(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3599197号
(P3599197)
(45)発行日　平成16年12月8日(2004.12.8)　(24)登録日　平成16年9月24日(2004.9.24)

(51)Int.Cl.[7]　F I
G06F 15/173　G06F 15/173

請求項の数 20　(全 27 頁)

(21)出願番号　特願平6-524304
(86)(22)出願日　平成6年4月12日(1994.4.12)
(65)公表番号　特表平8-511886
(43)公表日　平成8年12月10日(1996.12.10)
(86)国際出願番号　PCT/US1994/003994
(87)国際公開番号　WO1994/025920
(87)国際公開日　平成6年11月10日(1994.11.10)
審査請求日　平成13年4月12日(2001.4.12)
(31)優先権主張番号　08/055,814
(32)優先日　平成5年4月30日(1993.4.30)
(33)優先権主張国　米国(US)

(73)特許権者
クレイ・リサーチ・インコーポレーテッド
アメリカ合衆国　５５１２１　ミネソタ州　イーガン、ローン・オーク・ドライブ　６５５エイ番
(74)代理人
弁理士　青山　葆
(74)代理人
弁理士　河宮　治
(72)発明者　スティーブンソン、ブリッキー・エイ
アメリカ合衆国　５４７２９　ウィスコンシン州　チッペワ・フォールズ、ウェスト・レイク・ドライブ　６４０９番

最終頁に続く

(54)【発明の名称】待ち時間が可変の、プロセッサをメモリに接続する相互接続ネットワーク

(57)【特許請求の範囲】
【請求項１】
マルチプロセッサ・システムにおける複数の中央処理装置（CPU）（100）と複数のメモリ・モジュール（600）との間でパケットの経路を設定するための相互接続ネットワーク（300）であって、
上記相互接続ネットワーク（300）は、
（a1）前記複数のCPU（100）を前記複数のメモリ・モジュール（600）に接続するための出力経路ネットワーク（302）と、
（a2）前記複数のメモリ・モジュール（600）を前記複数のCPU（100）に接続するための入力経路ネットワーク（304）とを備え、
前記出力経路ネットワークは、
（b1）第１のクロスバー・スイッチ（320）と、
（b2）前記複数のCPU（100）への接続のための入力バッファ（310）と、
（b3）前記複数のメモリ・モジュール（600）への接続のための第２のクロスバー・スイッチ（330）とを備え、
前記入力バッファ（310）は前記第１のクロスバー・スイッチ（320）に接続され、パケットを前記CPU（100）から前記第１のクロスバー・スイッチ（320）へ転送し、
前記第２のクロスバー・スイッチ（330）は前記第１のクロスバー・スイッチ（320）に接続され、パケットを前記第１のクロスバー・スイッチ（320）から前記複数のメモリ・モジュール（600）へ転送し、

前記第１及び第２のクロスバー・スイッチ（320,330）の少なくとも一方は前記複数のCPU（100）から前記スイッチを介して転送されるパケットをキューイングするための先入れ先出し（FIFO）キュー構造体（321,331）を含み、
前記入力経路ネットワークは、
（c1）第３のクロスバー・スイッチ（360）と、
（c2）前記複数のメモリ・モジュール（600）への接続のための入力バッファ（350）と、
（c3）前記複数のCPU（100）への接続のための第４のクロスバー・スイッチ（370）とを備え、
前記入力バッファ（350）は前記第３のクロスバー・スイッチ（360）に接続され、パケットを前記メモリ・モジュール（600）から前記第３のクロスバー・スイッチ（360）へ転送し、
前記第４のクロスバー・スイッチは前記第３のクロスバー・スイッチ（360）に接続され、パケットを前記第３のクロスバー・スイッチ（360）から前記複数のCPU（100）へ転送し、
前記第３及び第４のクロスバー・スイッチ（360,370）の少なくとも一方は前記複数のメモリ・モジュール（600）から前記スイッチを介して転送されるパケットをキューイングするためのFIFOキュー構造体を含むことを特徴とする相互接続ネットワーク。
【請求項２】
前記出力経路ネットワーク（302）の前記入力バッファ（310）は、前記第１のクロスバー・スイッチ（320）へ転送されるパケットをキューイングするためのFIFOキュー構造体（331）を備えたことを特徴とする請求項１記載の相互接続ネットワーク。
【請求項３】
前記第１及び第２のクロスバー・スイッチ（320,330）の一方は、パケットの送信中にパケットを一時的に格納するための出力要求バッファをさらに含むことを特徴とする請求項１又は２記載の相互接続ネットワーク。
【請求項４】
前記第１及び第２のクロスバー・スイッチ（320,330）は、前記パケットの複数のメッセージ・ステアリング・ビットを、前記クロスバー・スイッチへの入力を表すビットで置換するための手段をさらに含むことを特徴とする請求項１、２又は３記載の相互接続ネットワーク。
【請求項５】
前記入力経路ネットワーク（304）の前記入力バッファ（350）は、前記第３のクロスバー・スイッチ（360）へ転送されるパケットをキューイングするためのFIFOキュー構造体（351）を備えたことを特徴とする請求項１、２、３又は４記載の相互接続ネットワーク。
【請求項６】
前記第３及び第４のクロスバー・スイッチ（360,370）の一方は、パケットの送信中にパケットを一時的に格納するための出力要求バッファをさらに含むことを特徴とする請求項１乃至５のうちのいずれか１つに記載の相互接続ネットワーク。
【請求項７】
請求項１記載の相互接続ネットワーク（300）によって、複数のメモリ・モジュールに接続された複数のCPU（100）を備えたマルチプロセッサ・システムにおいて使用するためのメモリ・モジュール（600）であって、
複数の入力と、
複数の出力と、
複数のメモリ・バンク（640a－640p）と、
前記複数の入力を前記複数のメモリ・バンクに接続する第１のルーティング・ネットワークと、
前記複数のメモリ・バンクを前記複数の出力に接続する第２のルーティング・ネットワークとを備え、
前記第１のルーティング・ネットワークは、前記メモリ・バンクの１つへ転送されるパケ

ットをキューイングするためのFIF0キュー構造体（331）を有する第１のスイッチを含む複数のスイッチ（610）を備え、
前記第２のルーティング・ネットワークは、前記メモリ・バンクの１つから転送されるパケットをキューイングするためのFIF0キュー構造体（331）を有する第２のスイッチを含む複数のスイッチ（610）を備えたことを特徴とするメモリ・モジュール。
【請求項８】
前記第１のルーティング・ネットワークにおける前記FIF0キュー構造体（331）はＮ個の入力を備え、
前記数字Ｎは指定されたスイッチ間の通信の往復時間をカバーするように選択されたことを特徴とする請求項７記載のメモリ・モジュール（600）。
【請求項９】
前記各メモリ・バンクは複数のメモリ・デバイスを備え、
前記各メモリ・デバイスはアクセス時間とサイクル時間とを有し、
前記各メモリ・バンクは、複数のメモリ・デバイスの前記アクセス時間とサイクル時間とに依存して前記メモリ・バンクの速度を調整するための遅延手段をさらに備えたことを特徴とする請求項７又は８記載のメモリ・モジュール（600）。
【請求項１０】
前記第２のルーティング・ネットワークにおける前記FIF0キュー構造体（331）はパイプライン方式のメモリ部分を扱うために十分な深さを有することを特徴とする請求項７、８又は９記載のメモリ・モジュール（600）。
【請求項１１】
クロスバー・スイッチ（320）を備えたネットワーク・モジュールとの組み合わせ装置において、
前記クロスバー・スイッチ（320）は、前記第１のルーティング・ネットワークの前記第１のスイッチに接続され、前記第１のスイッチから受信されたパケットをバッファリングするためのFIF0キュー構造体（331）を含むことを特徴とする請求項７乃至10のうちのいずれか１つに記載のメモリ・モジュール（600）。
【請求項１２】
前記クロスバー・スイッチ（320）は、パケットの送信中にパケットを一時的に格納するために使用される出力要求バッファをさらに含むことを特徴とする請求項11記載の組み合わせ装置。
【請求項１３】
前記パケットはメッセージ・ステアリング・ビットを備え、
前記クロスバー・スイッチは、複数のメッセージ・ステアリング・ビットを、前記複数の入力の１つを表すビットで置換するための手段をさらに含むことを特徴とする請求項11又は12記載の組み合わせ装置。
【請求項１４】
クロスバー・スイッチ（320）を備えたネットワーク・モジュールとの組み合わせ装置において、
前記クロスバー・スイッチ（320）は、前記第２のルーティング・ネットワークの前記第２のスイッチに接続され、前記第２のスイッチから受信されたパケットをバッファリングするためのFIF0キュー構造体（331）を含むことを特徴とする請求項７乃至10のうちのいずれか１つに記載のメモリ・モジュール（600）。
【請求項１５】
前記クロスバー・スイッチ（320）は、パケットの送信中にパケットを一時的に格納するための出力要求バッファをさらに含むことを特徴とする請求項14記載の組み合わせ装置。
【請求項１６】
前記クロスバー・スイッチ（320）は、前記パケットの複数のメッセージ・ステアリング・ビットを、前記複数のメモリ・バンクの１つを表すビットで置換するための手段をさらに含むことを特徴とする請求項14又は15記載の組み合わせ装置。

【請求項１７】
第１のクロスバー・スイッチと、第２のクロスバー・スイッチとを備えたネットワーク・モジュールとの組み合わせ装置において、
上記第１のクロスバー・スイッチは、前記第１のルーティング・ネットワークの前記第１のスイッチに接続され、前記第１のスイッチから受信されたパケットをバッファリングするためのFIFOキュー構造体を含み、
上記第２のクロスバー・スイッチは、前記第２のルーティング・ネットワークの前記第２のスイッチに接続され、前記第２のスイッチから受信されたパケットをバッファリングするためのFIFOキュー構造体を含むことを特徴とする請求項７乃至10のうちのいずれか１つに記載のメモリ・モジュール。
【請求項１８】
請求項１乃至６のうちのいずれか１つに記載の相互接続ネットワーク（300）と、
前記相互接続ネットワークに接続された複数のCPU（100）と、
前記相互接続ネットワークに接続された複数のメモリ・モジュール（600）とを備えたことを特徴とするマルチプロセッサ計算システム。
【請求項１９】
出力経路ネットワーク（302）と入力経路ネットワーク（304）とを有する相互接続ネットワークによって複数のCPU（100）に接続された複数のメモリ・モジュールのうちの１つのメモリ・モジュール（600）からの読み出し方法であって、
上記出力経路ネットワーク（302）は前記メモリ・モジュール（600）に接続され、入力と、出力と、前記入力を前記出力に接続する複数の第１のステージとを備え、前記各第１のステージはスイッチとバッファとを備え、
上記入力経路ネットワーク（304）は前記メモリ・モジュール（600）に接続され、入力と、出力と、前記入力を前記出力に接続する複数の第２のステージとを備え、前記各第２のステージはスイッチとバッファとを備え、
上記方法は、
メッセージ・ステアリング・ビットを有する読み出しコマンド・パケットを前記出力経路ネットワークの前記入力へ転送するステップと、
前記読み出しコマンド・パケットを前記出力経路ネットワークの第１のステージにおいてバッファに格納するステップと、
前記出力経路ネットワークの次のステージへの経路が空いているとき、前記読み出しコマンド・パケットを前記出力経路ネットワークの前記次のステージへ転送し、前記メッセージ・ステアリング・ビットを修正して前記読み出しコマンド・パケットがどこから来たかを示すステップと、
前記出力経路ネットワークのすべてのステージが通過されたとき、前記読み出しコマンド・パケットを前記メモリ・モジュールへ転送し、前記メッセージ・ステアリング・ビットを修正して前記読み出しコマンド・パケットがどこから来たかを示すステップと、
前記読み出しコマンド・パケットを前記メモリ・モジュールにおけるバッファに格納するステップと、
前記読み出しコマンド・パケットによって指示されたメモリ・ロケーションから読み出しを行ってデータを獲得するステップと、
メッセージ・ステアリング・ビットを含むデータをデータ・パケットにして転送するステップと、
前記データ・パケットを前記入力経路ネットワークの第１のステージにおいてバッファに格納するステップと、
前記入力経路ネットワークの次のステージへの経路が空いているとき、前記データ・パケットを前記入力経路ネットワークの前記次のステージへ転送し、前記メッセージ・ステアリング・ビットを修正して前記データ・パケットがどこから来たかを示すステップとを含むことを特徴とする読み出し方法。
【請求項２０】

