することを含む。このクレジットトークンは、１対または複数対の前記コードを含み、各対の前記コードは同一コードであるため、前記ラインそれぞれにある遷移総数が偶数または０となり、前記クレジットトークンの伝送後も、前記リンクの状態に何ら影響が及ばないものである。
【００３０】
本発明の他の態様によれば、複数のプロセッサと、３本のラインを含んでそのプロセッサ間を結ぶリンクとを含むデバイスが得られる。前記リンクは少なくとも３本のラインを含み、前記ラインは、複数のコードのうちの各コードを伝送するために用いられ、前記複数のコードのうちの各コードは、前記ラインのうちの１つのラインに対応する論理遷移によって伝送されるか、または、前記ラインのうちの複数のラインに対応する論理遷移の組み合わせによって伝送される。このデバイスにおいて、前記プロセッサの少なくとも第１のプロセッサが、前記複数のコードから選択された一連のコードを前記ライン上に伝送することにより、１つまたは複数のトークンの各々を伝送し、各ライン上の論理遷移の総数を調整することにより、前記１または複数のトークンおよび追加コードを伝送後、前記リンクを確実に静止状態に戻すために、前記ライン上にある１または複数の追加コードを伝送するように構成されている。
【００３１】
本発明の他の態様によれば、プロセッサ間を結ぶリンク上に１または複数のトークンを

送信するコンピュータプログラムであって、前記リンクが少なくとも３本のラインを含み、前記ラインは、複数のコードのうちの各コードを伝送するために用いられ、前記複数のコードのうちの各コードは、前記ラインのうちの１つのラインに対応する論理遷移によって伝送されるか、または、前記ラインのうちの複数のラインに対応する論理遷移の組み合わせによって伝送されるプログラムが得られる。このプログラムは、前記複数のコードから選択された一連のコードを前記ライン上に伝送することにより、１つまたは複数のトークンの各々を伝送するステップと、各ライン上の論理遷移の総数を調整することにより、前記１または複数のトークンおよび追加コードを伝送後、前記リンクを確実に静止状態に戻すために、前記ライン上にある１または複数の追加コードを伝送するステップとをプロセッサに実行させる。

【図面の簡単な説明】

【００３２】

【図１】インターフェースプロセッサの適用例を例示している。

【図２】インターフェースプロセッサの別の適用例を例示している。

【図３】インターフェースプロセッサのアーキテクチャを示す概略図である。

【図４】ポートを示す概略図である。

【図５】スレッドレジスタのセットを示す概略図である。

【図６】スレッドレジスタのセット間のインターコネクトを示す概略図である。

【図７】チャネル端を示す概略図である。

【図８】プロセッサ間のインターコネクトを示す概略図である。

【図９】トークンのフォーマットを示す。

【図１０】読取り要求メッセージのフォーマットを示す。

【図１１】成功した読取り応答メッセージのフォーマットを示す。

【図１２】失敗した読取り応答メッセージのフォーマットを示す。

【発明を実施するための形態】

【００３３】

図１は、インターフェースプロセッサを携帯電話で使用した適用例である。このモバイル用アプリケーションプロセッサ２は、複数の周辺機器８と通信して機能を果たすものである。アプリケーションプロセッサ２は、バス３、ＣＰＵ４、ハードドライブ（ＨＤＤ）８ａとインターフェースするメモリコントローラ６ａ、およびＳＤＲＡＭメモリ８ｂを含み、さらに、パワーコントローラ１０および無線プロセッサ１２を含む。

【００３４】

図１の配置では、ＣＰＵ４が汎用ポート７を介して外部と通信することができる。この例では、汎用ポート７ａおよび７ｂが、カメラ類８ｃおよびＬＣＤディスプレイ類８ｄとインターフェースし、汎用ポート７ｃがマイクロホン８ｅ、スピーカ８ｆおよびヘッドセット８ｇとインターフェースし、汎用ポート７ｄがキーボード８ｈ、汎用シリアルバス（ＵＳＢ）デバイス８ｉ、セキュアデジタル（ＳＤ）カード８ｊ、マルチメディアカード（ＭＭＣ）８ｋ、および汎用非同期送受信回路（ＵＡＲＴ）デバイス８ｌとインターフェースするように設けられている。

【００３５】

図１において、インターフェースプロセッサ１４ａ、１４ｂおよび１４ｃは、関連ポート７の出口に配置されている。このとき、第１のインターフェースプロセッサ１４ａは画像装置８ｃ～８ｄと汎用ポート７ａ～７ｂとの間に接続され、第２のインターフェースプロセッサ１４ｂはオーディオ装置８ｅ～８ｇ間に接続され、第３のインターフェースプロセッサ１４ｂは汎用ポート７ｄとさまざまな接続性デバイス８ｈ～８ｍとの間に接続されている。アプリケーション専用表示、オーディオ、および接続性機能は、後述するようにインターフェースプロセッサ１４ａ－１４ｃが実行するため、ポート７は汎用ポートであればよい。また、ＦＰＧＡから得られるはずの柔軟性および構成容易性はインターフェースプロセッサ１４で得られるため、ポート７でＦＰＧＡ論理を用いる必要はない。インタ

ーフェースプロセッサ１４ａには、ポート７ａおよび７ｂに接続されたポート２２ａおよび２２ｂと、外部デバイス８ｃおよび８ｇに接続されたポート２２ｃ、２２ｄ、２２ｅおよび２２ｆが設けられている。インターフェースプロセッサ１４ｂおよび１４ｃも、図１に示していないが、同様のポートを有する。

【００３６】

インターフェースプロセッサは通常、インターフェースを介したデータの転送に用いる特定のプロトコルの実施、パラレルフォーマットとシリアルフォーマットとの間のデータ変換などのデータの書式変更、そして、そのデータのコード化、圧縮、暗号化などの高次機能にまで関わるものである。

【００３７】

インターフェースプロセッサ１４のもう１つの適用例として、図２に例示するプロセッサアレイ２００の一部として用いることが挙げられる。このアレイ２００は、複数のプロセッサタイル２０２を含み、各タイルが、アレイ内のノードを画定し、かつ１つまたは複数のプロセッサ１４およびインターコネクト２０４を含んでいる。タイル２０２は、アレイ２００内のタイル２０２間通信を支援する高性能接続部２１９を介し、ポート２２を使って一部のプロセッサ１４に接続され、アレイ２００外側の他のデバイスと通信できるようになっている。このアレイは、１枚のチップ上で実装されていても、複数枚のチップを組み合わせて実装されていてもよい。

【００３８】

このインターフェースプロセッサは、内部通信、外部通信を問わず、通信を管理できることが大きな特徴である。これについては以下でさらに詳述する。各インターフェースプロセッサは、ＣＰＵ、メモリ、および通信機能を有する。このＣＰＵとポート類との間を直接かつ応答可能に接続するため、各プロセッサは、複数の同時実行プログラムスレッドを実施するハードウェア支援を備えている。このプログラムスレッドはそれぞれ、一連の命令を含み、命令の少なくとも一部を通信処理にあたるものとすることができる。以下でさらに詳しく説明するが、このハードウェア支援の例として次のものが挙げられる。

（１）スレッド１つに対するレジスタのセット
（２）実行すべきスレッドを動的に選択するスレッドスケジューラ
（３）入出力用ポートのセット（ポート２２）
（４）スレッド間にチャネルを形成するインターコネクトシステム

【００３９】

各プロセッサにスレッドの小セットが設けられているため、これを使って、そのプロセッサが処理している他の保留中のタスクと同時に通信や入出力を進行させることができる。また、スレッドによって継続したり、遠隔インターフェースプロセッサとの双方向通信を待ちながら中断させたりして、インターコネクトにおける遅延を隠蔽することができる。

【００４０】

図３は、本発明の一実施形態によるインターフェースプロセッサ１４の代表的アーキテクチャを概略的に示す。プロセッサ１４は、スレッドスケジューラ１８の制御下で複数の命令スレッドを実行する実行ユニット１６を含む。プロセッサ１４はさらに、プログラムコードおよびその他のデータを保持するランダムアクセスメモリ（ＲＡＭ）２４と、ブートコードなどの永続的な情報を格納するリードオンリーメモリ（ＲＯＭ）（図示せず）とを含む。

【００４１】

スレッドスケジューラ１８は、実行部１６が実行すべきスレッドを動的に選択する。従来、スレッドスケジューラの機能は、プロセッサを完全な占有状態に保つよう、プログラムメモリからスレッドを選択してスケジューリングしておくだけであった。しかし、本発明によれば、スレッドスケジューラ１８によるスケジューリングは、ポート２２におけるアクティビティにも及ぶ。この観点から、ポートにおける入出力アクティビティによりスレッドを実行できるようになった時点での遅れを最小限に抑えるため、スレッドスケジュ

ーラをポート２２に直接結合させてもよいことに留意されたい。

【００４２】

スレッドスケジューラ１８により対象となっているｍスレッドはそれぞれ、スレッドスケジューラ１８がアクセスするレジスタ２０のバンク内スレッドレジスタ20_1、…、20_mの各セットで表される。複数の命令バッファ（ＩＮＳＴＲ）１９も、メモリ２４からフェッチされた命令が、実行部１６に向けて送出される前に一時的に保持するよう設けられている。チャネルを介してレジスタセット２０の間におけるデータ通信を行うことができる。このレジスタおよびチャネルについては後で詳述する。

【００４３】

スレッドスケジューラ１８は、ｍスレッドの中から実行可能なスレッドの１セットを維持し（以下、これを「実行セット」とする）、そこから順番に、好ましくはラウンドロビン方式で、命令を取り出す。あるスレッドが継続不可となったら、そのスレッドを実行組から除去して中断する。継続不可となる理由として、そのスレッドが次のいずれか１つまたは複数のタイプのアクティビティを待っている、などが挙げられる。

（１）実行可能となる前に、そのレジスタが初期化される

（２）用意の出来ていない、または利用可能なデータのないポートまたはチャネルから入力を試みた

（３）用意の出来ていない、またはデータを受け入れる余地のないポートまたはチャネルに出力を試みた

（４）命令を実行し、これにより、ポートまたはチャネルが入力の用意ができると生成され得る１つまたは複数のイベントを待っている

【００４４】

本明細書でいう用語「イベント」は、特定の動作をいい、基本的な入出力動作とは多少異なることに留意されたい。その区別については、図４および図５と合わせて以下で説明する。

【００４５】

好ましくは、応答時間を短縮するため、スレッドスケジューラ１８と実行部１６との間には直接的な配線接続部２８が設けられており、これにより、スレッドスケジューラ１８は、どのスレッド（単数または複数）を実行部１６にフェッチおよび実行させるか制御するように構成できる。さらに、直接的な配線経路３０ａ、３０ｂ、３０ｃが、スレッドスケジューラ１８とポート２２のそれぞれとの間に、また直接的な配線経路29_1、…29_mが、スレッドスケジューラ１８とレジスタ２０それぞれとの間に設けられている。こうした直接的経路が制御路となって、スレッドスケジューラが各スレッドを１または複数のポート２２に関連付けられることが好ましい。特に、特定のアクティビティが発生した時点でポートから用意できている指示を返送させ、これにより、プロセッサがポート２２で発生したアクティビティまたは刺激に迅速に反応できることが好ましい。ポートに対するスレッドスケジューラの動作は、図４～図６について以下で説明する。

【００４６】

実行部１６はさらに、ポート２２ａ～２２ｃのそれぞれ、およびレジスタ20_1～20_mのそれぞれに直接的接続部２７および３１を介してアクセスできるため、中枢プロセッサとレジスタと外部環境との間を直接的にリンクできる。好ましくは、こうした直接的経路を制御路として、実行部がポートに条件をパスできるようにする。これについては、以下で図４を参照しながらさらに詳しく説明する。直接路２７および３１によりさらに、スレッドレジスタ２０とポート２２との間でデータの直接入出力をできるようにして、スレッドの外部環境との直接通信を可能としてもよい。例えば、データをメモリ２に書込んでからそのデータをフェッチするのではなく、データを外部装置から直接スレッドのオペランドレジスタの１つに書き込んでもよい。逆に言えば、書込み動作の後、オペランドレジスタからのデータを実行部１６が取り出し、これをポート２２に直接送信してもよい。これにより、反応時間が大幅に短縮される。

【００４７】

「直接的接続部」「直接経路」とは、実行部とプログラムメモリ２４との間の接続とは別の接続部であることに留意されたい。したがって、例えば、データをメモリ２４に格納してそこからフェッチしなくても、スレッドスケジューラ１８および実行部１６は、ポート２２から入力されたデータにアクセスできる。特に、実行部１６とメモリ２４との間の接続部がバス１３を介している場合、「直接的」な接続部または経路は、そのバスとは別のものを言う。したがって、バスを仲介させなくとも、ポート２２、レジスタ２０，スレッドスケジューラ１８、および実行部１６の間で種々の通信をすべて発生させることができ、反応時間の短縮となる。ポート２２にさらに、バス１３による追加接続部（図示せず）を設けてもよい。