出力経路ネットワーク（302）と入力経路ネットワーク（304）とを有する相互接続ネットワークによって複数のCPU（100）に接続された複数のメモリ・モジュールのうちの１つのメモリ・モジュール（600）への書き込み方法であって、
上記出力経路ネットワーク（302）は前記メモリ・モジュール（600）に接続され、入力と、出力と、前記入力を前記出力に接続する複数の第１のステージとを備え、前記各第１のステージはスイッチとバッファとを備え、
上記入力経路ネットワーク（304）は前記メモリ・モジュール（600）に接続され、入力と、出力と、前記入力を前記出力に接続する複数の第２のステージとを備え、前記各第２のステージはスイッチとバッファとを備え、
上記方法は、
書き込みコマンド・パケットを前記出力経路ネットワークの前記入力へ転送するステップを含み、前記書き込みコマンド・パケットはアドレスとメッセージ・ステアリング・ビットと書き込みデータとを含み、
上記方法は、
前記書き込みコマンド・パケットを前記出力経路ネットワークの第１のステージにおいてバッファに格納するステップと、
前記出力経路ネットワークの次のステージへの経路が空いているとき、前記書き込みコマンド・パケットを前記出力経路ネットワークの前記次のステージへ転送し、前記メッセージ・ステアリング・ビットを修正して前記書き込みコマンド・パケットがどこから来たかを示すステップと、
前記出力経路ネットワークのすべてのステージが通過されたとき、前記書き込みコマンド・パケットを前記メモリ・モジュールへ転送し、前記メッセージ・ステアリング・ビットを修正して前記書き込みコマンド・パケットがどこから来たかを示すステップと、
前記書き込みコマンド・パケットを前記メモリ・モジュールにおけるバッファに格納するステップと、
前記書き込みデータを前記アドレスによって指示されたロケーションに格納するステップとを含むことを特徴とする書き込み方法。

【発明の詳細な説明】

＜発明の背景＞

多くのデータ処理のタスクは、データの整然とした配列に対する大規模な算術操作を必要とする。一般に、この種の操作すなわち「ベクトル」処理は、データ集合の連続する各要素に対して同一操作を繰り返し実行することが必要となる。整然とした配列データを扱うときに処理速度およびハードウェア効率を向上させるために、ベクトル計算機が開発されている。ベクトル計算機は、そのハードウェア機構により整然とした配列データを扱うものであり、したがって、スカラ計算機よりも高い演算速度（speed of operation）が得られる。

スカラ計算機およびベクトル計算機の双方における処理速度（computer processing speed）および処理効率は、マルチ処理技術（multiprocessing techniques）を利用して更に向上させることができる。マルチ処理は、主記憶のようなシステム資源を共有する数百ないし数千のプロセッサを使用する必要がある。異なるジョブの独立したタスクまたは単一ジョブの関連するタスクは、複数のプロセッサ上で実行される。各プロセッサはそれ自身の命令セット従い、それぞれの命令を並列に実行する。プロセッサの数を増やしてそれらを並列に動作させることにより、より多くの作業をより短い時間で行うことができる。

マルチ処理は作業速度（performance speed）を向上させることができるが、その向上は使用されるプロセッサの数と線形の関係にはならない。これは、主として二つの要因すなわちオーバヘッドとロックアウト（lock out）によるものである。プロセッサとプロセッサの機能との間で調整をとるのに要する同期と制御のレベルが増大するため、マルチプロセッサの環境の下では著しいオーバヘッドが生じる。全プロセッサ間での通信および全プロセッサの制御は、マルチ処理システムに性能低下を生じさせる。数個のプロセッサが協調して一つのタスクを実行しているときには、データの依存性とプロセッサ間でのデータ

の転送は不可避である。或るプロセッサが他のプロセッサから転送されるべきデータを待たなければならないときに、プロセッサのアイドル時間が生じる。このプロセッサのアイドル時間はシステム性能を低下させることになる。

マルチプロセッサ・システムの性能低下の他の重大な原因は、プロセッサのロックアウトすなわちブロッキング（blocking）であり、複数のプロセッサが共通の資源を共有することに関連している。これは、一つのプロセッサが、他のプロセッサが既に使用している共有資源、例えば共有メモリにアクセスをしようとするときに生じる。そのプロセッサは、このときその共有資源の使用を阻止され（ブロックされ）、他のプロセッサが停止するまで待機しなければならない。この場合もまた、プロセッサのアイドル時間が発生し、システムの性能が低下する。

プロセッサのメモリに対するインタフェイスは、オーバヘッドとロックアウトの概念と緊密につながっており、それは計算機の全体的な性能にも影響を与える。マルチプロセッサのインタフェイスの一例を、BBNシステム・アンド・テクノロジー社（BBN Systems and Technologies Corporation）によって設計されたモナーク（Monarch）並列マルチ処理計算機に見ることができる。モナークはスカラ方式で単一スレッドのマルチ処理アーキテクチャとなっており、プロセッサとメモリとの間の通信に回線交換技術を使用している。この回線交換技術によれば、全てのプロセッサはメモリへ至る同一の経路を共有する。モナークの設計における或るプロセッサがメモリを要求するとき、そのプロセッサ網からメモリへの経路全体が開放され、そのメモリとそのプロセッサとの間の通信が完了するまで開放されたままとなる。この方式は、その回線交換網を経由してメモリを参照しようとしている他のプロセッサをブロックすることができ、メモリ参照の転送速度を制限し、その結果、プロセッサのアイドル時間が長くなる。したがって、このような設計をマルチプロセッサによるマルチスレッドのベクトル処理において採用するのは実用的ではなく、このようなベクトル処理ではプロセッサとメモリとの間で本来的に大量のデータを転送しなければならない。

マルチプロセッサのメモリインタフェイスの他の例を、ホライゾン（HORIZON）ルーティング方式に見ることができる。このホライゾンのインタフェイス・ネットワークは、自暴自棄ルーティング（desperation routing）またはホットポテト・ルーティング（hot potato routing）と呼ばれる方式を使用している。ホライゾンの自暴自棄ルーティングは、複数の入力と同数の出力を有する多段ネットワークである。この方式は、どのネットワークのサイクルにおいても各入力に対し一つの出力への経路を設定する必要がある。例えば、４個の入力情報（input references）が存在し、その４個の入力情報が同一の出力に行くことを欲している場合、その４個の入力情報のうち一つが正しい出力へ行き、他の全ての入力は欲していない他のいずれかの出力へ行く。これは、４個の入力のうち３個は格段に長い経路をとってそのネットワークを通過することを意味する。このホライゾンの自暴自棄ネットワークは、これらの他の三つの情報（references）は結局は望ましい入力へ戻って、望ましい出力に到達する別の機会を持つように、ネットワークに経路が定められる。したがって、それらの情報がネットワークの中で永久に失われるということはなく、ホライゾンのルーティング方式は、そのネットワーク内に最も長く存在している情報が最高の優先度を持つような機構を有しており、これにより、それらの情報はいつかは同一の出力に対する競合する情報に打ち勝つことになる。このようなルーティング方式によれば、一つの情報が要求する終点への可能な経路を複数持ち、多くの情報は、その目的地へ到達する前にネットワーク内を通行するのに非常に長い期間を費やすことになりかねない、ということが当業者には容易にわかるであろう。したがって、このホライゾンの自暴自棄ルーティング方式をマルチ処理の計算機で使用するのは望ましくない。

マルチプロセッサのメモリインタフェイスの更に他の例が、ムラタ（Murata）らによって1992年４月１日に出願されたヨーロッパ特許出願第0 510 821号において説明されている。この出願は、共有メモリ・モジュールと複数のプロセッサとの間でデータを転送するためにアドレスバスに接続されたスイッチを使用することを教示している。このアドレスバスは、特定のプロセッサのキャッシュ内に存在するデータのアドレスに対するメモリ参照

について調べる（snoop）のために使用することができ、一方、このスイッチは、メモリ・モジュールとプロセッサとの間での高速のデータ転送路を提供する。
マルチプロセッサのメモリインタフェイスの他の例が、ハイデルバーガ（Heidelberger）らによって1990年３月２日に出願されたヨーロッパ特許出願第0 392 184号において説明されている。この出願は、メモリとプロセッサ双方のキャッシュの使用について述べている。このメモリ・キャッシュは、共有メモリに対するアクセス時間を短縮するために共有メモリデータをキャッシュするのに使用される。
マルチ処理システムにおける他の重要な概念は、異なる使用者の要求に合わせて種々の異なるサイズにすることである。例えば、最大システムは1024個のプロセッサを持つことができるが、512個のプロセッサを有する構成、256個のプロセッサを有する構成、または他の或る構成というように縮小化されたシステムを利用できるようにするのが望ましい。最大システムを構成する基本的なビルディングブロック（building blocks）を修正することなく使用して最小システムを作り上げることができることが重要であり、逆もまた同様である。したがってスケーラブルなシステムは、格段に柔軟性に富み、このようなシステムの展開によって使用者の変化する要求に合わせることができる。
以上より、プロセッサをメモリに接続する相互接続ネットワークであって、例えばプロセッサが競合することなくメモリ参照を発行できるようにした相互接続ネットワークに対する要求が当業界にあり、それは、ネットワーク内でのメモリ参照の間での競合を低減し、どの１回の参照についてもネットワーク内で費やす時間を低減し、その結果、プロセッサのアイドル時間が減少し、システムの性能が向上する。また、モジュラーな相互接続ネットワークであって、その相互接続ネットワークを構成する個々のモジュールの再設計を必要とせずに、任意の個数のプロセッサおよび異なるサイズのメモリを有するマルチ処理システムに適合するように容易に規模を変更することができる相互接続ネットワークに対する要求もある。
＜発明の要約＞
上記技術における限界を克服するために、および本明細書を読んで理解すれば明らかになるであろう他の限界を克服するために、本発明は、プロセッサをメモリに接続するマルチ処理用の相互接続ネットワークを提供する。この相互接続ネットワークは、種々の個数のプロセッサおよび種々のメモリサイズを有するように構成されたシステムでの使用に適合したものである。すなわち、この相互接続ネットワークのモジュール性により、単純な複製で任意の規模のシステムを構築できるようになる。
この相互接続ネットワークは、ネットワーク・モジュールおよびメモリ・モジュールを含む。ネットワーク・モジュールおよびメモリ・モジュールは、一連のｎ×ｍスイッチで構成されていて、各ｎ×ｍスイッチはｎ個の入力のうちの一つからｍ個の出力のうちの一つへの経路を設定する。このスイッチは、相互接続ネットワークにおけるメッセージの競合が低減されるように設計されている。このスイッチ、したがってメモリとネットワークのモジュールは高いモジュール性を有しており、これにより、同一の構成要素を利用して任意の規模のマルチ処理システムを実質的に構築できるようになる。
また、メッセージの競合を低減するための機構および相互接続ネットワークを実現するために必要な相互接続の数が与えられる。最後に、相互接続ネットワーク内の特定の経路上のデータが正しいか否かを判定するために健全性コード（sanity code）が使用される。この健全性コードは、メモリ・バンクもしくはネットワーク経路の故障を検出し、または、システムのグレードを下げるためにも使用される。
本発明は、請求項１（ネットワーク）、請求項８（読み出し方法）、請求項９（書き込み方法）、請求項10（計算機システム）、および請求項17（メモリ・モジュール）において定義されている。
【図面の簡単な説明】
図面において、同一符号はいくつかの図を通して同一構成要素を示している。
図１は、４個のCPUを有するマルチ処理システムの一例のブロック図を示す。
図２は、図１に示した４個のCPUを有するマルチ処理システムのより詳細なブロック図で

ある。
図３は、32個のCPUを有するマルチ処理システムの一例のブロック図を示す。
図４は、図３に示した32個のCPUのシステムにおける、８個のCPUと４個のネットワーク・モジュールとの間の接続のより詳細なブロック図を示す。
図５は、図３に示した32個のCPUのシステムにおける、ネットワーク・モジュールと８個のメモリ・モジュールとの間の接続のより詳細なブロック図を示す。
図６は、メモリ・モジュールのブロック図を示す。
図７は、図６に示したメモリ・モジュールにおける出力経路（outgoing path）のより詳細なブロック図を示す。
図８は、図６に示したメモリ・モジュールにおける入力経路（return path）のより詳細なブロック図を示す。
図９は、ネットワーク・モジュールのブロック図を示す。
図10は、図９に示したネットワーク・モジュールにおける出力経路のより詳細なブロック図を示す。
図11は、図９に示したネットワーク・モジュールにおける入力経路のより詳細なブロック図を示す。
図12は、レディ／レジューム（READY/RESUME）のハンドシェイクのための制御回路の詳細構成を示す。
図13は、健全性コードの制御回路の詳細構成を示す。
図14～15は、本発明の相互接続ネットワークを使用することができるマルチ処理システムのいくつかの例についての種々の構成を示す表である。
図16は、１×２スイッチの遅延チェイン（delay chain）の詳細を示す。
＜好ましい実施形態の詳細な説明＞
以下の詳細な説明では、本文の一部を構成すると共に、例として本発明の特定の実施形態を示した添付図面を参照することにする。しかし、他の実施形態を利用してもよく、本発明の範囲を逸脱しない限り構造的または論理的な変更を施すこともできる。したがって、以下の詳細な説明は限定された意味で解されるべきではなく、本発明の範囲は添付した請求の範囲によって定めるべきである。
図１を参照しつつ、マルチ処理システムの一例の簡略ブロック図を説明する。図１は、４個のメモリ・モジュール600a－ｄから成る共通メモリに接続された４個のCPU100を示す。
図２は、図１のシステムのより詳細なブロック図を示す。４個のCPU100a－ｄのそれぞれは、ポート０とポート１の二つのポートを持っている。ポート０とポート１のそれぞれは、４個のメモリ・モジュール600a－ｄのうちの一つに経路が設定され、CPU1個当たり全部で８出力に対し、メモリ・モジュール１個当たり８入力に経路が設定される。
マルチ処理システムの他の例を図３に示す。図３は、32個のCPUを有する最大規模のシステムである。８個のCPUから成る４個のCPU群100a－ｈ、100i－ｐ、100q－ｘ、100y－ffが、４個のネットワーク・モジュールから成る４個のネットワーク・モジュール群300a－ｄ、300e－ｈ、300i－ｌ、300m－ｐを介して、８個のメモリ・モジュールから成る４個のメモリ・モジュール群600a－ｈ、600i－ｐ、600q－ｘ、600y－ffに接続されている。プロセッサをメモリに接続するこの相互接続ネットワークを使用して構築することができる４個、８個、および32個のCPUのシステム例を以下において更に詳細に説明する。
図１～３は、ネットワーク・モジュール300およびメモリ・モジュール600のアーキテクチャのモジュール性により、如何にして、単純な複製で任意の規模のシステムの構築が可能となるのかを示している。１～４個のCPUを有するシステムにおいては、それらのCPUは図１および２に示すようにメモリ・モジュール600に直結される。４個よりも多くのCPUを有するシステムにおいては、それらのCPUは、まずネットワーク・モジュール300に接続され、そのネットワーク・モジュールがメモリ・モジュール600に接続される。最小のマルチ処理システムを構成するものと同一のネットワーク・モジュールおよびメモリ・モジュールを、修正することなく、最大のシステムを構築するために使用することができる。モジュール性と拡張性は本相互接続ネットワークの設計に本来的に備わっているものであるた

め、それらはシステム実現のために選択される技術に依存せず、極めて柔軟性に富み、その結果、容易に規模を変更できるマルチ処理システムが得られる、ということが当業者には容易にわかるであろう。

４個、８個、および32個のCPUを有するマルチ処理システムの例を説明する前に、メモリの相互接続ネットワークを構成するメモリ・モジュール600およびネットワーク・モジュール300について詳細に説明する。一般に、メモリ・モジュール600およびネットワーク・モジュール300は、ｎ×ｍスイッチの集合から構成される。これらのｎ×ｍスイッチは、種々の形態に接続されてメモリおよびネットワークのモジュールを形成し、それらは、プロセッサと共通メモリとの間にメッセージの経路を設定する。相互接続ネットワークを構築するために同一のｎ×ｍスイッチ群が繰り返し使用されるように、そのスイッチが作製される。したがって、その結果得られる構造はモジュール性が高く、小規模のマルチ処理システムを構築するために使用されるものと同一の構成要素が大規模なシステムにおける接続にも使用できるようになっている。

＜メモリ・モジュール＞

次にメモリ・モジュール600について詳細に説明する。図６は、メモリ・モジュール600のブロック図を示す。出力経路（outgoing path）604および入力経路（return path）606の双方が示されている。各メモリ・モジュール600は、８個の入力０～７と16個のメモリ・バンク640a～640pとを備えている。実現の容易化のため、このメモリ・モジュールは、論理的に二つの独立部分に分離されている。入力０～３はメモリ・バンク640a～640h（上半部）へ経路が設定されており、一方、入力４～７はメモリ・バンク640i～640p（下半部）へ経路が設定されている。これら二つの部分は、完全に分離されていて、互いに影響されることはない。物理的に、これら二つの部分は、一つのプリント回路基板の上部側と底部側に配置されるのが好ましい。しかし、本相互接続ネットワークは、この物理的な実現方法に限定されないと理解すべきである。すなわち、多くの他の物理的な所定の実現が当業者によって容易に考えられるであろう。

入力０～７のそれぞれは、まず１×１スイッチ610によってバッファリングされる。４個のスイッチ610から成る２個のスイッチ群は、２個の４×４スイッチ620のうちの一つの入力へ接続される。２×２スイッチ420は、４個の入力のそれぞれから４個の１×２スイッチ630の一つへ経路を設定する。１×２スイッチ630のそれぞれは、一つの入力から２個のメモリ・バンク640の一つへ経路を設定する。

各メモリ・モジュール600の入力経路606において、２個のメモリ・バンクは２×１スイッチ650へ経路が設定されている。４個の２×１スイッチ650は、４×４スイッチ660の入力へ接続され、その後、それらは４個の１×１スイッチ670の一つへ経路が設定され、全部で８個の出力０～７へ経路が設定されている。

図７は、メモリ・モジュール600の出力経路側604のより詳細なブロック図を示す。上半部および下半部の双方が示されている。メモリ・モジュール600の入力０～７のそれぞれは、CPU100からそのメモリ・モジュールへのメモリ参照をバッファリングする１×１スイッチを有している。各１×１スイッチ610におけるバッファの数は、クロスバー・スイッチ間での通信の往復時間によって変わる。各クロックの周期がその通信時間に付加されるため、各１×１スイッチ610内のバッファを更に追加する必要がある。１×１スイッチ610の目的は、メッセージ・パケットを多重に送ることができるように、クロスバー・スイッチ間の通信時間をカバーすることにある。したがって、図６に示した本メモリ・モジュールには、６個のバッファＡ～Ｆが設けられていて、ｎ×ｍスイッチがクロックの１周期の期間で通信を行う。

１×１スイッチ610におけるバッファＡ～Ｆ、および本相互接続ネットワークを構築するために使用される全てのｎ×ｍスイッチにおけるバッファは、先入れ先出し（FIFO）のキュー構造体（queue structure）を有している。入力０～７のうちの一つを経由してメモリ・モジュール600に入るメッセージは、その経路用の対応するバッファＡに他のメッセージが既に存在するということがなければ、そのバッファＡに保持される。バッファＡが占有されている場合には、そのメッセージは、バッファＢが使用可能であれば、バッファ

Ａの代えてバッファＢに保持される。バッファＡがいったん空になると、バッファＢに存在するメッセージはバッファＡへ移される。別のメッセージがアクセスしようとしたときバッファＡおよびバッファＢの双方が満杯であれば、そのメッセージはバッファＣに入る。このようにして入力バッファは、先入れ先出しキューとして動作する。別のメッセージがその入力バッファ610に入ろうとしたときバッファＡ～Ｆが全て満杯であれば、そのメッセージは、メモリ・モジュールに入る前に、FIFOバッファの一つが解放されるまで単に待機するだけである。入力保持バッファ（input holding buffer）621が他のメッセージを受け取る用意ができているときは、１×１スイッチ610のバッファＡ内のメッセージは、４×４スイッチの入力保持バッファ621の中に保持される。

いったんメッセージが入力保持バッファ621に保持されると、バッファ621のバッファＡは２個のメッセージ・ステアリング・ビット（message steering bit）を解読してそのメッセージの経路を４個の出力要求バッファ（output request buffers）のいずれに設定すべきかを決定する。２個のメッセージ・ステアリング・ビットを解読した後、バッファＡは、最初のパケット内の２個のメッセージ・ステアリング・ビットを、８個の入力０～７のうちいずれの入力からそのメッセージがメモリ・モジュール600に入ったかを表す２個のビットに置き換える。このようにして、メッセージがメモリ・バンク640へ行く途中においてメモリ・モジュールの層を順次通過して行く間に、発信元のCPUへ戻すリターン・アドレス（return address）がそのメッセージ・ステアリング・ビットに組み込まれる。データが適切なメモリ・バンクから取り出された後、その取り出されたデータを発信元のプロセッサまで案内するために、プロセッサ・リターン・アドレス情報をそのメッセージ・ステアリング・ビットから得ることができるようになる。同様に、一旦そのメッセージがプロセッサに戻ると、その参照がどのメモリ・バンクから来たのかを示すリターン・アドレスをそのメッセージ・ステアリング・ビットから得ることができるようになる。この方法では、データ中に誤りがあれば、そのプロセッサは誤りのあるメッセージがどのメモリ・バンクから来たのかを知り、これにより、その誤りがどこで生じたのかを正確に指摘するための助けとなる。

このビット置換方式は、プロセッサ・リターン・アドレスをメッセージとともに送る必要がなくなるため、システムにおける相互接続およびメッセージ長を低減する。その代わりに、本発明の好ましい実施形態におけるビット置換方式は、出力経路上でメッセージを導き、自動的にプロセッサまたはメモリ・バンクのリターン・アドレスを生成するために、同じビットを使用して相互接続を行うだけである。

入力保持バッファ621のバッファＡ内で解読されたメッセージ・ステアリング・ビットは、メッセージが入力保持バッファＡを出た後に４個の出力要求バッファ622a－ｄのうちのいずれに行くのかを決定する。出力要求バッファ622の目的は、相互接続ネットワーク内におけるメッセージ競合およびブロッキングを低減することにある。例えば、群バッファ（group buffer）623aが満杯であれば、出力要求バッファ内で待機しているメッセージはブロックされ（封鎖され）、その経路が空くのを待っているメッセージが存在することになる。しかし、例えば、異なる群バッファ623b、ｃまたはｄに向かう別のメッセージが、入力０からメモリ・モジュール600に入ると、そのメッセージは群バッファ623aに対して待機しているメッセージによってブロックされることなく、適切な出力要求バッファ622の中へ入ることができる。このように、他のいずれかの群バッファ623に向かう異なるメッセージが入力０に入った場合、そのメッセージは、適切な出力要求バッファ622に入って行くだけであるため、群バッファ623へのアクセスをブロックされることがない。このようにして、より多くのメッセージはクロスバー・ネットワークにおけるブロックを「避けて通る」ことができる。相互接続ネットワークを通過する特定の経路が１×１スイッチ610を通る全ての通路でブロックされている場合にのみ、その経路に入ったメッセージは全ての出力経路からブロックされる。もし、４×４スイッチ620内に出力要求バッファ610が存在しなければ、４個の出力は全てブロックされるであろう。出力要求バッファ622を使用することにより、４個の出力の経路のうち一つのみがブロックされる。出力要求バッファは、各ｎ×ｍスイッチおよびネットワークにおける、メッセージ競合およびブロッキ