【００４８】

図４は、本発明の好適な一実施形態によるポート２２を概略的に示す。ポート２２は、プロセッサ１４との間でデータを入出力させるＩ／Ｏバッファ３２を含む。また、ポート２２はそれぞれ、そのポートで発生するアクティビティを監視し、アクティビティの発生を少なくとも１つのレディビットすなわちフラグ３７により伝送するアクティビティ処理論理３６を含む。状態フラグ３７は、直接路３０からスレッドスケジューラに伝送されることが好ましい。ポートが検知するアクティビティの例として以下が挙げられる。

（１）データがポートに入力された

（２）特定のデータがポートに入力された、および／または

（３）ポートが出力に利用できる状態になった

【００４９】

上述のアクティビティを検知しやすくするために、ポート２２にはレジスタのセット３８が設けられている。このレジスタには、関連スレッドの識別子を格納するスレッド識別子（ＴＩＤ）レジスタ、１または複数の条件を格納する制御（ＣＴＲＬ）レジスタ、実行が中断されたプログラム内の位置を格納する継続点ベクトル（ＶＥＣＴＯＲ）レジスタ、および１つの条件に関連するあらゆるデータを格納するデータ（ＤＡＴＡ）レジスタが含まれる。ＴＩＤ値が直接路３０（図３の３０ａ、３０ｂ、３０ｃ）を介してスレッドスケジューラ１８によりレジスタ３８に書き込まれると、ＶＥＣＴＯＲ値、ＣＴＲＬ値、およびデータ値が実行部１６により直接路３１を介して書き込まれる。所望のアクティビティが検知されると、ＴＩＤは、関連スレッドの識別のため、スレッドスケジューラ１８に返送される。このアクティビティ論理にはイネーブルフラグ３９も含まれるが、これについては以下で詳述する。

【００５０】

図４では、レジスタ３８はポート２２内に収まっているが、プロセッサ１４内であってポートに続いていれば実際にはどこにあってもよいことに留意されたい。

【００５１】

図５は、スレッドの表示に用いられるスレッドレジスタ２０の代表的バンクを示す。バンク２０は、スレッドスケジューラ１８により目下の対象となっているスレッドT_1～T_mそれぞれに対応するレジスタの複数セットを含む。この好適な例では、各スレッドの状態が、２個の制御レジスタ、４個のアクセスレジスタ、および１２個のオペランドレジスタ、合計１８個のレジスタによって表示されている。その内訳は以下の通りである。

【００５２】

（１）制御レジスタ：

・ＰＣは、プログラムカウンタ

・ＳＲは、ステータスレジスタ

（２）アクセスレジスタ：

・ＧＰは、グローバルプールポインタ

・ＤＰは、データポインタ

・ＳＰは、スタックポインタ

・ＬＲは、リンクレジスタ

（３）オペランドレジスタ：

・ＯＰ１…ＯＰ１２
【００５３】
制御レジスタは、スレッドの状態についての情報を格納するものであり、スレッド実行の制御に用いられる。具体的には、イベントまたは中断に反応するスレッドの性能が、スレッドステータスレジスタＳＲが保持する情報により制御されている。アクセスレジスタは、プロシージャの局所変数に用いるスタックポインタと、プロシージャ間で共有されるデータに通常用いられるデータポインタと、大きな定数およびプロシージャエントリポイントへのアクセスに用いられるコンスタントプールポインタとを含む。オペランドレジスタＯＰ１…ＯＰ１２は、命令により用いられ、算術演算および論理演算を行い、データ構造にアクセスし、サブルーチンを呼び出す。図６および図７に関連して説明するように、このプロセッサはさらに、異なるセット２０のオペランドレジスタ間にチャネルを形成するインターコネクトシステム４０を含む。
【００５４】
いくつかの命令バッファ（ＩＮＳＴＲ）１９も設けられており、スレッドの実際の命令を一時的に格納する。各命令が好ましくは１６ビット長として、命令バッファの長さがそれぞれ６４ビットで、４つの命令に対応できると好ましい。命令は、スレッドスケジューラ１８の制御下でプログラムメモリ２４からフェッチされた後、一時的に命令バッファ１９内に置かれる。
【００５５】
実行部はレジスタ２０およびバッファ１９のそれぞれにアクセスできる。さらに、スレッドスケジューラ１８が、スレッドごとに少なくともステータスレジスタＳＲにアクセスできる。
【００５６】
上述したように、本明細書で言う「イベント」とは、特定種類の動作、またはその動作に相当するアクティビティをいう。イベントベースの動作は、基本的な入出力動作とは多少異なり、次のように作用する。イベントはまず、実行部１６からの継続点ベクトルおよびスレッドスケジューラ１８からのスレッド識別子が、好ましくは直接路３１および３０を介して、ポート２２に付随するＶＥＣＴＯＲおよびＴＩＤレジスタ３８に転送されることにより、スレッドごとにセットされる。関連する条件および条件データをさらに、そのポート２２のＣＴＲＬおよびデータレジスタ３８に書き込んでもよい。したがって、このイベントはポートにセットされるが、必ずしもイネーブルにされなくてもよい。ポートをイネーブルにしてイベントの指示を生成させるには、ポートのイネーブルフラグ３９も、好ましくは直接路３０を介してスレッドスケジューラ１８により、アサートしなくてはならない。さらに、そのスレッド自体をイネーブルしてイベントを許可させるため、そのスレッド用ステータスレジスタＳＲ内にある、そのスレッドのイベントイネーブル（ＥＥ）フラグを、イベントイネーブルに設定しなくてはならない。このようにイベントをセットおよびイネーブルしたら、そのスレッドは、スレッドスケジューラ１８に働きかけるイベントベースの待機命令を使って、イベントを待ちながらの保留状態でいることができる。この時点で、目下保留中の命令を関連命令バッファ１９から破棄してもよい。例えば、何らかのデータが入力されたなど、イベントが発生したら、その発生を、ポート２２からスレッド識別子および継続点ベクトルをそれぞれスレッドスケジューラ１８および実行部１６に返送することで伝送する。これにより、継続点ベクトルに識別された命令がプログラムメモリ２４から命令バッファ１９にフェッチされ、そのコード内の適切な時点で実行が再開される。例えば、待っているイベントが特定データの入力であれば、継続点ベクトルは、そのデータを入力する入力命令を含むコードを識別すればよい。
【００５７】
イベント発生後、各ステータスレジスタＳＲ内にあるそのスレッドのＥＥフラグを、イベントディセーブルにセットして、その発生直後、スレッドがイベントに反応しないようにしてもよい。イベント発生時にスレッドが命令を実行した結果として、イネーブルフラグ３９をディアサートしてもよい。

【００５８】
いくつかのポートを１または複数のポートからのイベントを待つ態勢にセットしながら、イネーブルフラグ３９をアサートすることができる。ポートイネーブルフラグのセットをイネーブルする前に、スレッドのＥＥフラグをイベントイネーブルにセットしてもよく、この場合、用意のできた、イネーブルすべき第１のポートが、継続点ベクトルにある命令を直ちにフェッチおよび実行することにより、目下の命令を破棄し、実行を処理するイベントを生成することになる。
【００５９】
このポートのイネーブルフラグ３９およびステータスレジスタＥＥフラグの利点は、イベントのイネーブルおよびディセーブルが、待機命令によりイベントのセッティングおよびスレッドの保留のどちらとも切り離されることであり、このため、特定のスレッドおよび／またはさまざまな異なるスレッド用にさまざまな入出力条件のオンオフを簡単に切り替えられる。例えば、あるイベントがディセーブルであっても、そのイベントをポート２２にセットした状態にしておくことができる。したがって、スレッドでのイベントの再利用が可能となる。というのも、そのイベントはすでに一度発生したにもかかわらず、そのスレッド識別子、継続点ベクトル、および条件がポート２２のＴＩＤ、ＶＥＣＴＯＲ、ＣＴＲＬ、およびデータレジスタ３８にまだ格納されているからである。そのため、スレッドがそのイベントを再利用する必要があれば、そのポートのレジスタ３８への再書き込みは不要であり、その代わりに、ポートのイネーブルフラグ３９を再度アサートする、かつ／またはスレッド用ステータスレジスタＳＲ内ＥＥフラグをイベントイネーブルにリセットするだけでよい。次に待機命令が出れば、そのスレッドは同一イベントが再度発生するまで中断される。
【００６０】
さらに、継続点ベクトルを使用すれば、複数のイベントをスレッドごとにイネーブルにすることができる。つまり、任意のスレッドは、継続点ベクトルを１つのポート２２ａに転送することにより、そのポートに１つのイベントを設定し、別の継続点ベクトルを別のポート２２ｂに転送することにより、そのポートに別のイベントを設定することができ、その他も同様である。このスレッドはまた、ポートごとに異なるイネーブルフラグ３９を別々にアサートまたはディアサートすることにより、さまざまなイベントを個々にイネーブルおよびディセーブルできる。次に待機命令が出れば、そのスレッドは、イネーブルになったイベントの発生を待つ状態で中断される。
【００６１】
イベントとは対照的に、基本的なＩ／Ｏ動作は、先の待機命令は使わず、入力命令または出力命令だけを使用する。基本的なＩ／Ｏ動作を使用する際、スレッドスケジューラ１８は、継続点ベクトルをＶＥＣＴＯＲレジスタに送信せず、またステータスレジスタＳＲ内のポートのイネーブルフラグ３９やＥＥフラグも使用しない。代わりに、次の保留命令が命令バッファ１９に残されるだけであり、必要に応じて、入力命令または出力命令がスレッドスケジューラ１８に出されて実行を停止し、状態フラグ３７が示すように、データの入力または出力用ポートの利用性を待つ状態に入る。ポートがすぐに利用可能であれば、すなわち、入力命令または出力命令が実行された時点で状態フラグ３７がすでにセットされていれば、そのスレッドが停止されることはない。一部の実施形態において、基本的Ｉ／Ｏによるスケジューリングに必要なのがＴＩＤレジスタのみとなる。基本的Ｉ／Ｏは、ＣＴＲＬおよびデータレジスタ内の条件を使用しても使用しなくてもよい。使用しない場合、ポートの準備ができ次第、Ｉ／Ｏは完了する。基本的なＩ／Ｏ動作はスレッドを停止または再開するが、ステータスレジスタＳＲ内のポートのイネーブルフラグ３９やＥＥフラグを実行することも、コントロールをイベントベクトルに転送することもない。
【００６２】
同様のイベントおよびＩ／Ｏ技術を、スレッド間、より正確には、スレッドに関連する情報を格納するスレッドレジスタセット２０の間の通信に適用することができる。図６は、チャネルを形成するための回路を含むインターコネクトシステム４０を示す。図をわか

りやすくするため、図６では４つのスレッドレジスタセット２０$_1$～２０$_4$のみを図示しており、それぞれが各スレッドＴ$_1$～Ｔ$_4$用の情報を格納している。このスレッドレジスタセットはそれぞれ、インターコネクトシステム４０により互いに接続されている。このシステム４０は、少なくとも２つのスレッドレジスタセット２０間で直接データを転送するための少なくとも１つのチャネルを形成するように動作可能な直接的なハードウェア内部結線である。この内部結線は、直接メモリアクセス（ＤＭＡ）を使用せず、その転送は、ＲＡＭ２４などの共有メモリを介さず、バス１３などのいかなる汎用システムバスを介するものでもないという意味で直接的である。チャネルは、オペランドレジスタＯＰとの間のデータ転送に使用されることが好ましいが、原理としては、ステータスレジスタＳＲなどの他の型のレジスタとの間の情報転送に使用することも可能である。スレッドスケジューラ１８は、ポートについて上述したのと同様に、チャネルに発生するアクティビティに基づいてスレッドをスケジュールすることができる。ポート、チャネル、およびその他のアクティビティのソースを総括して本明細書で使用する用語は「資源」である。

【００６３】

インターコネクトシステム４０は、スレッド間のチャネル形成に用いる複数のハードウェア端末４２を含む。以下、この端末を「チャネル端」とする。各チャネル端（すなわち、チャネル端末）は、スレッドレジスタセット２０のいずれにも割り当て可能であり、各チャネル端４２は他のいずれのチャネル端４２にもインターコネクトシステム４０により接続可能である。図６では、わかりやすくするため、４つのチャネル端のみ図示しているが、この数は増減可能であり、また一般にレジスタセット２０の数とチャネル端４２の数は異なっていてもよいことを理解されたい。

【００６４】

各チャネル端４２は、受信して、それが入力されるまでデータを保持するバッファを含み、そのバッファは、好ましくは、保持されているデータの量の記録も保持する。チャネル端４２はまた、そのチャネルを介して出力されたデータが正しい入力バッファに書き込まれるように、別のチャネル端に接続されたかどうかの記録、そして接続されたチャネル端のアドレスの記録も保持する。このバッファおよび記録は２つのファイル、すなわちチャネル入力ファイルとチャネル出力ファイルとを用いて実行される。このチャネル入力および出力「ファイル」は、「レジスタファイル」の一部であるため、この意味から、レジスタおよびバッファを実行するためのプロセッサ１４上の専用メモリの小さな１ブロックということができる。ただし、レジスタファイル内（すなわち各レジスタ）内の各エントリは特定の目的のために確保される上、レジスタへのアクセスはシステムバス１３を介さないことから、このレジスタファイルは、メモリ２４などの汎用ＲＡＭとは性質が異なる。