ングの可能性とメモリ・モジュール自体とを大いに低減する、ということが当業者には容易に理解できるであろう。
メッセージが出力要求バッファ622に入った後は、そのメッセージは適切な群バッファ623a－ｄに導かれる状態になる。各群バッファ623は、各入力の経路に対応する出力要求バッファ622からメッセージを探す。例えば、群バッファ623aは、入力０、１、２、および３から４×４スイッチ620にそれぞれ入った適格なメッセージを、その対応する出力要求バッファ622a、622e、622iおよび622mから探す。適切な出力要求バッファ622内に適格なメッセージが一つだけ存在する場合は、群バッファ623はそのメッセージを対応する１×２スイッチ630へ送るだけである。好ましい実施形態では、適切な出力要求バッファ622内に適格なメッセージが複数存在すれば、群バッファ623は、最後に選ばれた出力要求バッファからラウンドロビンの順序での次のメッセージが送られる。例えば、群バッファ623aが転送のために出力要求バッファ622bを選択した場合において、もし、出力要求バッファ622cが適格なメッセージを有していて、適格なメッセージが複数存在すれば、その群バッファは、転送のために出力要求622cを次に選択するであろう。
一旦或るメッセージが４×４スイッチ620の群バッファ623を通過すると、そのメッセージは、対応する１×２スイッチ630へ進む。好ましい実施形態では、１×２スイッチ630に入るメッセージは、２個のメモリ・バンク640のうちの一つに経路が設定される。
或るメッセージが１×２スイッチ630のバッファＡに入った後は、バッファＡはメッセージ・ステアリング・ビットを解読して、そのメッセージを２個のバンク入力バッファ（bank input buffer）632aまたは632bのうちの一つに導く。各バンク入力バッファ632は、異なるメモリ・バンク640に対応づけられている。例えば、１×２スイッチ630aに入るメッセージは、メモリ・バンク04へ経路を設定されるべきものであって、バンク入力バッファ632bへ導かれる。この段階において、メッセージは、バンク制御論理チップにより選択されて適切なメモリチップにアクセスできる状態になる。
メモリ・モジュール600から要求元のCPUへ戻るために、参照は、上述のようにプロセッサからメモリへの出力経路側参照がメモリ・モジュールの出力経路側604を通って進むのと同様にして、そのメモリ・バンクを出て、メモリ・モジュール600の適切な入力経路側606を通って発信元のプロセッサに戻る。メモリ・モジュール600の入力経路側606の詳細を図８に示す。この場合もまた、上半部と下半部の双方が示されている。入力経路参照（return reference）は、メモリ・バンク616を出た後、２×１スイッチ650に入る。各２×１スイッチ650は８個のバッファFIF0652を備えている。このスイッチにおけるFIF0は、パイプライン方式のメモリ部分の可能性に対応できるように８段の深さがなければならない（これについては下記において詳細に論じられる）。各２×１スイッチ650は、２個の対応するメモリ・バンクからメッセージを受け取り、そのメッセージを８個のバッファFIF0によってバッファリングし、そのメッセージの経路を４×４スイッチ660の入力へと設定する。４×４スイッチ660は、図５を参照しつつ上記において説明された４×４スイッチ620と同じように動作する。リターン・メッセージ（return message）は、４×４スイッチ660から適切な１×１出力バッファ670へと経路が設定される。１×１スイッチ670は、図５を参照しつつ上記において説明された１×１スイッチ610と同じように動作する。このリターン・メッセージは、対応する出力０～７からメモリ・モジュール600を出て行く。
＜ネットワーク・モジュール＞
図９は、ネットワーク・モジュール300のブロック図を示す。各ネットワーク・モジュールは、16個の入力０～15と16個の出力０～15とを有している。実現の容易化のため、各ネットワーク・モジュールは二つの部分に論理的に分離されている。入力０～７はメモリ・モジュール600a－ｈ（上半部）へと経路が設定され、入力８～15はメモリ・モジュール600i－ｐ（下半部）へと経路が設定されている。メモリ・モジュールに関し上記で説明したように、これら二つの部分は完全に分離されていて、互いに影響されることはない。物理的に、これら二つの部分は、プリント回路基板の上部側と底部側に実装されるのが好ましい。しかし、本相互接続ネットワークは、ここに述べられた物理的な実現方法に限定されるものではなく、多くの他の物理的な所定の実現が当業者にとっては明白であろう。

或る特定のネットワーク・モジュール（図３参照）に接続された８個のプロセッサのそれぞれは、メモリに対して二つのポートを持っていて、それらは、16個の１×１スイッチ310a－ｐを介してネットワーク・モジュール300の入力へ接続されている。例えば、CPU0に対する二つのポートは、１×１スイッチ310aおよび１×１スイッチ310iを介して接続されている。そして各１×１スイッチ310は４個の４×４スイッチ320a－ｄのうちの一つに接続されている。そして各４×４スイッチは、その４個の入力のそれぞれから４個の出力の一つへと経路を設定し、出力側は、８個の２×２スイッチ330a－ｈで構成されている。そして８個の２×２スイッチ330a－ｈのそれぞれは、16個のメモリ・モジュール600a－ｐの一つへの入力を有している。

メモリ・モジュール300の入力経路304は、出力経路302上におけるものと同様のこれらのスイッチから構成されている。16個の１×１スイッチ350a～350pは、16個のメモリ・モジュール600a－ｐの一つに接続されている。１×１スイッチ350は、１×１スイッチ310aと同じように動作し、それらと同様の構造となっており、上記において図６～８を参照しつつ１×１スイッチ610および670について論じられたのと同様である。４×４スイッチ360a－ｄは、それぞれ、４個の１×１スイッチ350から成るスイッチ群からの入力を受け取り、その４個の入力のそれぞれに対し、４個の２×２スイッチ370の一つへ経路を設定する。４×４スイッチ360は、ネットワーク・モジュール300の入力側302における４×４スイッチ320と同じように動作する。同様に、２×２スイッチ370は、ネットワーク・モジュール300の入力側302における２×２スイッチ330と同じように動作する。一つの２×２スイッチからの各出力は、８個のプロセッサ100a－ｈのうちの２個のプロセッサに接続された出力のうちの一つである。このようにして、図９に示すように、CPU100aのポート０は２×２スイッチ370aに接続され、CPU100aのポート１は２×２スイッチ370eに接続される。

図10は、ネットワーク・モジュール300の出力経路302のより詳細なブロック図を示す。図10は出力経路の上半部のみを示している。すなわち、入力310a～310hは示されているが310i～310pは示されていない。しかし、図10に示されていないネットワーク・モジュールの下半部は、図10および11を参照しつつ説明される上半部と同じように動作する。図10は、１×１スイッチ310、４×４スイッチ320、および２×２スイッチ330のそれぞれの内部構造を示している。図10に示された詳細より、１×１スイッチ310は、図６～８を参照しつつ上記において説明された１×１スイッチ610aおよび１×１スイッチ670と同じように動作し、それらと同様の構造となっている。また、４×４スイッチ320は、図６～８を参照しつつ説明された４×４スイッチ660および620と同じように動作し、それらと同様の構造となっている。

２個の４×４スイッチ320aおよび320bからの４個の出力のそれぞれは、４個の２×２スイッチ330a－ｄのうちの一つへ経路が設定されている。２×２スイッチ330のそれぞれは、４個のバッファを有する２個のFIFO331aおよび331bを有しており、これらのFIFOのそれぞれは、メッセージの経路を２個のモジュール・バッファ332aおよび332bへ設定している。FIFO331におけるバッファＡは、メッセージ・ステアリング・ビットを解読して、そのメッセージに対し適切なモジュール・バッファ332へ経路を設定する。各モジュール・バッファは、メモリ・モジュールの一つに対する入力となっている。

図11は、ネットワーク・モジュールを通過する入力経路304を示す。図10と同様に、図11は、ネットワーク・モジュール300における入力経路の上半部のみを示している。図11に示された詳細より、１×１スイッチ350は、図10に示された１×１スイッチ310、図７に示された１×１スイッチ610、および図８に示された１×１スイッチ670と同じように動作し、それらと同様の構造となっている。同様に、４×４スイッチ360は、図10に示された４×４スイッチ320、図８に示された４×４スイッチ660、および図７に示された４×４スイッチ620と同じように動作し、それらと同様の構造となっている。最後に、２×２スイッチ370は、図10を参照しつつ上記において説明された２×２スイッチ330と同じように動作し、それと同様の構造となっている。

＜マルチ処理システムの一例についての詳細な説明＞

上記の説明より当業者にとっては、メモリ・モジュール600およびネットワーク・モジュ

ール300の設計におけるモジュール性は明白であろう。最小のマルチ処理システムを構成するものと同一のｎ×ｍスイッチを、修正することなく、最大のシステムを構築するために使用することができる。したがって、モジュール性と拡張性は本ネットワーク・モジュールおよびメモリ・モジュールの設計に本来的に備わっているものであるため、それらはシステム実現のために選択される技術に依存せず、また、極めて柔軟性に富み、容易に規模を変更できるマルチ処理システムを得ることができる、ということが当業者には容易に理解できるであろう。

本発明の相互接続ネットワークを用いて構築することができるマルチ処理システムの三つの例を詳細に説明する。

再び図１を参照しつつ、４個のCPUを有するマルチ処理システムを説明する。図２は、各CPU100からの８個の出力のそれぞれに対し共通メモリ602への経路を設定する方法の詳細を示している。図２に示された接続により、各CPUは共通メモリ602内の全てのメモリ・バンクにアクセスできるようなる。図２は、例えば、CPU100aのポート０および１がメモリ・モジュール600aの入力０および４にそれぞれ接続されていることを示している。図６は、入力０がメモリ・バンク640a－ｈへのアクセス経路を有し、入力４がメモリ・バンク640i－ｐへのアクセス経路を有することを示している。このように、各メモリ・バンクは、各CPU100のポート０またはポート１のいずれかに対応づけられる。CPU100aのポート０および１は、同様にしてメモリ・モジュール600b－ｄにも接続されている。CPU100b－ｄも同様にして各メモリ・モジュール600に接続され、これにより、各CPUが共通メモリ602内のメモリ・バンク640の全てにアクセス経路を有するようになっている。

次に図３を参照しつつ、CPUが32個のマルチ処理システムを説明する。32個のCPU100a－ffとネットワーク・モジュール300a－ｐとの間の接続の細部を図４により詳細に示す。図４は、８個のCPUと、ネットワーク・モジュール300a、300e、300iおよび300mの間の、それらのCPUに関連する相互接続とを示している。CPU100a－ffは、各CPUがメモリ・モジュール600の各メモリ・バンクへのアクセス経路を有するように、ネットワーク・モジュール300に接続されている。このようにして、CPU100aのポート０は、ネットワーク・モジュール300aの入力０、ネットワーク・モジュール300eの入力０、ネットワーク・モジュール300iの入力０、およびネットワーク・モジュール300mの入力０に接続される。図９に示すように、ネットワーク・モジュール300において、CPU100aのポート０は入力０経由でメモリ・モジュール600a－ｈへアクセスする経路を有し、CPU100aのポート１は入力８経由でメモリ・モジュール600i－ｐへアクセスする経路を有している。図３に示された32個のCPUのシステムにおける残りのプロセッサも同様にして接続され、各CPUが共通メモリ602内の全てのメモリ・バンクへの経路を有するようになっている。