【００６５】

図７に示すように、チャネル端４２のそれぞれは、一対のポートに似ており、スレッド間に全二重データ転送を行うための入力バッファ４４と出力バッファ４６とを有する（任意に、単一バッファとしてもよい）。入力バッファ４４は、別のチャネル端４２からのデータをスレッドのレジスタセット２０に入力するように動作可能であり、出力バッファ４６は、そのスレッドのレジスタセット２０からのデータをその他のチャネル端４２に出力するように動作可能である。各バッファが、少なくとも１ワードをバッファさせられるように十分なトークンを保持できることが好ましい。

【００６６】

ポート２２同様、チャネルの入力バッファ４４および出力バッファ４６をそれぞれ、アクティビティ処理論理３６’と関連づけることができる。この論理３６’は、チャネルで発生するアクティビティを監視し、少なくとも１つの状態フラグ３７’により（１フラグは１ビットのレジスタ）アクティビティの発生を伝送するためのものである。このアクティビティの例として、データがチャネルに入力された、チャネルが出力用に利用可能となったなどが挙げることができる。チャネルが満杯でデータを受け取ることができない時に出力命令が実行された場合、スレッドスケジューラ１８がその命令を一時停止し、その命

令を完遂できるほどにチャネルに余地ができた時点で再スタート、すなわち再実行させる。同様に、入力命令が実行されたが利用可能なデータが十分にない場合、十分なデータが利用可能となるまで、スレッドスケジューラ１８がそのスレッドを一時停止する。チャネル端４２にはカウンタ４７が設けられており、これが入力バッファ４４および出力バッファ４６内のデータ量の記録をつけている。

【００６７】

スレッドレジスタの２セット間にチャネルを形成するには、２つのチャネル端の割り当ておよび接続が必要である。上述したように、各チャネル端はどのスレッドにも割り当て可能であり、各チャネル端４２は他のチャネル端４２のいずれとも接続可能である。チャネル端４２の割り当ておよび接続を容易にするため、各端４２に、その端がどのチャネル端に接続されているかを記録するチャネル端識別子レジスタＣＥＩＤ４１と、そのチャネル端が接続されたかどうかを記録する被接続フラグ４３と、そのチャネル端がスレッドにより要求されたかどうかを記録する被要求フラグ４５とを含めることができる。

【００６８】

各チャネル端４２を２つのスレッドそれぞれに割り当てるため、２つの「チャネル端獲得」命令が実行され、各命令が、スレッドの一方で使用するように１つのチャネル端４２を確保する。この命令はそれぞれさらに、各チャネル端４２の被要求フラグ４３をアサートする。各「チャネル端獲得」命令は２つのスレッドそれぞれで実行しても、両方の「チャネル端獲得」命令を１つのマスタースレッドで実行してもよい。

【００６９】

次に、チャネル端を、次のようにチャネル端識別子を交換して、互いに接続する。第２のスレッドのチャネル端への出力を行うために第１のスレッドの出力命令が実行される場合、第２のスレッドのチャネル端にある被接続フラグ４３を用いて、第２のスレッドのチャネル端が目下接続されているかどうかを判断する。第２のスレッドのチャネル端が接続されていなければ、そのチャネル端に供給されるデータが、第１のスレッドのチャネル端の識別子であると翻訳される。この識別子が第２のスレッドのチャネル端のＣＥＩＤレジスタ４１に記録され、その結果、第２のスレッドのチャネル端の被接続フラグ４３がアサートされる。これと交代に、第２のスレッドの出力命令が、第１のチャネルへの出力を行うために実行される。第１のスレッドのチャネル端の被接続フラグ４３がまだアサートされていないとすると、第１のスレッドのチャネル端に供給されるデータが、第２のスレッドのチャネル端の識別子であると翻訳される。この識別子が第１のスレッドのチャネル端のＣＥＩＤレジスタ４１に記録され、その結果、第１のスレッドのチャネル端の被接続フラグ４３がアサートされる。

【００７０】

チャネル端４２同士が接続されると、第２のチャネル端への出力はいずれも、第２のスレッドのチャネル端のＣＥＩＤレジスタ４１内の記録から、関連する第１のチャネル端を特定する。第２の命令の入力バッファにそのデータを保持する余地があれば、そのデータは転送され、余地がなければ、第１のスレッドの出力命令は一時停止される。出力命令によるデータの第２のチャネル端への供給も、第２のスレッドが第２のチャネル端への入力を待って一時停止されていた場合は、第２のスレッドを再開して、データが取り込めるようにする。同様に、第２のチャネル端からデータを入力する第２のスレッドによって、第１のチャネル端から第１のスレッドの一時停止された出力からデータ用にスペースが得られるのであれば、第１のスレッドの出力を再開して、その実行を完了させる。この入力をイベントのトリガとしてもよい（下記参照）。各スレッドに対して、スレッドスケジューラ１８は、すべての一時停止された出力命令、その関連データ、およびデータを転送しようとしているチャネル端に関する記録を保持する。

【００７１】

チャネルが不要となったら、「メッセージの終了」（ＥＯＭ）制御トークンを出力する命令を実行して、チャネル端４２同士を切り離すことができる。これでチャネル端４２は他のいずれのチャネル端とも接続出来る状態となる。また、各チャネル端４２は、「フリ

ーチャネル」命令を実行すれば、スレッドからも自由になる。これにより、チャネル端４２は他のいずれのスレッドが使用してもよい自由な状態となる。

【００７２】

チャネルレジスタファイルについて説明したように、プロセッサがチャネル端ｃに出力を行うと、チャネル出力ファイルのｃ番目のエントリがチェックされて、ｃが接続されているかどうかを判断する。接続されていない場合、出力データｄが、その後のｃへの出力が送信されるチャネル端のアドレス（すなわちＩＤ）であると翻訳される。次にそのアドレスｄを調べて、同一プロセッサ上のチャネル端のアドレスかどうかを判断する。そうであれば、アドレスｄはチャネル出力ファイルのｃ番目のエントリに書き込まれる。この後のｃを介する出力はｃ番目のエントリにアクセスして、ｃに接続されたチャネル端を判断し、そして、満杯でなければ、接続されているチャネル端ｄの入力データに出力データを書込む。バッファが満杯であると判明した場合、出力命令は、バッファに十分な空きができるまで一時停止される。またこの場合、出力スレッドは入力命令により解放されるため、十分な空きができる。

【００７３】

入力がチャネル端Ｃで実行された場合、入力バッファファイル内Ｃ番目のエントリが読み取られて、Ｃ番目のエントリがデータを持っているかどうかを判断する。持っていれば、そのデータが取り出されて、その入力が完了する。持っていない場合、入力スレッドは一時停止され、引き続く出力命令により解放され、Ｃの入力バッファに十分なデータの書込みが行われる。

【００７４】

スレッドスケジューラ１８は、各スレッドについて１つのエントリを含む「一時停止テーブル」を維持している。このテーブルは、一時停止されているチャネル端（あれば）の記録に使用するものである。チャネル端ｃへの入力が完了するたび、またはチャネル端ｃに関連するチャネルへの出力が完了するたびに、このテーブルがチェックされ、もしｃのために一時停止されているスレッドがあれば、そのスレッドは解放される。

【００７５】

ＥＯＭトークンがｃを介して出力される場合、その出力ファイル内ｃ番目のエントリは、そのチャネル端が既に接続されていないことを記録するよう修正される。

【００７６】

必要となる論理量を低減するため、好ましくは、チャネルを初期化する命令は常時１つだけとし、所定の１チャネル端に動作するのに必要な通信命令も、常時１つだけにしておく。ただし、これは、複数のチャネルに対する動作の可能性を排除するものではない。

【００７７】

上述したチャネル端のシステムは、コード密度の面で特に効率がよい。というのも、スレッド間通信の制御および実行のために各スレッドが有する命令数が低減されて、その機能の大半はハードウェアのチャネル端で実行され、しかもＤＭＡやメモリ２４へのアクセスが不要となるからである。

【００７８】

再度ポート２２に関して、チャネルで発生しているアクティビティの検知を容易にするために、各チャネル端４２の入力バッファ２４をレジスタ類３８’に関連づける。レジスタ３８’は、関連スレッドの識別を格納するスレッド識別子（ＴＩＤ）レジスタ、およびイベントの発生と同時に実行を再開すべきプログラム内の位置を格納する継続点ベクトル（ＶＥＣＴＯＲ）レジスタを含む。このＴＩＤレジスタおよびＶＥＣＴＯＲレジスタを、ポート２２の場合と同様、イベントに応じてスレッドをスケジュールするため、スレッドスケジューラ１８および実行部１６が使用できる。つまり、レジスタは、イベントを設定するため、スレッドごとのスレッド識別子および継続点ベクトルを格納し、待機命令によりそのスレッドを中断し、イベントが発生したら、その継続点ベクトルが指定するコード内位置に戻る。この場合のイベントは、チャネル端４２へのデータの入力である。ＶＥＣＴＯＲレジスタを使用すれば、チャネルは割込みを生成することが可能となる。このチャ

ネル端はまた、チャネルをイネーブルしてイベントを生成させるイネーブルフラグ３９’を有する。好適な実施形態の一部において、このチャネル端４２をＣＴＲＬレジスタおよびＤＡＴＡレジスタに設けなくてもよいが、これは、その可能性を排除するものではない。

【００７９】

通信の遅延を最小限に押さえるため、有利なことに、チャネルにデータを転送する入力および出力命令をスレッドスケジューラ１８に直接出せることに留意されたい。つまり、その命令は、実行部１６に実行されると、そのチャネル用のレディビット３７’がその時点でチャネルの準備ができていないことを示している場合、スレッドスケジューラ１８に関連スレッドを実行セットから外すことで一時停止をさせる。同様に、イベントが発生していない場合、スレッドのイベントイネーブルフラグＥＥがスレッドのステータスレジスタＳＲ内にセットされていない場合、かつ／またはチャネル端のイベントイネーブルフラグがアサートされていない場合、イベントベースの待機命令がスレッドスケジューラにスレッドの実行を中断させる。

【００８０】

チャネルは、異なるプロセッサ上のスレッド間にも形成できる。図８は、複数のプロセッサ１４上に位置するスレッドレジスタ間にチャネルを形成するためのインターコネクトノード２０４を含むタイル２０２を例示している。このプロセッサ１４はみな、上述したタイプのものである。このインターコネクトシステムは、ポートとは別に、同一基板またはチップ上にあるプロセッサ間に設けられた直接的なハードウェアによるリンクである。このリンクを、基板および／またはチップで使用するシリアルインターコネクトおよびパケットルーティング機構としてもよい。ただし、当業者には明らかなように、他の型のインターコネクトシステムも使用可能である。こうしたシステムは、極めて低い電力、低いピンアウト、および使いやすさを求めた構成となっていると好ましい。

【００８１】

各インターコネクトノード２０４は、図２に例示するように、システムスイッチ２１６と、他の同様のタイル２０２をアレイとして互いに接続するのに使用可能なシステムリンク２１８とを含む。このように接続された１つまたは複数のノード２０４がインターコネクトシステムを構成する。プロセッサが異なれば、そのメモリのサイズもそれぞれ目的用途に合わせることができ、同じサイズに統一する必要はない。タイル同士は、同一チップ上にあっても、異なるチップ上にあってもよい。各ノード２０４も、１つのプロセッサ１４につき１つのプロセッサスイッチ２１４を含む。各プロセッサスイッチは、プロセッサリンク２２０を介してシステムスイッチ２１６と、そしてチャネルリンク２２２を介してプロセッサ１４のチャネル端４２との間を接続している。

【００８２】

インターコネクトシステムでの電力消費量を最小限に抑えると有益であることに留意されたい。複数の実施形態において、本発明のインターコネクトシステムは、電力排出のない静止状態を有し、受信側のクロック、位相同期回路または同期追従回路をサンプリングする必要がない。好ましくは、アレイ２００などのインターコネクト２４を含むシステムは、リンク２１８、２２０、２２２にデータが到達し始めた時点でのみ電力が供給される構成要素を用いる。

【００８３】

各リンク２１８、２２０、２２２は４本のワイヤを使用している。すなわち、各方向に１本ずつの論理１ワイヤおよび論理０ワイヤである。ビットは、「２線式非ゼロ復帰」方式で送信される。すなわち、論理１ビットの信号は、論理１ワイヤ上の遷移により伝送され、論理０ビットの信号は、論理０ワイヤ上の遷移により伝送される（すなわち、立ち上がり遷移または立ち下がり遷移がビット信号を伝える）。

【００８４】

プロセッサ１４間の通信はトークンにより発生し、このトークンは、通信制御用の制御トークンでも、通信される実際のデータを含むデータトークンでもよい。チャネルは、デ

ータトークンおよび制御トークンで構成されるメッセージを、チャネル端からチャネル端まで運搬する。各トークンは１バイトであると好ましい。このデータトークンはデータを含み、制御トークンは、インターコネクトのさまざまな局面を制御するため、通信プロトコルのコード化に使用される。スイッチ２１４、２１６はそれぞれ、チャネルの形成、制御、および閉鎖を目的として特定の制御トークン（下記参照）に作用するよう構成されたハードウェア切り替え論理を含んでいる。

【００８５】

１つのメッセージは、典型的にはデータトークンと制御トークンとの双方を含む、一連のトークンで構成されている。任意に、メッセージを、特定数のメッセージのトークンをそれぞれ含む複数のパケットに分割してもよい。メッセージまたはパケットの第１のトークンはヘッダトークンであり、このヘッダトークンは、宛先ノード、宛先プロセッサ、および宛先チャネル端を識別する宛先アドレスを含んでいる。メッセージまたはパケットの末尾のトークンは「メッセージ終了」ＥＯＭまたは「パケット終了」ＥＯＰトークンである。「メッセージ終了」ＥＯＤトークンも、パケット内の記述に利用可能である。ソフトウェアの開発者は、どのように選択しようとも、こうしたＥＯＭ、ＥＯＰ、ＥＯＤトークンを使用して、メッセージおよび／またはパケット内への通信内容配置をすることができる。このＥＯＭ、ＥＯＰ、ＥＯＤは、インターコネクトの切り替えの観点から、区別できない状態であることが好ましいが、ソフトウェア内で異なる使い方をすることは可能である。

【００８６】

プロセッサ１４はそれぞれ、上記で説明したようにスレッドレジスタ２０のセットを含む。それぞれのプロセッサでチャネル端に接続する場合、そのチャネルの使用方法には３つある。第１として、チャネルを単一のプロセッサ内で形成する場合と同様に「ストリーム用」チャネルを形成することができる。すなわち、各プロセッサのチャネル端を２本のスレッドそれぞれに割り当て、チャネル端ＩＤを交換することでそのチャネル端を接続し、そのチャネルを、連続したデータストリームの転送または複数のメッセージの転送に使用するのである。これにより、２本のスレッド間に回路を有効に形成することができ、このチャネルで送信される情報は、個々のトークンのストリームそのものとなる。第２として、パケットルーティングを行うために「パケット用」チャネルを使用することが出来る。この場合、チャネルを形成してメッセージまたはメッセージパケットを開始し、ＥＯＰまたはＥＯＭ制御トークンで非接続にすることで終了する。これにより、数多くの同時実行の通信間でそのインターコネクトを共有することができる。ここで送信される情報は、出力セットが入力のマッチングセットに対応した、明確なパケット構造を有するものである。このチャネルには、未知量のバッファリングが存在することになる。第３として、「同期用」チャネルが挙げられ、これは、パケット用チャネルと同様であるが、バッファがなく、通信を行うだけでなく、スレッドが同期化されたものである。

【００８７】

チャネルが形成されると、同一プロセッサのチャネル同様、このチャネル上でのＩ／Ｏおよびイベントの実行が可能となる。

【００８８】

動作時、ヘッダトークンは、システムリンク２１８またはプロセッサリンク２２０を介してシステムスイッチ２１６に受信される。このシステムスイッチ２１６は、その宛先ノードアドレスを読取り、そのアドレスがローカルノードアドレスと一致しない場合、そのパケットを別のノードにシステムリンク２１８経由で発送する。一方、宛先ノードアドレスがローカルノードアドレスと一致すれば、システムスイッチ２１６はその宛先プロセッサアドレスを読取り、そのパケットをプロセッサリンク２２０経由でローカルプロセッサ１４の１つに発送する。プロセッサスイッチ２１６は宛先チャネルアドレスを読取り、そのメッセージをチャネルリンク２２２およびインターコネクト４０を介して正しいチャネル端４２に発送する。

【００８９】

各リンク２１８、２２０、２２２は、制御レジスタ（図示せず）を含む。図２および図８を再度参照すると、ヘッドトークンが各スイッチ２１４および２１６を通過すると、ヘッダトークンによりスイッチの切替ロジックが開始されて、ソースリンクから動的に割り当てられた対象リンクまでのルートが作成される。この作成は、対象リンクアドレスを制御レジスタに書込み、ソースリンクアドレスをその対象リンク用制御レジスタに書き込むことにより行われる。ルートができると、トークンはすべてそのルートに沿って送信されるようになる。このルートは、ＥＯＰ、ＥＯＭ、またはＥＯＤトークンがこのルートに沿って送信された時点で非接続となる。ＥＯＰ、ＥＯＭ、またはＥＯＤがスイッチ２１６および２２０をパスした時点でそのルートの各段階が非接続となる。

【００９０】

具体的には、データｄが別のプロセッサに位置する未接続のチャネル端に出力されると、リンク２２２の１つが動的にそのチャネルに割り当てられ、そのアドレスｄ（上述したように、このデータはヘッダ、すなわち宛先チャネルのアドレスである）のインターコネクトスイッチへの転送に使用される。使用したリンクの識別子は、チャネル出力ファイルにｃ番目のエントリに関連付けられて書き込まれる。これに引き続くｃ経由の出力は、ｃ番目のエントリにアクセスして、その出力データの転送に使用すべきリンクを特定する。そのリンクのバッファが満杯であれば、出力スレッドは一時停止される。これが再度解放されるのは、そのリンクがデータを転送して、別の出力用のバッファを有するようになった時点である。これは、一時停止テーブルを用いて行われる。

【００９１】

データが未接続リンクに到達すると、これが宛先チャネル端ｄのアドレスであると翻訳され、このアドレスが、そのリンクに関連するレジスタに記録される。これに引き続きこのリンク経由のデータは、チャネル端ｄの入力バッファに書き込まれる。チャネル端ｄ（すなわち、チャネル入力ファイル内のｄ番目のエントリ）のバッファが満杯の場合、そのリンクレベルのフロー制御が働いて、スレッドが十分なデータを入力して入力バッファに余裕ができるまで、スイッチによるその後のデータ送信を阻止する。

【００９２】

チャネル端ｄで入力が行われると、入力バッファがデータを含むかどうかを読み取られる。含む場合、そのデータは取り入れられて入力は完了する。含んでいない場合、入力スレッドは一時停止され、そのリンクがｄの入力バッファに十分な新たなデータを供給した時点で解放される。

【００９３】

最後のＥＯＭ、ＥＯＰまたはＥＯＤトークンがｃ経由で出力されると、ＥＯＭ、ＥＯＰまたはＥＯＤがスイッチに転送され、その出力ファイルのｃ番目のエントリが、そのチャネルはもう接続されていないとの記録内容に書き換えられる。そのリンクがＥＯＭ、ＥＯＰまたはＥＯＤを受信すると、これがｄに転送され、リンクは非接続となる。

【００９４】

ここで注目すべきは、有利なことに、チャネル端の同じ機構が、同一プロセッサのスレッド間の通信にも異なるプロセッサのスレッド間にも使えることである。重要なことに、これはつまり、チャネル端のアドレス（すなわち、チャネル端ＩＤ）がシステム全体に使用可能ということである。言い換えると、各チャネル端ＩＤは、相互接続されているプロセッサのシステム全体で、たとえばアレイ２００内で、一意なのである。したがって、システム全体で資源を効率よく共有でき、プログラミングが容易となる。

【００９５】

複数のチャネル端およびリンクを、複数のスレッドで共有可能である。これは、１つのチャネル端を複数のスレッドからのメッセージ受信に使用できるようにするために有益である。このために、各入力チャネル端は、目下使用中であるかどうかを示す被要求フラグ４３を有する。ヘッダが出力されたメッセージのスタート時点でそのチャネル端が使用中であることがわかった場合、出力スレッドは一時停止される。これが解放されるのは、ＥＯＭ、ＥＯＰまたはＥＯＤが次にそのチャネル端を非接続としたとき（したがって、新た

な接続に利用可能となったとき）である。同様の機構をリンク２１８、２２０、２２２それぞれに用いて、これらのリンクを複数の出力スレッド間で共有することができる。

【００９６】

これらのチャネルが双方向性であることにも留意されたい。各チャネル端に入力機能および出力機能（ステータスバッファおよびデータバッファ）があるため、入出力双方に同時に使用可能である。つまり、どのチャネルも完全に独立した単一方向性チャネルの一対として使用可能であり、異なるプロセッサにおけるスレッド間で動作する複数のチャネルの場合、その動作方向は互いに反対になる。別法として、１本のチャネルを２つのスレッド間で双方向性通信経路として使用することもできる。この場合、通信方向はスレッドの進行方向に応じて変更される。

【００９７】

さらに、形成後は、１つの識別子（ローカルチャネル端ＩＤ）を双方向性チャネルの識別に使用でき、ローカルチャネル端ＩＤおよびリモートチャネル端ＩＤの双方を使用する必要はないことにも留意されたい。チャネル端４２の集合体の提供と合わせ、これにより、チャネル通信の効率が格段に向上する。単一の識別子の使用は、次の機能により簡単になる。

（１）ローカルチャネル端の宛先ヘッダを、チャネル端識別子レジスタＣＥＩＤ４１を用いて格納する。レジスタＣＥＩＤ４１の設定を明確にする命令を出してもよい。これを以下で説明するＳＥＴＤ命令としてもよい。

（２）非活動の（すなわち未接続の）チャネルに出力が行われる毎、プロセッサのスイッチ２１４に、レジスタＣＥＩＤ４１から、別の命令により出力されるべきヘッダではなく、上記ヘッダをまず自動的に送信させる。非活動のチャネルとは、最後に非接続となって以来、出力を行っていないチャネルである（ＥＯＭが、応答としてなど、送信された唯一のトークンであっても、そのヘッダがまず自動的に出力される）。

（３）ＥＯＭ（またはＥＯＰかＥＯＤ）トークンにより、そのチャネルを非活動（すなわち非接続）状態に戻す。

【００９８】

これにより、チャネルを適時にセットすることができる。すなわち、プログラム内で宣言されたら、ローカルチャネル端アドレスだけをチャネルの識別のために巡回させればよい。これは、双方向性チャネルの場合も同じであり、スレッドは、送信にも受信にも１つの識別子を使用すればよい。

【００９９】

プロセッサ間の通信トークンについては、後に詳述するが、まず、ポートおよびチャネルを制御する命令について詳しく説明する。インターフェースプロセッサは、そのスレッドベース構造により、いくつかのプログラミング方法をサポートすることができる。標準入出力を行う１つの従来型プロセッサとして扱うこともできるし、あるいは、数百の通信部品の平行アレイの一部としてプログラムしてもよい。こうしたオプションを支援するものとして、命令セットが提供されている。この命令セットには、初期化、終了、開始および停止スレッドを支援し、入出力通信を提供する専用命令が含まれる。これらの入出力命令により、外部デバイスとの迅速な通信が可能となる。また、高速かつ低遅延の入出力およびハイレベルな同時実行プログラミング技術が支援される。本明細書における、この命令のポートおよびチャネルアクティビティの処理への応用については、本発明の実行に使用可能な命令の例を挙げながら、以下で詳述する。

【０１００】

資源は、必要な資源の種類を指定するＧＥＴＲ命令を用いてスレッド用に確保され、ＦＲＥＥＲ命令で再度解放される。

【０１０１】

ポートは、入力モードまたは出力モードで使用可能である。入力モードでは、スレッドをパスするデータにフィルタをかけるため、１つの条件を使用することができる。以下に説明するように、データが利用可能になると、ポートはイベントまたは割り込みを生成す

る。これにより、スレッドは、いくつかのポートを監視し、準備のできたポートのみを使用することができる。次いで、入出力命令ＩＮおよびＯＵＴを用いて、準備のできたポートとの間でデータを転送する。この場合、ＩＮ命令は、ｎビットポートから最下位ビットを入力し、ゼロ拡張する。一方、ＯＵＴ命令はｎ最下位ビットを出力する。

【０１０２】

さらに２つの命令、ＩＮＳＨＲおよびＯＵＴＳＨＲが、データの転送を最適化する。ＩＮＳＨＲ命令は、レジスタの内容を右にｎビットシフトし、左端のｎビットをｎビットポートから入力されたデータを埋める。ＯＵＴＳＨＥＲ命令は、データのｎ最下位ビットをｎビットポートに出力し、レジスタの内容を右にｎビットシフトする。

【０１０３】

```
OUTSHR port, s  port ≪s[bits 0 for width(port)] ；ポートから出力
                  s ← s >> width(port)            およびシフト

INSHR port, s    s ←s >> width(port)；シフトおよび
                 port ≫ s[bits (bitsperword - width(d)) for width(d)]；ポートから入力
```

このとき、「≫」は入力、「≪」は出力を表す。

【０１０４】

ポートは、使用前に構成を決定しておかなければならない。この構成には、ポートのいくつかの独立したセッティングを定義するＳＥＴＣ命令を使う。セッティングはそれぞれデフォルトモードを持っているため、異なるモードが必要な場合にのみ構成の必要が生じる。

```
SETC port, mode    port[ctrl] ← mode       ポート制御を設定
```