図５は、図３に示された32個のCPUのシステムの一例における、ネットワーク・モジュール300a－ｄとメモリ・モジュール600a－ｈとの間の接続をより詳細に示している。各メモリ・モジュール600a－ｈは、４個のネットワーク・モジュール300a－ｄのそれぞれにおける２個の出力に接続されている。このネットワーク・モジュールとメモリ・モジュールは、32個のプロセッサのいずれからの参照も共通メモリ602内のいずれのメモリ・バンクにも到達できるように接続されている。

例えば、CPU100dからの参照に対しメモリ・モジュール600hのメモリ・バンク640eへ経路を設定するために（図３参照）、CPU100dのポート０はネットワーク・モジュール300aの入力３へ経路が設定される（図４参照）。ネットワーク・モジュール300aからは、メッセージがネットワーク・モジュール300aの出力７より出される（図５に示されている）。この出力はメモリ・モジュール600hの入力０に接続されている。次に図６を参照すると、各メモリ・モジュール600の入力０は、それぞれのバンク640eへの経路として、４×４スイッチ620aおよび１×２スイッチ630c経てバンク640eへ到達するという経路が設定されている。再び図３を見ると、CPU100qからの参照に対し、ネットワーク・モジュール300gを経由して、または、いずかのCPUとメモリ・バンクとの間で上述の経路と同様の相互接続ネットワーク内の経路に沿って、メモリ・モジュール600iへの経路を設定することができる。

４個よりも多いが32個よりも少ないプロセッサを有するシステムは、図３の32個のCPUのシステムを変形したものを用いて構築することができる。例えば、８個のCPUのシステムを、ネットワーク・モジュール300aを介してメモリ・モジュール600a－ｈに接続されたCPU100a－ｈで構築することができる。図４および５に示すように、８個のCPUのシステム内の各CPUの各ポートに対し８個のメモリ・モジュール600a－ｈへの経路を提供するためには、各CPU100a－ｈの出力０および１のみが必要である。したがって、各CPUの出力２～７は無効なものとなる。

８個のCPUのマルチ処理システム内のネットワーク・モジュール300aは、図５に示すように８個のメモリ・モジュール600a－ｈのそれぞれに接続されている。ただ一つのネットワーク・モジュールのみが必要であるため、各メモリ・モジュールの入力０および１のみが使用される。残りの入力は接続されず、したがって残りの経路におけるどのデータも無効である。

８個のCPUと８個のメモリ・モジュールを有するマルチ処理システムのようなシステムでは、CPUの全ての出力が使用されているわけではなく、そのようなシステムに対しては、相互接続ネットワークにおける各経路上のデータが有効か否かを示す機構が設けられている。使用されていないそれらの出力に対しては、ノイズまたは他の無効な情報が所定のプロセッサからメモリ経路へ送られる。もし、メモリ・バンクがこれらの無効な信号を或るメモリ位置に対して読み出すべき又は書き込むべき命令またはデータと解釈すれば、エラーが発生する可能性がある。

有効なデータが各信号線に存在することを保証するために、プロセッサの有効な出力のそれぞれは、シリアルなビット・ストリームを相互接続ネットワークの最初のスイッチに送り込む。このシリアルなビット・ストリームは、その出力からのデータが有効か否かを示すために、その中に「健全性コード（sanity code）」が埋め込まれている。この健全性コードは、プロセッサの有効な出力のそれぞれに相互に接続された最初のネットワーク・スイッチへシリアルにかつ連続的に送り出される６ビットのパターンである。したがって、図２に示した各CPUに対しては、CPUの有効な出力のそれぞれが、それぞれのメモリ・モジュール600の最初のスイッチへ健全性コードを送る。図３に示した各CPUに対しては、CPUの有効な出力のそれぞれが、それぞれのネットワーク・モジュール300の最初のスイッチへ健全性コードを送る。この健全性コードの目的は、相互接続ネットワーク内のモジュール間のインタフェイスを保護することである。正しい健全性コードが受け取られると、その入力がアクセス経路を有している８個の出力の全てに、その受け取ったものが同報送信される。このようにして、例えば、図７では、問題となっているCPUの出力がメモリ・モジュール600の入力２に接続されていれば、１×１スイッチ610cが健全性コードが正しいか否かを調べ、正しい健全性コードを受け取っていれば、それを、入力２がアクセス経路を有している８個のメモリ・バンク640a－ｈへ同報送信する。

健全性コードを監視するための制御回路を図13に示す。各モジュールの最初のスイッチにおける健全性検査回路700は、受け取った健全性コードを監視して、モジュールのその入力に入って来たデータが有効か否かを判定する。メモリ・モジュール600に対しては、各入力０～７に対応する各１×１スイッチ610a－ｈ内に健全性検査回路700が存在する。ネットワーク・モジュール300に対しては、各入力０～15に対応する各１×１スイッチ310a－ｐ内に健全性検査回路700が存在する。

健全性検査回路700は、対応するスイッチのFIFOをイネーブルにするものとして動作する。健全性コードが正しく受信されなければ、そのスイッチ内のFIFOは使用可能とはされず、そのスイッチに入る如何なる情報もFIFOバッファの中へは格納されない。

無効なCPUによって又はネットワーク・モジュールもしくはメモリ・モジュールの未接続の入力によって生じるノイズが健全性コードと同じになるということが極めてまれであるように、健全性コードを選択しなければならない。したがって、６ビットの健全性コードの適切な一例は、

010011

である。

上述のように、この６ビットの健全性コードは、CPUの有効な出力によってのみ、シリアルにかつ連続的に送り出される。その６ビットのコードがシリアルに16回連続して受け取られたと健全性検査回路が判定すると、一つのメモリ・バンクが動作可能とされて、相互接続ネットワークの所定の経路上のデータに注意を払うようになるだけである。如何なるときであっても正しい健全性コードが受け取られない場合は、そのメモリ・バンクは、正しい６ビットのコードが再びシリアルに16回連続して受け取られるまで、その経路上のデータを無視する。

結果として得られる96ビットのコード（６ビットのコードが16回繰り返されたもの）は、非常に明瞭な周波数とその中に組み込まれた信号パターン（signalling patterns）を有する信号である。これらの周波数および信号パターンがノイズと同じとなるのは極めてまれである。

本相互接続ネットワークのモジュール性により、小規模システムを構築するために使用されるのと同一のモジュールを、修正を加えずに、最大規模のシステムを構築するために使用することができる。図13および14は、本発明の相互接続ネットワークを使用して構築することができるマルチ処理システムのいくつかの例を記載した表を示している。図13は、１～４個のCPUを有するマルチ処理システムについてのモジュール数、セクション構成（section configurations）、およびアドレス・ビットの割り付け（address bit layout）を記載した表を示している。図14は、８～32個のCPUを有するマルチ処理システムの例に対する表を示している。図13および14に示したシステムおよびこの中で説明されたシステムの例は、決して、本相互接続システムを使用して構築できる唯一のマルチ処理システムではない。それどころか、無限の種類の構成が本発明の範囲を逸脱することなく構築可能であろう。

＜パケット化メッセージ＞

好ましい実施形態では、書込情報（write references）は、２個の連続したパケットでメモリ・モジュール600へCPUによって送られる。読出情報（read references）は、単一のパケットで転送される。書込情報に対しては、１番目のパケットが必要なアドレスおよび制御情報の全てと書込データの半分とを収容している。２番目のパケットは、書込データの後半部を収容している。共通メモリへの書込については時間は本質的なものではないため、この方法は、ネットワーク・モジュールおよびメモリ・モジュールの出力経路において必要とされる相互接続を最小化する。

好ましい実施形態では、２番目のパケットは１番目のパケットに追随して、直後のクロックの周期でスイッチに入ってそれを通過し、１番目のパケットと同一の相互接続経路に沿って進む。このため、各データ移動にはクロックの２周期の期間を要する。このように、２番目のパケットは、１番目のパケットに対してちょうどクロックの１周期分だけ遅れて、メモリ・モジュールおよびネットワーク・モジュールの各ｎ×ｍスイッチ内に存在する。

書込情報をパケット化して、それらにネットワーク・モジュールおよびメモリ・モジュールを順次巡回させることにより、各ｎ×ｍスイッチの実現に要する相互接続の数が1/2に低減される、ということが当業者には容易にわかるであろう。現在の設計では、チップ内のゲートを利用し得る相互接続よりも、比率としては多くのゲートがチップに含まれている。したがって、相互接続は、多くのチップ設計において乏しい資源である。例えば、４個の独立した入力の経路および４個の独立した出力の経路を有する４×４スイッチと、120ビット幅のメモリ参照とを想定されたい。そのようなクロスバー・スイッチを作製するのに必要な相互接続の数は、この場合、120の８倍の数となる。これは極めて大きな数であって、単一チップに適合する数よりもはるかに大きく、実際、数個のチップに適合する数よりもはるかに大きい、ということが当業者には容易にわかるであろう。

メモリ参照をパケット化することにより、120ビット幅のメモリ参照が1/2に減少する。これにより、システムの全ての相互接続も1/2に減少する。これは極めて大きな意味を持ち得る低減である。本発明において使用されるこのパケット化方法は、各スイッチを単一チップ内に実現できるという点で、システム設計を大幅に簡潔にするものである、というこ

とが当業者には容易にわかるであろう。
しかし、読出情報は書込情報のようにはパケット化されない。出力経路上において、読出情報は制御情報と所望の読出アドレスのみを有している。読み出しの待ち時間をできるだけ短くすることが重要であるため、メモリ・モジュールおよびネットワーク・モジュールの入力経路606および304は、それぞれ、その幅が完全に１ワード分となっている。このため、各読出情報は、書込情報の出力経路のように２クロックを必要とするのではなく、各クロック毎に転送される。また、相互接続を更に節約するために、目的地コード（すなわち、プロセッサ・リターン・アドレス）を出力経路の読出情報の書込データのフィールドの中に入れて送る。この方法は、相互接続ネットワークの出力経路上で必要となる相互接続の数を更に節約する、ということが当業者には容易にわかるであろう。
＜ハンドシェイク・プロトコル＞
レディ／レジューム（READY/RESUME）のハンドシェイク・プロトコルにより、連続するｎ×ｍスイッチ層の間での円滑な通信が達成される。図12は、４×４スイッチ620に接続された１×１スイッチ610の制御回路の更なる詳細を示す。本ハンドシェイク・プロトコルは、図12に示した１×１スイッチおよび４×４スイッチを特に参照して説明されるが、スイッチ610および620は単に例に過ぎず、これらは、図６～11を参照しつつ上記において説明されたスイッチのいずれかの代表として考えている。
本相互接続ネットワークにおける各ｎ×ｍスイッチは、そのスイッチの各出力に対応づけられたカウンタを備えている。したがって、例えば、図12に示したスイッチ610は、その単一の出力の経路に対応づけられた１個のカウンタ612を備えている。４×４スイッチ620は４個のカウンタ624a－ｄを備えていて、それらのカウンタのそれぞれは、４個の群バッファ623a－ｄのうちの一つに対応している。目的スイッチ内のFIF0バッファの数に等しい値を最大のカウント値としている。したがって、例えば、カウンタ612は、４×４スイッチ620のFIF0621a内のバッファＡ～Ｄの数である４までカウントする。
これらのカウンタは、２個のスイッチの間で伝送されるレディおよびレジュームのハンドシェイク信号により、値が増加または減少する。レディ信号はカウンタの値を増加させる。レジューム信号はカウンタの値を減少させる。したがって、１×１スイッチ610は、１×１スイッチ610のカウンタ612の値が４よりも小さい間は更にメッセージを４×４スイッチ620に送信できることを知る。カウンタ612の値が４であれば、１×１スイッチは、４×４スイッチ620内のFIF0バッファ621aが満杯であり、１×１スイッチ610から更なるメッセージの受理はできないことを知る。
カウンタの値は０から始まる。一つのメッセージが１×１スイッチ610内のFIF0のバッファＡから送り出される毎に、レディ信号もそのメッセージの内容とともに４×４スイッチ620へ送られる。このレディ信号は二つの目的を持っている。第１にレディ信号は、一つのメッセージがバッファＡから送り出される毎にカウンタ612に入力される。レディ信号はカウンタ612の値を増加させる。第２にレディ信号は、FIF0621aにより、１×１スイッチ610から送られるメッセージ・データに対する有効信号として使用される。このカウンタの値は０から始まるため、１×１スイッチ610は、４個のメッセージを送ることができ、それらの送出後は４×４スイッチ620から送られるレジューム信号を待たなければならない。
４×４スイッチ620内のFIF0621aのバッファＡからメッセージが出て行くと、必ずレジューム信号が１×１スイッチ610へ送り返される。このレジューム信号はカウンタの値を減らし、FIF0621a内で一つのスペースが空いたことを示す。
レディ／レジュームのハンドシェイクは、この中で説明された各ｎ×ｍスイッチに対して同じように機能する。このようにして本相互接続ネットワーク内の各スイッチ層の間での円滑な通信が達成される。
＜メモリのタイミング＞
好ましい相互接続ネットワークは、広範囲のメモリ・アクセス時間およびサイクル時間での使用に適応できる。制御回路は、各メモリ・モジュール600の１×２スイッチ630内に配置されている。このスイッチの更なる詳細を図16に示す。図16は、FIF0バッファ631、ア

ドレス経路633、制御ビット経路634、および遅延チェイン（delay chain）636を示しており、これらの全ては、１×２スイッチ630内に存在する。遅延チェイン636は、マルチ処理システムに取り付けられたメモリの速度とサイクル時間に応じて、１×２スイッチ630における遅延を制御する。マルチ処理システムに取り付けられたメモリのアクセス時間は、８位置セレクタ637aによって１×２スイッチ630へ知らされる。セレクタ637aは、制御タグ（control tag）が入力635a－ｈのいずれに入るかを制御することにより、遅延チェイン636aの長さを制御する。この制御タグは、遅延チェインの８入力635a－ｈのうちの一つに入る。この制御タグは、目的地コード、リターン・アドレス、および他の制御情報を有している。セレクタ637aは、遅延チェイン636a内のいずれの場所に制御タグが入るかを制御する。

このタグが一旦遅延チェイン636aに入ると、このタグは、遅延チェイン636aの最後に到達するまで、そのチェインを流れていく。その後、この制御タグは、メモリ・バンクから出て来るメッセージ・データを採取してそのメッセージをFIFO652における最初の空いているバッファにロードする時点になると、メモリ・バンクの出力側における２×１スイッチ650に信号を送るために使用される。

メモリのサイクル時間は、メモリのアクセス時間と同様に制御される。８位置のセレクタ637bが、まさに上述のようにして同様の遅延チェイン636bを制御する。しかし、セレクタ637bおよび遅延チェイン636bはメモリのサイクル時間を制御するため、この遅延チェインの出力は、１×２スイッチ630のバッファ632aおよび632bまで送られて、メッセージがその１×２スイッチを出てメモリ・バンクへ行く頻度を制御する。

更なるセレクタ637cは、マルチ処理システムに取り付けられたメモリ部分がパイプライン方式のメモリ部分か、パイプライン方式ではないメモリ部分かを示す２位置のセレクタである。このように、本相互接続ネットワークは、異なるアクセス時間およびサイクル時間を有する多くの異なる種類のメモリを本相互接続ネットワークにおいて使用することを想定している。例えば、SRAMまたはDRAMを使用することができる。どのようなメモリが使用されるかに応じて、セレクタ637はそのメモリの種類に対応するように設定される。

本相互接続ネットワークの設計はマルチ処理システムのスケーラビリティ（scalability）を大きく増大させる、ということが当業者には容易に認識できるであろう。ネットワーク・モジュールおよびメモリ・モジュール内の各ｎ×ｍスイッチはその隣接部と通信するだけでよいため、この相互接続ネットワークは、極めてスケーラビリティに富んでいる。個々のｎ×ｍスイッチのそれぞれは、相互接続ネットワーク全体について、または、２つ若しくは３つ前方の層で起こっている事象についての大域的な情報を持つ必要はない。各ｎ×ｍスイッチは、CPUとメモリとの間で調停を行いデータを送るために、局所的な情報を持つだけでよい。同様に、CPUは、メモリ参照を送出する前に、相互接続ネットワークについて又はメモリの応答時間についての大域的な知識を持つ必要はない。この方法において、CPUは、そのCPUからメモリ・バンクまでのデータ経路全体が空いていることを確かめる必要はない。本相互接続ネットワークを使用すると、CPUは、簡単に「火をつけて忘れる（fire and forget）」ことができる。本相互接続ネットワークは大域的な調停回路（arbitrator）を必要としないことから、CPUや、ｎ×ｍスイッチ、オペレーティングシステムについての如何なる再設計も必要としないので、単純な複製によって極めて容易にシステムの規模を変えることができる、ということを当業者は理解するであろう。異なる構成における構成要素を接続し直すだけで、アーキテクチャの規模が直ちに変更される。

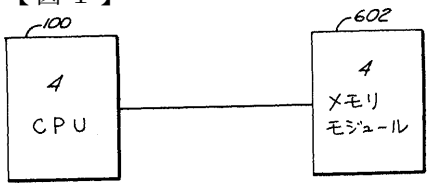
本相互接続ネットワークの他の利点は、この相互接続ネットワークは使用されるCPUの種類に依存せず、またメモリの速度にも依存しないということである。メモリの応答時間に依存しない如何なる種類のプロセッサの構成も、本発明のシステムに適応できる。このようなプロセッサ構成の一つは、同時係属中で本出願人に譲渡され、1994年12月28日に発行された、オバーリン（Oberlin）らによる「SCALABLE PARALLEL VECTOR COMPUTER SYSTEM（スケーラブルな並列ベクトル計算機システム）」という名称のEP公報第0553158号の中に見ることができる。参考までにこの出願をここに挙げておく。また、たとえメモリの全てのバンクが異なる速度で動作しても、本相互接続ネットワークは正しく機能し、複雑な

再設計を行う必要はない。したがって本相互接続ネットワークの設計はこの点において極めて柔軟性に富む、ということが当業者はわかるであろう。このように、本相互接続ネットワークで使用されるCPUは、時間に影響を受けやすいものではなく、したがって、メモリの待ち時間がより長い場合であっても効率よく動作する。本相互接続ネットワークの設計は、大量のデータがCPUとメモリとの間で移動できるように、バンド幅を広げるものである。このような設計方式により、極めて広いバンド幅で極めて高いスループットの計算機が得られる、ということを当業者は容易に理解するであろう。すなわち、これは、並列度の高いベクトル処理のタスクに特によく適合した設計である。

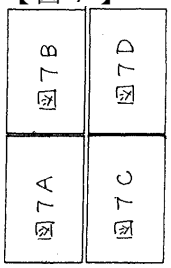
ここでは特定の実施形態が説明されているが、同一の目的を達成することを意図している如何なる構成も、その説明された特定の実施形態の代わりに用いることができる、ということを当業者は理解するであろう。例えば、本発明のアーキテクチャは、異なる数のプロセッサ、異なる量のメモリ、または、プロセッサ１個当たりの異なる数のプログラム・スレッド（program threads）で実現されてもよい。また、本発明のアーキテクチャは、使用されるプロセッサの特定の種類、メモリの速度、または、この詳細な説明において開示された特定の論理設計のいずれにも依存しない、ということが当業者にはわかるであろう。本発明の範囲を逸脱することなく、異なる種類のプロセッサを使用できるであろう。

この出願は、本発明の改造または変形をカバーするものである。したがって、この発明は請求の範囲によってのみ限定されることは明かである。

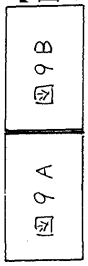
【図１】



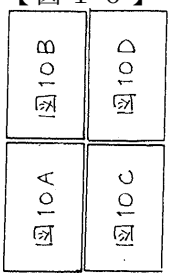
【図７】



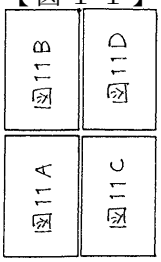
【図９】



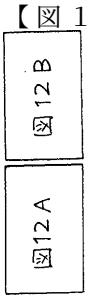
【図１０】



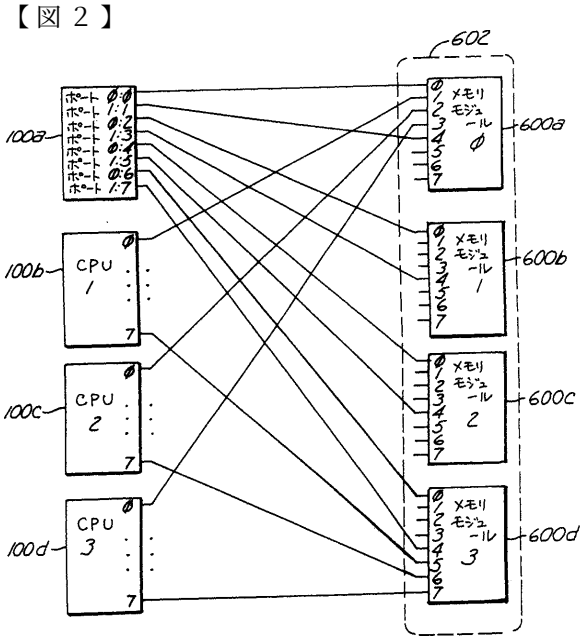
【図１１】



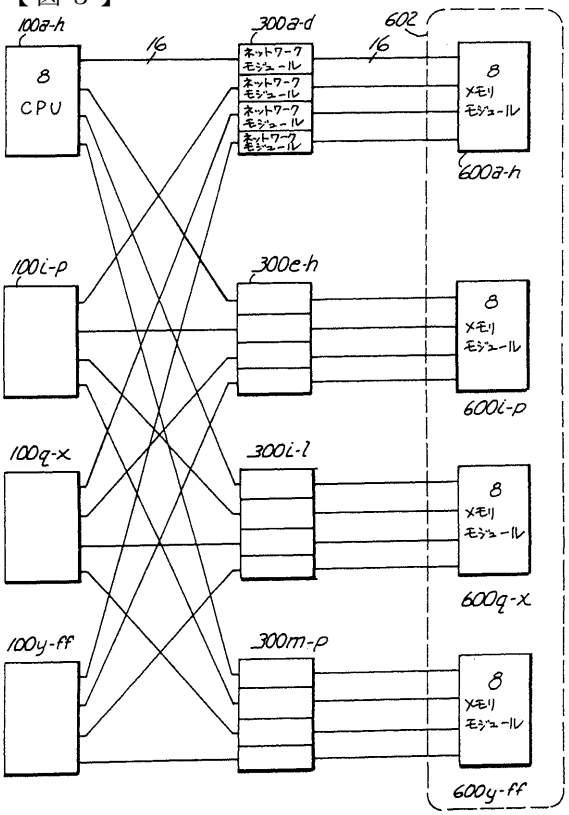
【図１２】



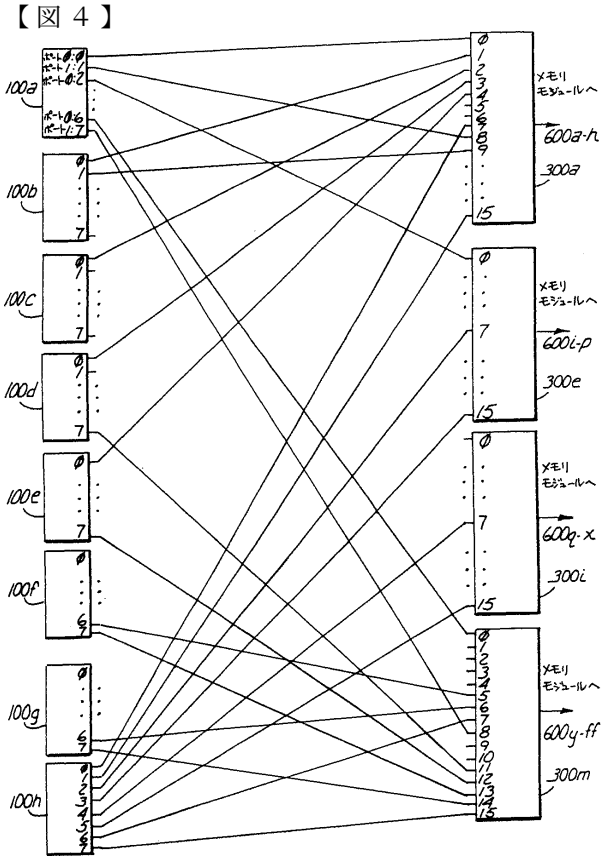
【図２】


602
100a
100b
100c
100d
CPU 0
CPU 1
CPU 2
CPU 3
メモリモジュール 0
メモリモジュール 1
メモリモジュール 2
メモリモジュール 3
600a
600b
600c
600d