【０１０５】

ＳＥＴＣモードセッティングの効果を以下に説明する。各セッティングの第１のエントリがデフォルトモードとなる。

【０１０６】

モード	効果
ＯＦＦ	ポートは非活動状態；ピンはハイインピーダンス
ＯＮ	活動状態
ＩＮ	ポートが入力口
ＯＵＴ	ポートが出力口（ただし、入力により目下のピン値を返送）
ＥＶＥＮＴ	ポートがイベントを発生させる
ＩＮＴＥＲＲＵＰＴ	ポートが割込みをかける
ＤＲＩＶＥ	ピンをハイおよびローに駆動
ＰＵＬＬＤＯＷＮ	ピンを０ビット用にプルダウン、それ以外はハイインピーダンス
ＰＵＬＬＵＰ	ピンを１ビット用にプルアップ、それ以外はハイインピーダンス
ＵＮＣＯＮＤ	ポートは常にレディ状態；　インプットは直ちに完了
ＥＱＵＡＬ	ポートは、その値がそのデータ値と等しくなればレディ状態
ＮＥ	ポートは、その値がそのデータ値と異なればレディ状態
ＴＲＡＮＳＩＴＩＯＮ	ポートは、その値がそのデータ値に向けて変化するとレディ状態
ＧＲ	ポートは、その値がそのデータ値を超えればレディ状態
ＬＳ	ポートは、その値がそのデータ値を下回ればレディ状態

【０１０７】
ＤＲＩＶＥ、ＰＵＬＬＤＯＷＮおよびＰＵＬＬＵＰモードは、ポート方向がＯＵＴのときに関連するだけである。ＴＲＡＮＳＩＴＩＯＮ条件は、１ビットのポーにのみ関連し、ＧＲおよびＬＳ条件は、１ビットを超えるポートにのみ関連する。
【０１０８】
各ポートは、レディビット３７を有する。このビットは、データフローの制御に使われ、そのポートが入力命令または出力命令を完了できるかどうかを定義する。レディビットの設定は、ポート構成により変更可能である。ＳＥＴＣ、ＳＥＴＤまたはＳＥＴＶが実行されると、レディビットはクリアされる。
【０１０９】
入力モードにあるポートを、条件付き入力を実行する構成にすることができる。その条件にあうデータのみがプログラムに返送されるよう、入力データは、この条件のフィルタにかけられる。条件が設定されたら、ポートが準備できた段階でＩＮおよびＩＮＳＨＲ命令を完了させるだけでよい。上述したように、準備のできていないポートに入力命令を実行するとスレッドが一時停止される。この場合、ポートは、準備ができたら、レディビットを設定する。これがスレッドスケジューラに伝送される。すると、スレッドスケジューラは、実行部１６のパイプライン内にある関連命令をリスタートさせるか、その命令を再実行する、すなわち、パイプライン内に再発行することにより、スレッドを再開する。ポートの準備ができたら、データが返送され、レディビットはクリアされる。
【０１１０】
ポートのレディビットが設定されたら、ソフトウェアがその条件に満たした値を得られるように、たとえそのポート上の値がその後変化していたとしても、条件を満たしたデータ値がキャプチャされる。ＩＮおよびＩＮＳＨＲ命令が実行され、レディビットが設定されたら、データが返送され、レディビットがクリアされる。レディビットを設定されていない場合、レディビットが設定されるまでデータは一時停止される。条件が設定されたら、データはその条件と比較され、条件が満たされた段階でレディビットを設定するだけである。
【０１１１】
ＯＵＴまたはＯＵＴＳＨＲ命令が実行されたとき、レディビットがなければデータはポートに取り入れられて、レディビットが設定される。レディビットが設定されたら、これがクリアされるまで、スレッドは一時停止される。
【０１１２】
スレッド間の通信はチャネルを使って行われる。このチャネルにより、端部間の全二重データ転送が可能となる。この場合の端部は、同一プロセッサ内にあっても、同一チップ上の複数のプロセッサ内にあっても、複数のチップ上のプロセッサ内にあってもよい。チャネルは、２つのチャネル端の間をデータトークンと制御トークンで構成されたメッセージを運搬するものである。制御トークンは、通信プロトコルの符号化に使用される。大半の制御トークンはソフトウェアで利用可能であるが、数は、インターコネクトハードウェアに使用されるプロトコルの符号化に確保されており、命令による送受信ができない。
【０１１３】
以下で説明するように、データが利用可能となると、チャネル端をイベントおよび割込みの生成に使用することができる。これにより、スレッドはいくつかのチャネルおよび／またはポートを監視することができ、準備のできたものだけを使用する。
【０１１４】
２つのスレッド間で通信を行うためには、２つのチャネル端それぞれを１つのスレッドに割り当てなければならない。これを、GETR　CHAN命令で行う。次いで、第１のスレッド用のチャネル端の識別子を第２のスレッドに与え、逆も同様に行う。これにより、２つのスレッドは資源識別子を使って、入力命令および出力命令によるメッセージの転送ができるようになる。
【０１１５】

```
OUTT          d ≪ s          トークン出力
OUTCT         d ≪ s          制御トークン出力
INT           d ≫ s          トークン入力
OUT           d ≪ s          データワード出力
IN            d ≫ s          データワード入力
TESTCT        d ←isctoken(s)          制御トークン用テスト
TESTWCT       d ← hasctoken(s)  制御トークン用テストワード
```

【０１１６】

各メッセージは、他のスレッドの資源識別子を含むヘッダを先頭としている。この次に通常、一連のデータまたは制御トークンが続き、最後に終了またはメッセージ（ＥＯＭ）制御トークンがくる。ＯＵＴおよびＩＮ命令を氏用意して、チャネル内のデータワードの送信が行われる。データのバイト送信には、ＯＵＴＴ、ＩＮＴＴ、ＯＵＴＴＳＨＬおよびＩＮＴＴＳＨＬ命令が使用される。ＯＵＴＴＳＨＬおよびＩＮＴＴＳＨＬは、ワードの最重要バイトで始まる通信を最適化するために使用されるシフト命令であり、主に、メッセージヘッダ内のルーティングアドレス形成に用いられる。

【０１１７】

```
OUTTSHL channel, s  channel ≪ s[bits (bps - 8) for 8];  チャネルから出力
                      s ← s << 8                          およびシフト
```

【０１１８】

```
INTSHL channel, s    s ← s << 8;                          チャネルからシフト
                   channel ≫ s[bits 0 for 8]               および入力
```

【０１１９】

チャネル端は、少なくとも１ワードをバッファできるように、十分なトークンの保持が可能なバッファを有する。チャネルが満杯でデータを受け取れないときに出力命令が実行されると、その命令を実行されたスレッドが一時停止される。この再開は、その命令を完遂するのにチャネルに十分な余地ができたときである。同様に、その命令が実行されても利用可能なデータが十分にないときも、スレッドは一時停止され、この再開は、十分なデータが利用可能となったときである。

【０１２０】

チャネル上への制御トークンの送信には、ＯＵＴＣＴ命令が使用される。制御トークンは、そのチャネル内の１バイトの記憶を取り出す。受信側端部では、ソフトウェアが、ＴＥＳＴＣＴ命令を使用してその次のバイトが制御トークンであるかどうかをテストし、少なくとも１トークンが利用可能となるまで待つ。ＴＥＳＴＷＣＴ命令を使用して、次のワードが制御トークンを含むかどうかをテストすることも可能である。この命令の場合、少なくとも１つの制御トークンが受信されるまで、またはデータワード全体が受信されるまで待つ。

【０１２１】

トークンが制御トークンであるとテストしたら、そのトークンはＩＮＴで受信される。受信後、おそらく、それが制御トークンであったかどうかをチェックする方法はない。チャネル端がデータトークンおよび制御トークンの双方を含む場合、ＩＮ命令によりそのトークンはすべてデータとして返送される。

【０１２２】

不要となったチャネルを、ＦＲＥＥ　ＣＨＡＮ命令で解放することができる。解放しない場合は、別のメッセージに使用してもよい。

【０１２３】

１システム内のインターコネクトは、チャネルすべてで共有される。プロセッサ内では、接続性に何ら制限もないため、インターコネクトの共有のためにチャネル端を互いに非接続にする必要はない。チャネル端を非接続としなければならないのは、対象のチャネル端が別のチャネル端と共有される場合だけである。

【０１２４】

しかし、異なるプロセッサ上のチャネル端に接続する際には、そのインターコネクトを確実に、そのシステム内で他のチャネルに効率よく共有された状態にすると有用である。これは、送信されるデータを複数のパケットおよびメッセージに分解することで行う。各パケットまたはメッセージは、ヘッダを先頭とし、終了パケット（ＥＯＰ）またはＥＯＭ制御トークンを末尾とする。

【０１２５】

イベントおよび割込みを使用すると、資源(ポートおよびチャネル)は自動的に、制御を所定のイベントハンドラに転送する。イベントや割込みを許容するスレッドの機能は、スレッドステータスレジスタＳＲ（図４参照）内の情報で制御され、ＴＳＥおよびＴＳＤ命令で明確に制御できる。この情報には、イベントイネーブルフラグ（ＥＥ）および割込み有効フラグ（ＩＥ）が含まれる。

【０１２６】

TSE s　　SR ← SR ∨ s　　スレッド状態イネーブル
TSD s　　SR ← SR ∧ ¬s　　スレッド状態ディセーブル

【０１２７】

上記命令のオペランドは以下のうちの１つとする。

【０１２８】

EE　　イベントをイネーブルまたはディセーブルする
IE　　割込みをイネーブルまたはディセーブルする

【０１２９】

イベントは、これがセッティングされた適用範囲と同じ範囲内で処理される。したがって、イベント発生時、スレッドの状態は有効であり、そのイベントに迅速に反応できる。このスレッドは、イベントを立ち上げたポートを使用して、そのイベントの一部またはすべての情報を変更しないまま、入出力動作を実行することができる。これにより、スレッドはイベントの処理を完遂し、すぐに次の同様のイベントの待機状態に入ることができる。

【０１３０】

イベントハンドラのプログラム位置を、ＳＥＴＶ命令を使用して、イベントをイネーブルする前に設定しなければならない。ポートには、いつイベントを生成したよいかを判断する条件があり、この条件はＳＥＴＣおよびＳＥＴＤ命令で設定される。チャネルは、十分なデータを含んだ時点、または出力用データを受け入れる余地ができた時点で準備ができたと見なされる。

【０１３１】

特定ポートまたはチャネルによるイベントの生成は、イベントイネーブル無条件（ＥＥＵ）命令でイネーブルにされ、イベントディセーブル無条件（ＥＤＵ）命令でディセーブルにされる。イベントイネーブルｔｒｕｅ（ＥＥＴ）命令により、その条件オペランドがｔｒｕｅであればイベントがイネーブルにされ、ｔｒｕｅでなければディセーブルにされる。これとは逆に、イベントイネーブルｆａｌｓｅ（ＥＥＦ）命令により、その条件オペランドがｆａｌｓｅであればイベントがイネーブルにされ、ｆａｌｓｅでなければディセーブルにされる。これらの命令を使用して、ガード付き入力の実行を最適化する。以下に、ポートにイベントを構成するための命令書式の数例を挙げる。同じ命令をチャネルに関しても適用可能であることを理解されたい。

【０１３２】

SETV port, v　port[vector] ← v　　イベントベクトル設定
SETD port, d　port[data] ← d　　イベントデータ設定
SETC port, c　port[ctrl] ← c　　イベント制御設定

【０１３３】

EET port, b　port[enable]← b; port[tid] ← thread　イベントイネーブルｔｒｕｅ
EEF port, b　port[enable]← ¬b; port[tid] ← thread　イベントイネーブルｆａｌｓｅ

```
EDU  port    port[enable]← false; port[tid] ← thread  イベントディセーブル
EEU  port    port[enable]← true; port[tid] ← thread  イベントイネーブル
```

【０１３４】

１つまたは複数の資源に対してイベントをイネーブルしたら、スレッドはＷＡＩＴＥＵ命令を使用して、少なくとも１つのイベントの発生を待つことができる。この結果、ＥＥ（イベントイネーブル）フラグをクリアしてイベント類をディセーブルにした後、対応するイベントベクトルに指定されたイベントハンドラに制御が転送されて即座に新たなイベントが発生する可能性がある。あるいは、イベントが発生するまでスレッドを中断してもよい。この場合、ＥＥフラグはイベント発生とともにクリアされ、スレッドは実行を再開する。

【０１３５】

```
WAITET b    if b then SR[EE] ← true              ｔｒｕｅならイベント待機
WAITEF b    if ¬ b then SR[EE] ← true            ｆａｌｓｅならイベント待機
WAITEU      SR[EE] ← true                        イベント待機
```

【０１３６】

```
CLRE        SR[EE] ← false;                      イベントすべてをディセーブル
              forall port                          スレッド用
              if port[tid] = thread then port[enable] ← false
```

【０１３７】

条件がみたされて１つまたは複数のイベントが発生するのを繰り返し待つという一般的なケースを最適化するため、イベント待機命令の条件付き形式が提供されている。ＷＡＩＴＥＴ命令は、その条件オペランドがｔｒｕｅであれば待機し、ＷＡＩＴＥＦは、その条件オペランドがｆａｌｓｅであれば待機する。