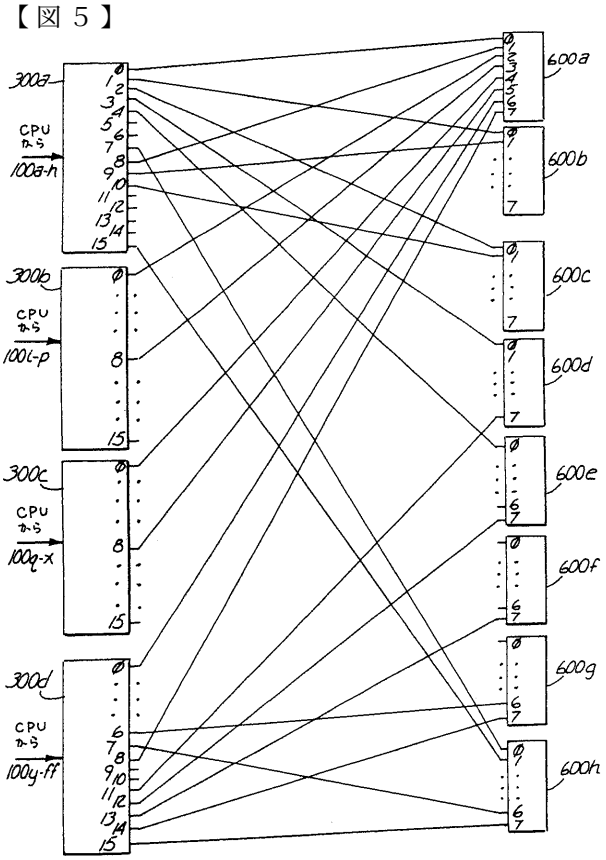
【図３】


100a-h
8 CPU
16
300a-d
ネットワークモジュール
602
8 メモリモジュール
600a-h
100i-p
300e-h
600i-p
100q-x
300i-l
600q-x
100y-ff
300m-p
600y-ff

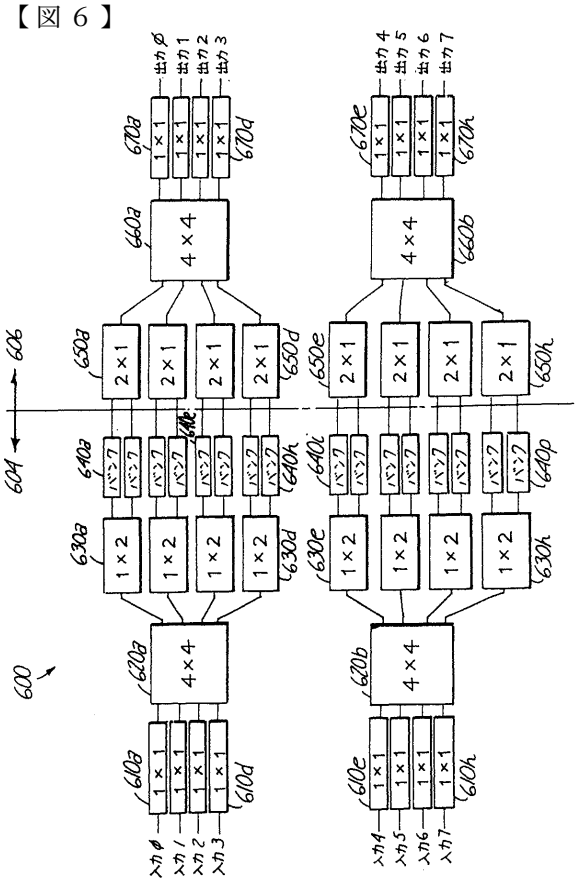
【図４】


100a
100b
100c
100d
100e
100f
100g
100h
メモリモジュールへ
600a-h
300a
600i-p
300e
600q-x
300i
600y-ff
300m

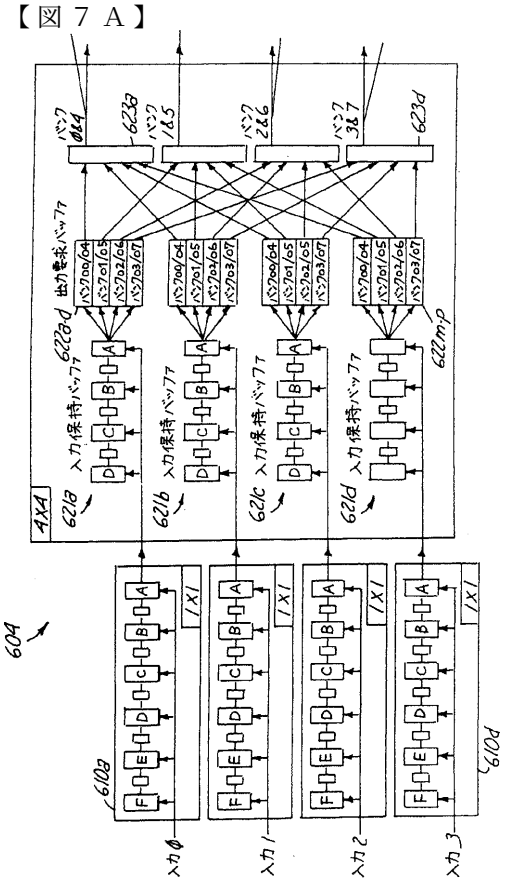
【図５】


300a
CPUから 100a-h
300b
CPUから 100i-p
300c
CPUから 100q-x
300d
CPUから 100y-ff
600a
600b
600c
600d
600e
600f
600g
600h