【０１３８】

スレッドによってイネーブルにされたイベントはすべて、１つのＣＬＲＥ命令でディセーブルにできる。この命令は、スレッドによってイネーブルにされたイベントを有していたポートすべてにおけるイベント生成をディセーブルにする。ＣＬＲＥ命令はまた、そのスレッドのステータスレジスタにあるイベントイネーブルのステータスをクリアする。

【０１３９】

優先度の高い資源に対してスレッドを最適に反応させるため、TSE　EE命令を使用して、ポートおよび／またはチャネルをイネーブルにすることを開示し、イベント待機命令の１つを使用する前に、まずスレッド上のイベントをイネーブルにすることができる。これにより、プロセッサは資源を優先順に走査することができる。その結果、イベントは、イネーブルにされてすぐ、直ちに処理される可能性がある。

【０１４０】

イベントとは異なり、割込みは目下の適用範囲内では処理できないため、使用中のＰＣおよびＳＲ（場合によって、他のレジスタの一部またはすべても同様）を割込みハンドラを実行する前にセーブしておかなければならない。資源ｒにより割込みが生成されると、以下が自動的に発生する。

【０１４１】

```
SAVEPC ← PC;
SAVESR ← SR;
SR[EE] ← false;
SR[IE] ← false;
PC ← r[vector]
```

【０１４２】

ハンドラの処理が完了すると、ＲＦＩＮＴ命令により、割り込まれたスレッドの実行が行われる。

【０１４３】

```
RFINT       PC ← SAVEPC;割込みから復帰
```

SR ←SAVESR

【０１４４】

割込みは、イベントを待って中断されているスレッドに割込むことができる。

【０１４５】

ここで、プロセッサ間通信に戻り、この通信で使用するデータトークンおよび制御トークンの詳細を説明する。上述したように、リンク２１８、２２０、２２２はそれぞれ４本のワイヤを使う。すなわち、各方向に１本ずつの論理１ラインおよび論理０ラインである。ビットは、２線式非ゼロ復帰方式で送信される。すなわち、論理１ビットの信号は、ライン１上の遷移により伝送され、論理０ビットの信号は、論理０ライン上の遷移により伝送される。実際のデータは、１０ビットトークン内で８ビットをそれぞれ有するデータトークンで送信され、制御情報は、やはり１０ビットトークン内で８ビットをそれぞれ有する制御トークンで送信される。どちらのレールも、各トークンの末尾で（エラーがなければ）、休止（ゼロ）状態に戻る。

【０１４６】

データ（および制御）は、同時に双方向に運搬される。これらのトークンを用いて、さまざまな長さのパケットまたはメッセージを移動させることができる。制御トークンによって、物理的なリンク制御用に確保されたり、ソフトウェアリンク制御（高次層プロトコル）のためのソフトウェアに利用されたりする。

【０１４７】

制御トークンのコーディングは、トークンの最後で必ずリンクが静止状態に戻るように設計されている。トークンは次のように、そして図９で概略を例示したように符号化される。

【０１４８】

トークン９００はいずれも、情報部分９０１および第１の追加ビット９０２で構成される第１の部分を含む。この情報部分は、好ましくは、１バイト（８ビット）であり、そのトークンが有する実際のデータまたは制御情報である。第１の追加ビットは、そのトークンがデータトークンであるか制御トークンであるかを示すものである。

【０１４９】

したがって、第１の部分は９ビット長となり、奇数である。この奇数ビットの送信に続いて、２つの可能性がある。

（ａ）１つは、奇数の論理０ビットおよび偶数の論理１ビットの送信である。この場合、ライン０は奇数の遷移により高電圧となり、ライン１は偶数の遷移により低電圧となる。

（ｂ）もう１つは、偶数の論理０ビットおよび奇数の論理１ビットが送信されることである。この場合、ライン０は偶数の遷移により低電圧となり、ライン１は偶数の遷移により高電圧となる。

【０１５０】

そこで、確実にリンクを静止状態に戻すためには、すなわち、確実にライン０およびライン１を低電圧にするには、第２の部分、この場合は第２の追加ビット９０３を各トークン９００に含める。上記（ａ）では、第２の追加ビットは論理０であり、上記（ｂ）では、第２の追加ビットは論理１である。いずれの場合も、トークン内の０および１の総数は偶数になるため、リンクはその静止状態に戻る（ライン０もライン１も、トークンの送信前は低電圧で始動するものとする）。

【０１５１】

第１の部分が奇数個のビット（この場合、情報ビット９０１および第１の追加ビット９０２の１バイト）を有する場合、第２の追加ビット９０３は、第１の部分のビットごとにＸＯＲ演算をするだけで大変効率よく算出できる。算出を高速化させるため、この演算を、ソフトウェアではなく、インターコネクト２０４のＸＯＲ論理回路またはプロセッサ１４で行うことが好ましい。

【０１５２】

送信の順序については、好ましくは、第１の追加ビット９０２をまず送信し、続いて情

報部分９０１、第２の追加ビット９０３とする。

【０１５３】

ただし、第１および第２の追加ビット９０２および９０３の位置はどこであってもあまり問題はないことに留意されたい。第１および／または第２の追加ビットがトークンの先頭、末尾、またはその間のどこかにあったとしても、受信側が各ビットの場所を把握していれば問題ない。

【０１５４】

上記は、奇数個のビットを有する第１の部分（すなわち情報部分プラス第１の追加ビット）について説明した。しかし、第１の部分が偶数個のビットを有する場合には（例えば、第１の追加ビットがない、または奇数の情報ビットが使用されている）、リンクを静止状態に戻すため、第２の部分を２ビットとして計算できることに留意されたい。

【０１５５】

また、各方向に２本のワイヤのみを用いる上記２線式は比較的基本的な実施形態であるため、任意方向におけるシリアル通信のみしかできない。リンクをむしろ、低速シリアルモードと高速広域モードとの間で切り替え可能とすると好ましい。広域モードでは、以下に説明するように、２本ワイヤによるシリアル方式とは異なる符号化方式を用いる。

【０１５６】

広域モードでは、各リンク２１８、２２０、２２２が「５線式(one-of-five)」コードを用いる。５線式コードによると、５本のワイヤが各リンクの各方向に用意される（したがって、各リンクにつき双方向を含めると、全部で１０本）。コードは、１回の送信につき、この５本のワイヤの１本に遷移を伝送することにより、そのリンク送信される。これにより、合計５つのコードがそれぞれの意味をもって対応付けられることになる（２ワイヤ方式の２つのコードとはこの点で異なる。２本による「２線式」コードでは、論理１か論理０かにそのまま対応付けられるだけである）。このコードの対応付けとして、例えば以下が挙げられる。

【０１５７】

コード	意味
００００１	値００
００００１０	値０１
００１００	値１０
０１０００	値１１
１００００	エスケープ

（上記「コード」における「１」は、そのリンクの各ワイヤ上に立ち上がりまたは立ち下がり論理遷移があることを示し、「０」は論理遷移がないことを示す。）

【０１５８】

この一連のコード、すなわち「記号」を使ってトークンを符号化する。４サイクルで、5^4、すなわち６２５トークンをコーディングできる。これはデータトークンおよび制御トークンの双方の符号化に十分である。例えば、v_iが記号ｉでの値であり、e_iが記号ｉでのエスケープコードであるとすると、以下のようになる。

【０１５９】

トークン	使用
$v_0v_1v_2v_3$	２５６個のデータトークン
$e_0v_1v_2v_3$	６４個の制御トークン　１９２～２５５
$v_0e_1v_2v_3$	６４個の制御トークン　１２８～２１９１
$v_0v_1e_2v_3$	６４個の制御トークン　６４～２１２７
$v_0v_1v_2e_3$	６４個の制御トークン　０～２６３

【０１６０】

ただし、１つ以上の記号がエスケープである追加コードがいくつかある場合もある。こうしたコードは、ＥＯＭなどの特定の制御トークンをコーディングするのに使用可能である。例えば$e_0e_1v_2s_3$であれば、v_2でＥＯＭを伝送し、S_3を選択して、信号線

すべてを０にしてリンクを静止状態に戻す。この追加記号S_3は何でもよく、５本のラインのいずれかで並行している１つ以上の遷移に有効である（５本のいずれか１本のみにある単一遷移に制限されるコード値ｖおよびｅやエスケープとは異なる）。つまり、S_3の遷移は、ラインを静止状態にリセットするために必要に応じて選択されるものである。

【０１６１】

上記の例を用いると、２つの連続したエスケープコード$e_0 e_1$によって常に、第１のワイヤ上の遷移は偶数となり、他のワイヤには遷移がない状態となる。そこで、コード値v_2が、第２から第５のうちの１ワイヤ上で臨時遷移となる。したがって、例えば、５本とも低電圧論理値で開始されると、$e_0 e_1$はワイヤすべてを低電圧論理値に保ち、これに続くv_2が、第２から第５のワイヤの１本を高電圧論理値とする。そして最後にv_2と同じものとしてコードS_3が選択されて、同じワイヤ上で別の遷移を実行し、低電圧論理値に戻す。しかし、すべてのワイヤが低電圧論理値で開始されたわけではない場合、ワイヤは全体にどの状態にあってもよく、必要に応じて５本のラインのうちいずれかまたはすべてに１つ以上の並行遷移を構成する場合など、適宜S_3を選択する。

【０１６２】

好ましくは、２ワイヤ式符号化と５ワイヤ式符号化との間のデータ変換はオンザフライで行うことが好ましい。

【０１６３】

この５ワイヤ方式を２ワイヤ方式の代替モードとして使用すると好ましいが、５ワイヤ方式単独での使用や他のモード方式に対する代替としての使用も可能である。

【０１６４】

例えば、別の方式として３ワイヤ方式を用いることができる。この方式では、各リンク２１８、２２０、２２２に各方向において３本のワイヤが使用され、データトークンおよび制御トークンの送信に「３線式」コードが用いられる。この場合、３本のワイヤのいずれか１本にのみ遷移を伝送することによりコードが送信される。一例として、複数対のコードを連続的に送信して、値およびエスケープを次のように符号化することができる。

【０１６５】

コード１	コード２	意味
００１	００１	値０００
０１０	００１	値００１
１００	００１	値０１０
００１	０１０	値０１１
０１０	０１０	値１００
１００	０１０	値１０１
００１	１００	値１１０
０１０	１００	値１１１
１００	１００	エスケープ

（再度、上記「コード」における「１」は、そのリンクの各ワイヤ上に立ち上がりまたは立ち下がり論理遷移があることを示し、「０」は論理遷移がないことを示す。）

【０１６６】

この方式では、６回のハンドシェイクサイクルで、3^6トークン、すなわち７２９トークンをコーディングできる。この場合もやはり、データトークンおよび制御トークンの双方の符号化に十分である。また、このエスケープコードの１つを使って、例えば、すべての信号ワイヤをゼロにするためにv_3、v_4、およびv_5を選択した$v_0 v_1 e_2 v_3 v_4 v_5$など、ＥＯＭなどのいくつかのゼロ復帰トークンを提供することができる。

【０１６７】

データトークンおよび制御トークンの例をここで挙げる。このトークンは、上述した２ワイヤ、３ワイヤ、５ワイヤ方式など、適した符号化方式であればいずれを用いても符号化が可能であることを理解されたい。

【０１６８】

従来、基板またはチップ上のプロセッサ間のインターコネクトは、ハードウェアのみで制御されるものであり、ソフトウェアには認識も対応もできるものではなかった。しかし、本発明の一部の態様によると、制御トークンを「アーキテクチャで定義される」（すなわちハードウェアで定義）ものか「ソフトウェアで定義される」されるものかに大別することができる。アーキテクチャで定義される制御トークンは、インターコネクト２０４内のスイッチ２１４、２１６またはリンク２１８、２２０、２２２の１つ以上に、トークンの値を検知し、これに反応して、インターコネクト２０４の一部側面を制御するように開始されるハードウェア論理が含まれるものである。つまり、アーキテクチャで定義される制御トークンの機能は、インターコネクト２０４内のハードウェア論理により予め定められるものである。このハードウェア論理は、トークンの宛先プロセッサ１４で稼働しているソフトウェアの助けを必要とすることなく、開始され、その機能を実行する。ところが、本発明は、ソフトウェアに、アーキテクチャで定義される特定の制御トークンへのアクセスを認めるものである。すなわち、アーキテクチャで定義される特定の制御トークンは、ソフトウェア開発者が定めた追加機能を実現するために、ソフトウェアにより翻訳できるようになっている。一方、ソフトウェアで定義される制御トークンは、トークン値を検知したり、これに作用したりというハードウェア論理がなく、代わりに、受信側プロセッサ１４で稼働するソフトウェアによってのみ翻訳されるものである。ソフトウェアで定義される制御トークンは、そのように定義されたロジックを含まないことから、インターコネクト２０４内のハードウェアで翻訳されることはない。

【０１６９】

一部の実施形態において、制御トークンは実際には４つのグループに分けられる。アプリケーショントークン、専用トークン、特権トークン、およびハードウェアトークンである。好ましくは、制御トークンの８ビット部分９０１において、値０～１２７を用いてアプリケーショントークンを符号化し、値１２８～１９１を用いて専用トークンを符号化し、値１９２～２３３を用いて特権トークンを符号化し、値２２４～２５５を用いてハードウェアトークンを符号化する。ただし、これ以外の組み合わせも、特定用途の用件に応じて可能である。制御トークンの４種類は以下の通りである。

（１）アプリケーショントークンはハードウェアにより翻訳されることはなく、ソフトウェアで定義される。これらは、コンパイラやアプリケーションソフトウェアによる使用を目的としており、データ構造の符号化および特定用途向けプロトコルの実施を促進するものである。

（２）専用トークンは、アーキテクチャで定義され、ハードウェアまたはソフトウェアで翻訳される。一般のデータ型および構造に対する符号化標準の提供、そして、データ、プログラム、およびチャネル（例えば）の転送用プロトコルの符号に用いられる。

（３）特権トークンは、アーキテクチャで定義され、ハードウェアまたはソフトウェアで翻訳される。ハードウェア資源の共有、制御、監視、およびデバッグなどのシステム機能の遂行に用いられる。このトークンの１つでも非特権ソフトウェアとの間で転送しようとすると、例外が発生する。

（４）ハードウェアトークンは、ハードウェアによってのみ使用される。このトークンの１つでもソフトウェアとの間で転送しようとすると、例外が発生する。

【０１７０】

本発明の一部の態様によると、制御トークンおよびデータトークンの両方を含むメッセージが、ソフトウェアで構成される。上述したように、基板またはチップ内の物理的インターコネクトの制御は従来、そのインターコネクト内の専用ハードウェアがその責任を担っていた。つまり、物理的インターコネクトを制御する信号の生成は、インターコネクト内ハードウェアが行っており、プロセッサ内のソフトウェアではなかった。このような制御例としては、スイッチおよびリンクの制御レジスタへのアクセスが挙げられる。しかし、本発明によれば、データトークンも制御トークンも、そしてアーキテクチャで定義される制御トークンもソフトウェアで定義される制御トークンも、実行部１６が実行する命令（ＯＵＴＣＴ命令）のオペランドからインターコネクト２０４へと出力することができる

。このオペランドは、命令自体から直接読み出される即値オペランドでも、関連命令に指定されるオペランドレジスタＯＰから読み出されるオペランドでもよい。好ましいオプションではないが別法として、命令で指定したメモリアドレスからデータトークンまたは制御トークンを読み出すことができる。ハードウェアトークンのみ、ソフトウェアに生成されることはなく、単独でインターコネクトハードウェア回路で内部使用される。

【０１７１】

ここで、制御トークンの種類別の例をいくつか説明する。アプリケーショントークンには所定の機能はなく、ソフトウェア開発者の選択する目的通りに使用可能である。上述したように、コンパイラやアプリケーションソフトウェアによる、データ構造の符号化および特定用途向けプロトコルの実施のための使用を想定している。

【０１７２】

専用トークンの例は次の通りである。

ＥＯＭ　　メッセージ終了
ＥＯＰ　　パケット終了
ＥＯＤ　　データ終了

ＲＥＡＤ　　遠隔メモリからの読み出し
ＷＲＩＴＥ　遠隔メモリへの書込み

ＡＣＫ　　処理が成功したことの確認
ＮＡＣＫ　　エラーとなったことの確認

【０１７３】

動的経路制御（すなわちパケット交換ルーティング）を用いる場合、接続部が１つまたは複数のヘッダトークンにより形成され、ＥＯＭ、ＥＯＰまたはＥＯＤトークンにより非接続にされる。ヘッダトークンは実際にはデータトークンであるが、スイッチ２１８、２２０、２２２は、接続されていないチャネル端４０から出力されたデータトークンをヘッダトークンであると認識するように構成された論理を含んでいることに留意されたい。

【０１７４】

ＥＯＭ、ＥＯＰおよびＥＯＤはアーキテクチャで定義される。というのは、このそれぞれが、インターコネクト２０４内でハードウェア論理をトリガして、宛先プロセッサ１４のソフトウェアのいずれとも独立した専用機能を発揮する、すなわち、スイッチ２１８、２２０，２２２をトリガしてチャネルを非接続にするからである。ＥＯＭ、ＥＯＰおよびＥＯＤは、インターコネクトハードウェアが関与している範囲では、互いに区別できるものではない。しかし、いずれもソフトウェアにもアクセス可能であるため、ソフトウェア開発者はＥＯＭ、ＥＯＰおよびＥＯＤを用いてソフトウェアに種々の意味を持たせることができる。例えば、ソフトウェア開発者は、ＥＯＰトークンでメッセージを複数のパケットに分割し、ＥＯＤトークンを用いて１つのパケット内に記述することができる。

【０１７５】

制御トークンの集合体を使って、データ通信およびアクセスプロトコルを提供することができる。この集合体は通常ソフトウェアにインタープリタされ、他のタイルのメモリへのアクセスに用いるＲＥＡＤおよびＷＲＩＴＥなどの動作を含むものである。アーキテクチャにより定義されるＲＥＡＤおよびＷＲＩＴＥトークンは、プロセッサ１４に専用のハードウェア論理があり、トークンによりトリガされるように配置されている。これは、読取りまたは書込み機能に必要となるものである。別法として、またはこれに追加して、読取りおよび書込み型動作を、アプリケーショントークンで実行してもよい。

【０１７６】

ＡＣＫおよびＮＡＣＫトークンは、その前に受信されたメッセージまたはパケットを受けて送信され、そのメッセージまたはパケットの処理が成功したかどうかを示すものである。別法として、またはこれに追加して、応答型動作をアプリケーショントークンで実行してもよい。

【０１７７】
一部の実施形態において、特権トークンは、システムの初期化、デバッグ、監視などに使用される。例として以下が挙げられる。
【０１７８】
ＷＲＩＴＥＩＤ　デバイス識別番号の書込み
ＲＥＡＤＩＤ　デバイス識別番号の読出し
ＲＥＡＤＴＹ　デバイス型の読出し

ＷＲＩＴＥＣ　構成の書込み
ＲＥＡＤＣ　構成の読出し

ＳＴＡＲＴ　デバイスを開始
ＳＴＯＰ　デバイスを停止

ＱＳＴＡＴＵＳ　デバイス状態の問合わせ
【０１７９】
ＷＲＩＴＥＩＤトークンおよびＲＥＡＤＩＤトークンは、スイッチ２１４、２１６の制御レジスタとの間で識別番号の書込みおよび読出しをするためのものである。ＲＥＡＤＴＹトークンは、その制御レジスタからスイッチ２１４、２１６の型を読出すためのものである。この型により、そのスイッチがシステムスイッチ２１６であるかプロセッサスイッチ２１４であるかがわかる。スイッチ２１４、２１６はそれぞれ、その識別番号および型からアレイ２００内で一意に識別される。
【０１８０】
ＷＲＩＴＥＣトークンおよびＲＥＡＤＣトークンは、スイッチ２１４、２１６の制御レジスタとの間で構造情報の書込みおよび読出しをするためのものである。この構造情報は、例えば、スイッチが使用するルーティング表や、タイルアドレスに関するものであり、この２つのトークンは、アレイ２００の初期設定などに使用可能である。
【０１８１】
ＳＴＡＲＴトークンおよびＳＴＯＰトークンは、スイッチ２１４、２１６のイネーブルおよびディセーブルに使用する。
【０１８２】
ＱＳＴＡＴＵＳトークンは、リンク２１８、２２０、２２２の制御レジスタにその状態判断のために問合わせをするものであり、例えば、リンクが使用中であるかどうか（もし使用中であればどの方向か）の判断に使用する。
【０１８３】
ハードウェアトークンは、通信リンク２２０、２２２の動作の制御に使用される。その例として以下が挙げられる。
【０１８４】
ＣＲＥＤＩＴ　データの送信を可能にする
ＬＲＥＳＥＴ　リンクをリセットする
【０１８５】
ＣＲＥＤＩＴ制御トークンは、受信側リンク２２０または２２２から生成されて送信側リンク２２０または２２２へと送信されて、受信側がトークンを受け入れ可能であることを示し、また、その受信側リンクで利用可能な、トークン分のスペース数を示す。
【０１８６】
リンク２２０および２２２のリスタートは、エラー後、リンクがＬＲＥＳＥＴトークンを生成し送信すると可能となる。この送信に対して、リンクは、ＬＲＥＳＥＴトークンを返送することで応じる。両方のリンクともがＬＲＥＳＥＴトークンを送信および受信した後でのみリセットされる。ＬＲＳＥＴを両方のリンクが同時に送信しようとしても問題ないことに留意されたい。

【０１８７】
上述した５ワイヤ方式を用いる特に有利な一実施形態において、ＣＲＥＤＩＴトークンは、エスケープ、値、別のエスケープ、再度同じ値として、例えばv_1をv_3と同じ値とする$e_0 v_1 e_2 v_3$として符号化される。こうして、ＣＲＥＤＩＴトークンは、どのワイヤの状態も変更することのないものとして保証される。というのは、２つのエスケープ記号は常に同一ワイヤ上の２つの反対方向の遷移となり、同様に２つの値記号も常に同一ワイヤ上の２つの反対方向の遷移となって、リンクを、ＣＲＥＤＩＴトークンの送信前と同じ状態に保つからである。したがって、もしリンクが他の値を有していると、その値はＣＲＥＤＩＴトークンによる影響を受けないことになる。このＣＲＥＤＩＴトークンは、こうした値を有効に「パス」させられる。これは、リンク上のソフトウェアトークンや他の値との干渉を懸念せずとも、インターコネクト内ハードウェアによりＣＲＥＤＩＴトークンをパスさせられるため、有用である。インターコネクトハードウェアによるＣＲＥＤＩＴトークンの送信は、完全な自律操作とすることができる。

【０１８８】
リンク状態への影響がまったくないトークンを構成できるのであれば、同様の概念を、３ワイヤ方式および他の方式におけるＣＲＥＤＩＴトークンに適用することができる。

【０１８９】
ソフトウェアメッセージの構成例を、図１０を見ながら説明する。図１０は、１つのプロセッサ１４（送信元プロセッサ）により、別のプロセッサ１４（宛先プロセッサ）のメモリ２４を読み取るために出力された読み取りメッセージ１０１を例示している。

【０１９０】
動作時、送信元プロセッサ１４で実行されているコードが、まず宛先プロセッサに出力するヘッダ１０２を生成する。このヘッダ１０２は、上述したようにチャネルを形成するため、宛先プロセッサのアドレスを指定するものである。この例において、ヘッダは、２つのデータトークンを含む。続いて、ヘッダ１０２の生成後、そのソフトウェアは、宛先プロセッサのメモリ読み取りを要求するため、ＲＥＡＤ制御トークン１０４を生成する。このＲＥＡＤ制御トークンは、送信元プロセッサが実行したＯＵＴＣＴ命令のオペランドから生成される。この制御トークンは宛先ソフトウェアがどの機能を実行せねばならないかを通知するものである。ＲＥＡＤトークンの生成後、送信元ソフトウェアは、アドレス部分１０６とリターンヘッダ部分１０８とを生成する。この例ではアドレス部分１０６が４トークン長、リターンヘッダ部分１０８が２トークン長であり、これもそれぞれＯＵＴＴ命令により出力される。これらの部分１０６および１０８により、要求の実行に必要な情報、すなわち、ロードすべきワードのアドレスと、そのデータを返信すべきプロセッサのアドレスとが得られる。読み取りアドレス１０６およびリターンヘッダ１０８の生成後、そのソフトウェアにより、チャネルを閉鎖するＥＯＭ制御トークン１１０が生成される。

【０１９１】
上記メッセージのフォーマットには、アドレスをワード単位で位置調整しているか、ローカルメモリにオフセットを有するかなど、何ら制限はないことに留意されたい。フォーマットは、そのメッセージを扱うソフトウェアに依存する。

【０１９２】
図１１に例示するように、うまく読み取られる応答メッセージ１１１は、メッセージ読取り要求１０１に供給されたリターンヘッダ１０８を先頭としている。次に、肯定応答制御トークンＡＣＫ１１２が続いて、読み取りが成功したことを送信元プロセッサに示す。ＡＣＫ制御トークン１１２の生成後、宛先ソフトウェアは、リターンデータ部分１１４を生成する。このデータ部分１１４は、アドレス部分１０６に指定された宛先プロセッサのメモリのアドレスから出力されるものである。ＡＣＫ１１２の生成後、同じソフトウェアでＥＯＭ制御トークン１１６が生成され、このチャネルが閉じられる。

【０１９３】
図１２に例示するように、読み取りが不成功に終わる返信メッセージ１２１もメッセー

ジ読取り要求１０１に供給されたリターンヘッダ１０８が先頭となる。次に、否定応答制御トークンＮＡＣＫ１１８が続いて、アドレス部分１０６に指定されたアドレスが存在しなかったなどの理由で読み取りが不成功だったこと、すなわちエラーがあることを送信元プロセッサに示す。このようなエラーが発生した場合、データの返送は必要なく、ＮＡＣＫ１１８の次に、ＥＯＭ制御トークン１１６が生成され、このチャネルが閉じられる。

【０１９４】

上述したように、チャネルの使用法には３つあり、ストリーム用、パケット用、同期用である（同期されたチャネルはパケット用チャネルの１種）。以下、この通信の効率化のためにトークンおよび命令セットをさらに改良することについて説明する。

【０１９５】

第１の改良点は、スイッチ２１４および２１６のルートを閉鎖するが、受信側プロセッサ１４には見えない（少なくとも、受信側プロセッサで実行される入力命令には無視される）ＰＡＵＳＥ制御トークンを提供することである。このＰＡＵＳＥトークンは、ハードウェアトークンの性質も専用トークンの性質も持つ特殊なものであり、ハードウェアトークンのように、しかし専用トークンとは異なり、受信側プロセッサのソフトウェアにアクセスできない。しかし、ハードウェアとは異なり、そして専用トークンのように、送信側プロセッサのソフトウェアにより生成可能なものである。つまり、ストリームは一時停止可能であり、そのインターコネクトルートは、受信側の専用コードがなくても一時的に解放される。継続するには、送信側がトークンの送信を開始すればよい。ＰＡＵＳＥは、宛先チャネル端が同一プロセッサ上にある場合には無効である。ＰＡＵＳＥをＥＯＰトークンの代わりに使用することもできる。

【０１９６】

第２の改良点は、ＥＯＭトークンの送信およびチェックを迅速にすることである。これは、１つのアドレスＯＵＴＥＯＭおよびＣＨＫＥＯＭ命令を使用することで実現できる。ＯＵＴＥＯＭはＥＯＭトークンを出力する。ＣＨＫＥＯＭは、次に受信されるトークンがＥＯＭトークンでなければトラップする。ＩＮＴおよびＩＮ命令は、ＥＯＭトークンで使用されるとトラップする。トラップは割り込みと同様の効果があるが、トラップは特定のエラー条件で自動的に生成されるものであり、特定トラップベクトルを転送する。トラップの原理は、当業者には周知であろう。

【０１９７】

第３の改良点は、ＯＵＴＰＡＵＳＥ命令を提供して、ＰＡＵＳＥトークンの別途コード化を不要とする。

【０１９８】

チャネルのセッティングおよび制御用のコードシーケンスの例を以下に挙げる。同一プロセッサ上２つのスレッド間におけるチャネルのセッティングは以下のようになる。

【０１９９】

```
GETR CHAN c1
GETR CHAN c2
SETD c1, c2
SETD c2, c1
```

【０２００】

チャネル端識別子ｃ１またはｃ２を、初期化した後、別のスレッドにパスさせることができる。

【０２０１】

以下を実行して遠隔プロセッサ、すなわち２つの異なるプロセッサ間のスレッド間をブートして、遠隔プロセッサを形成することができる。

```
GETR CHAN,  c1
```

次に、以下を含むブートストラッププログラムを送信し、

```
…
GETR CHAN, c2
```

```
SETD c2, c1
OUT c2, c1        //チャネル端識別子を出力
…
```

最後に、以下を実行する。

```
IN c1, c2          //チャネル端識別子を入力
SETD c1, c2
```

【０２０２】

上記どちらの例においても、チャネルの一端の識別子を使うだけで通信ができるようになる。

【０２０３】

ここで、３種類のチャネル、すなわちストリーム用、パケット用、同期用チャネルをセッティングおよび制御するコード例について説明する。

【０２０４】

ストリーム用チャネルｃは、出力および入力を用いるだけで操作することができる。「ｐａｕｓｅ（ｃ）」（チャネルの一時停止）命令も、ＰＡＵＳＥトークンの生成に利用可能であり、この一時停止は、転送を中断するためにいつでも実行することできるが、入力側スレッドにこれは見えない。高水準において、ストリーム用チャネルでトークンを受信するコードは以下のようになる。

【０２０５】

```
switch c=>> s
        case ct1…              //制御トークン１が…の場合
        case ct2…              //制御トークン２が…の場合
        …
        default control…
        case dt1               //データトークン１が…の場合
        case dt2               //データトークン２が…の場合
        default data…
```

【０２０６】

これをコンパイルすると以下のようになる。

```
TESTCT      c,        flag
IN            c,        s
BFF          flag, data        //データトークンの場合「data」に分岐
[code for control tokens…]
BFU end                      //「end」に分岐
data:
[code for data tokens…]
end:
```

【０２０７】

パケット用チャネルｃ上の単方向通信に対して、送信側プロセッサＰの高水準コードは、

```
out (c) {…}                  // cへの出力を含む命令シーケンス
```

であり、これをコンパイルすると以下となる。

```
OUTT c, token1
OUTT c, token2
…
OUTEOM c
```

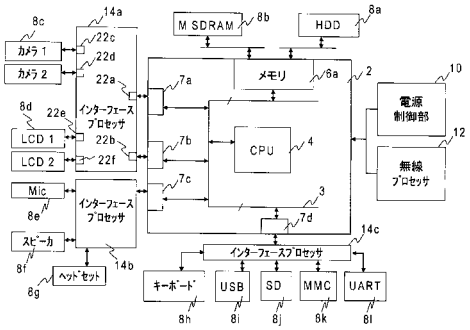
（好適な実施形態では、上述したように、未接続のチャネル端４２に出力される際にＣＥＩＤレジスタ４１から自動的にヘッダが送信されるため、ＯＵＴＴでヘッダを出力する必要はないことに留意されたい。）

【０２０８】

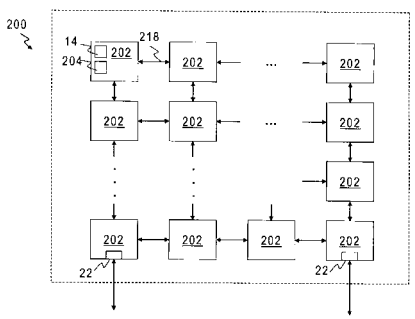
受信側プロセッサＱの高水準コードは、
in（c）{…}　　// cへの入力を含む命令シーケンス
であり、これをコンパイルすると以下となる。
INT c，token 1
INT c，token 2
…
CHKEOM c
【０２０９】
ただし、Ｐがトークンを送信しすぎると、ＱのＣＨＫＥＯＭがトラップし、Ｐが送信するトークンが少なすぎると、Ｑの入力の１つがトラップする。したがって、ＣＨＫＥＯＭ命令によって、プロセッサ間の通信のパケット構造を強化する。
【０２１０】
同期用チャネルｃ（パケット用チャネルの１種）上の単方向通信に対して、送信側プロセッサＰのコードは、
out（c）{…}　　// cへの出力を含む命令シーケンス
であり、これをコンパイルすると以下となる。
OUTT c，token 1
OUTT c，token 2
…
OUTEOM c
CHKEOM c
【０２１１】
受信側プロセッサＱの高水準コードは、
in（c）{…}　　// cへの入力を含む命令シーケンス
であり、これをコンパイルすると以下となる。
INT c，token 1
INT c，token 2
…
CHKEOM c
OUTEOM c
【０２１２】
この場合も、ただし、Ｐがトークンを送信しすぎると、ＱのＣＨＫＥＯＭがトラップし、Ｐが送信するトークンが少なすぎると、Ｑの入力の１つがトラップする。また、ＰがそのＥＯＭを送信するまでＱは処理できず、ＱがそのＥＯＭを送信するまでＰは処理できない。このためＰおよびＱは同期化される。つまり、ＱのＣＨＫＥＯＭは、ＱがＰからパケット全体を受信するまで応答できないことを保証し、ＰのＣＨＫＥＯＭは、ＰがＱからリターンＥＯＭを含む応答パケット全体を受信するまで、続きの通信を継続できないことを保証するものである。
【０２１３】
パケット用通信、同期用通信の双方において、ＰおよびＱが双方向に通信すると、上記命令が使用されて、適切な数のトークンを各方向に送信することができる。
【０２１４】
上記実施形態は実施例としてのみ記載したものであることを理解されたい。この他の実施形態では、別のレジスタおよび命令セットも、チップの仕様に応じて使用可能である。イベントバッファ３８’も、入力バッファ４４の代替として、またはこれに追加するものとして、チャネル端４２の出力バッファ４６用に使用可能である。スレッドのスケジュールは、ポートおよびチャネルの他に、他の発信元からのアクティビティを基準にしてもよい。チャネル端は上記では入力バッファおよび出力バッファを有するが、単方向チャネル端も使用可能である。プロセッサのさまざまな部品間に、さまざまな接続を設けることも、かつ／または、プロセッサ間および／またはタイル２０２間にインターコネクト２０４

をさまざまに配置することも可能である。データトークンおよび／または制御トークンの生成かつ／または構成もさまざまな順序で可能である。ヘッダ、メッセージ、アドレス、および／またはトークンにさまざまな長さを持たせることも、これらをさまざまな品質のデータに用いることも可能である。さらに、本発明は、端末用アプリケーションプロセッサを備えたモバイル端末での仕様に特定されるものではない。他の用途および構成も、当業者には明白であろう。本発明の範囲は上記実施形態に限定されるものではなく、以下の請求の範囲によってのみ限定されるものである。

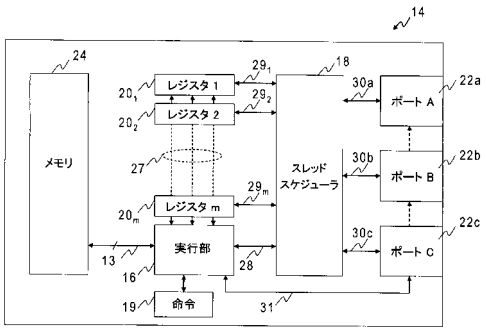
【図１】



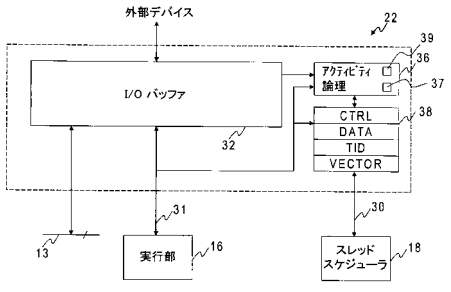
【図２】



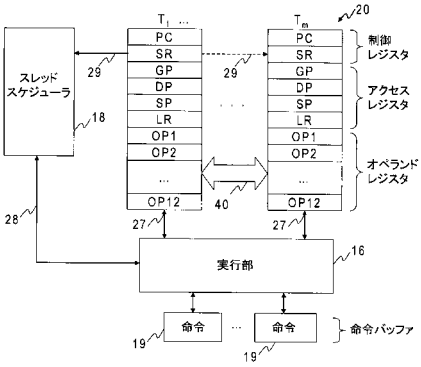
【図３】



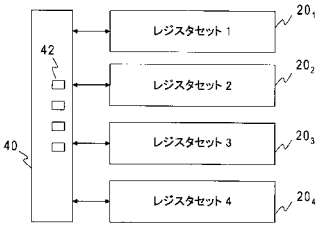
【図４】



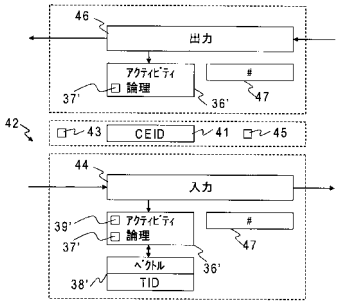
【図５】



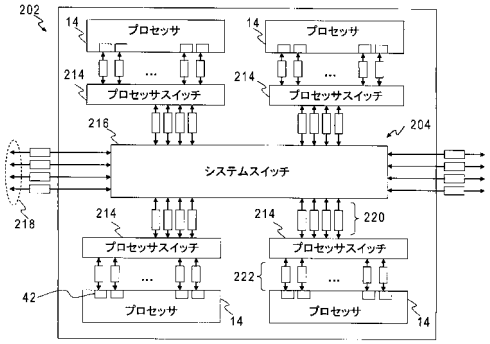
【図６】



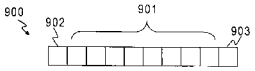
【図７】



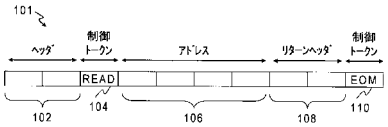
【図８】



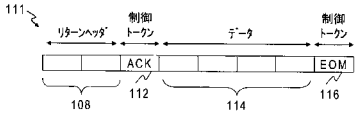
【図９】



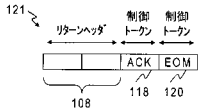
【図１０】



【図１１】



【図１２】



フロントページの続き

(56)参考文献 特開平０４－２３２５５３（ＪＰ，Ａ）
特開２００４－３４７９７５（ＪＰ，Ａ）
国際公開第０１／０３００１４（WO，Ａ１）
特開２００５－３５４４３１（ＪＰ，Ａ）
Ethiopia Nigussie, Sampo Tuuna, Juha Plosila, Jouni Isoaho, Analysis of Crosstalk and Process Variations Effects on On-Chip Interconnects, Proceedings of International Symposium on System-on-Chip 2006, IEEE, ２００６年１１月１３日, Pages:1-4

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ１５／１６－１５／１７７
Ｈ０４Ｌ１２／００－１２／２６
Ｈ０４Ｌ１２／５０－１２／９５１
Ｈ０４Ｌ２５／００－２５／６６