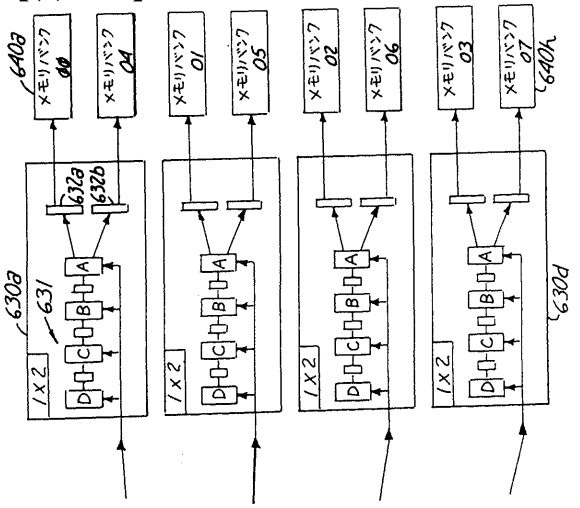
【図６】



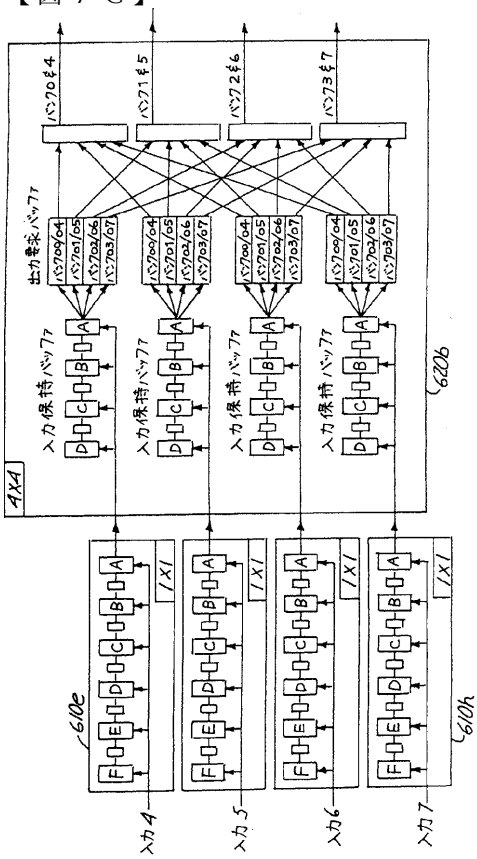
【図７Ａ】



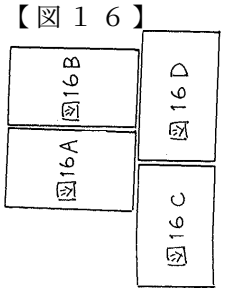
【図７Ｂ】



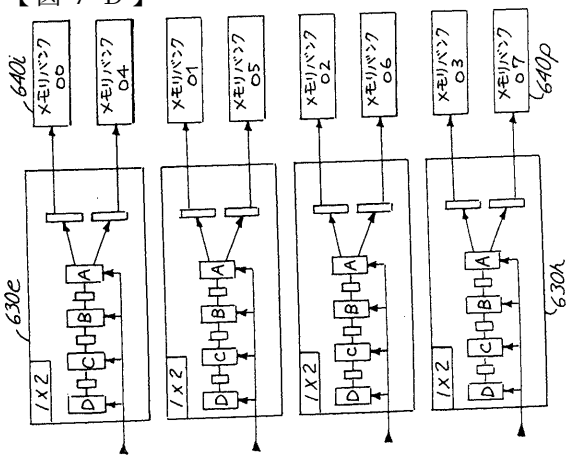
【図７Ｃ】



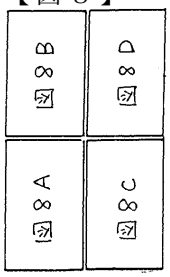
【図１６】



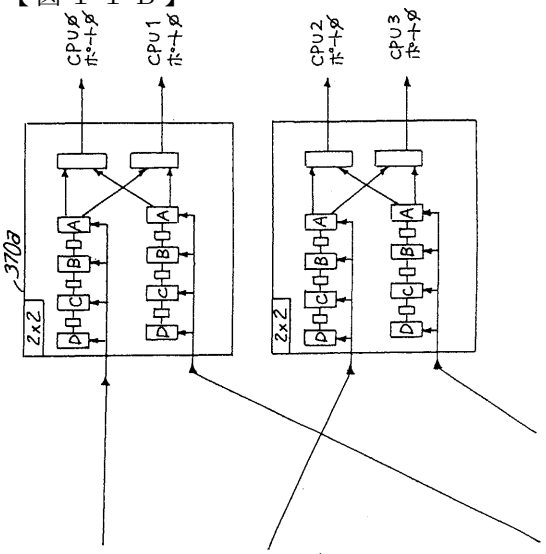
【図７Ｄ】



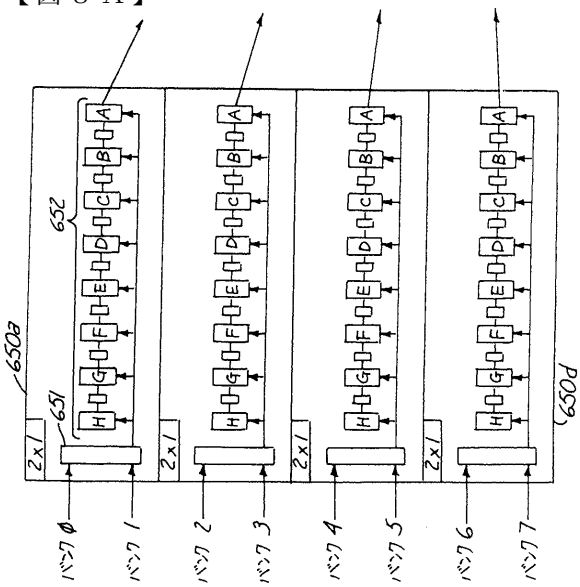
【図８】



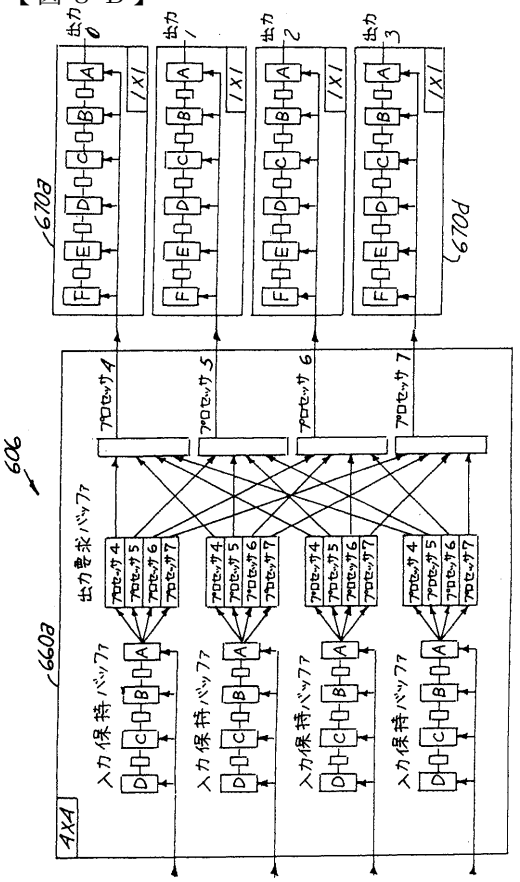
【図１１Ｂ】



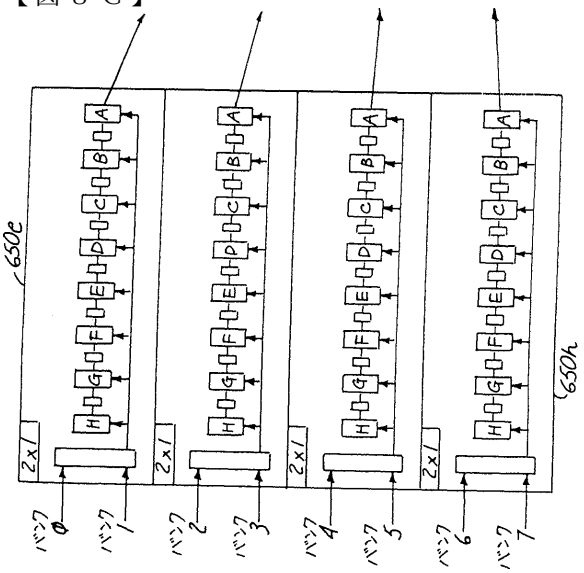
【図８Ａ】



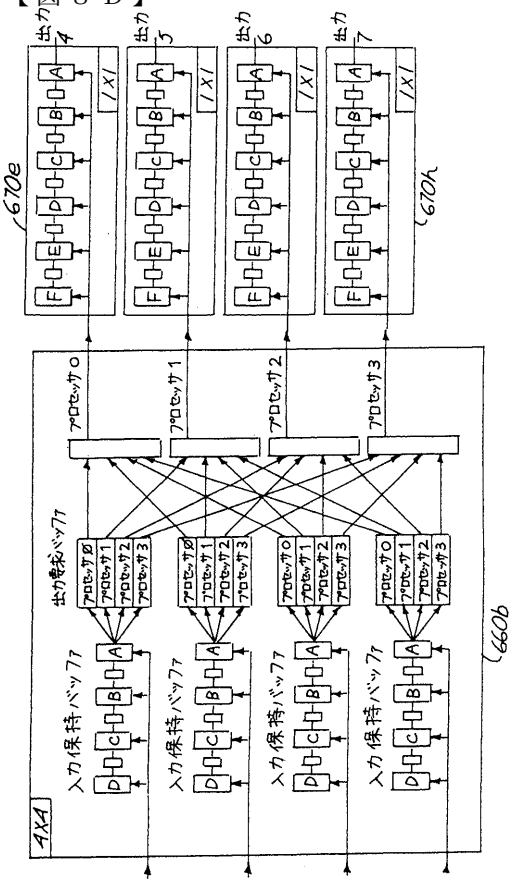
【図８Ｂ】



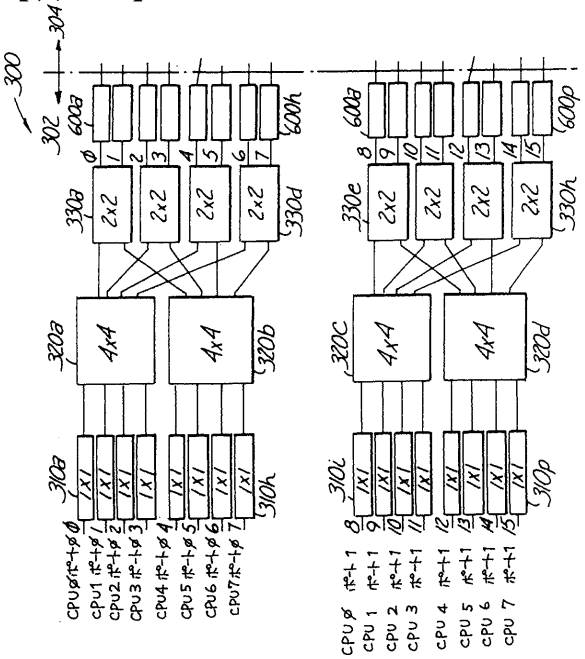
【図８Ｃ】



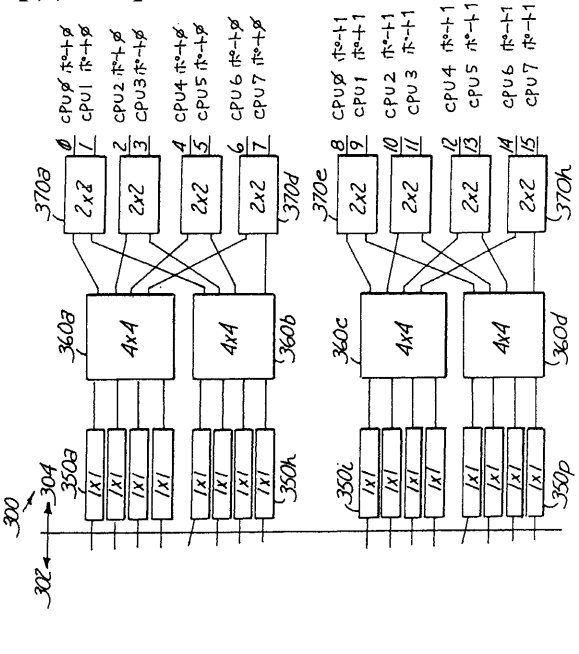
【図８Ｄ】



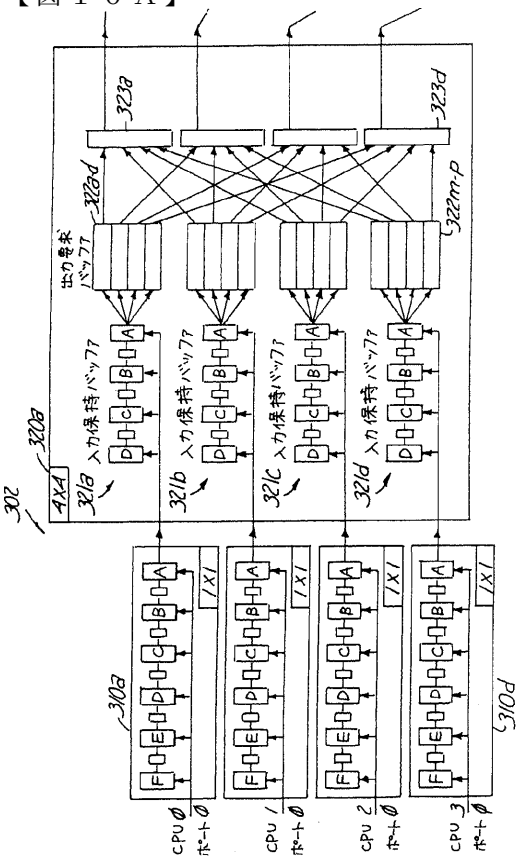
【図９Ａ】



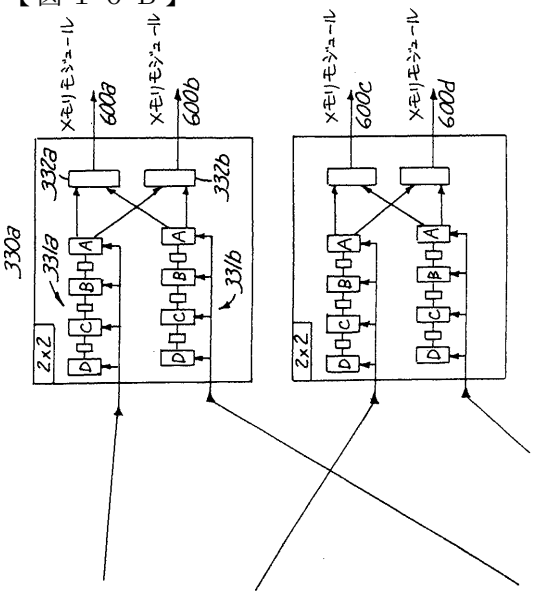
【図９Ｂ】



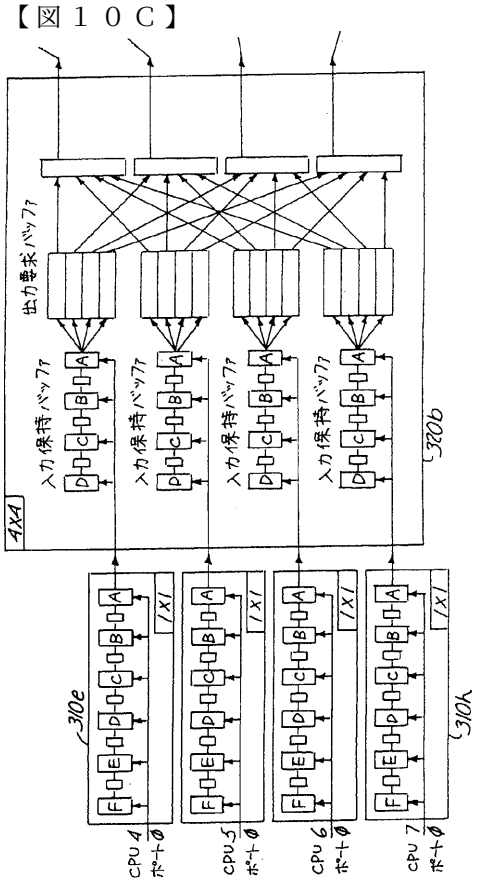
【図１０Ａ】



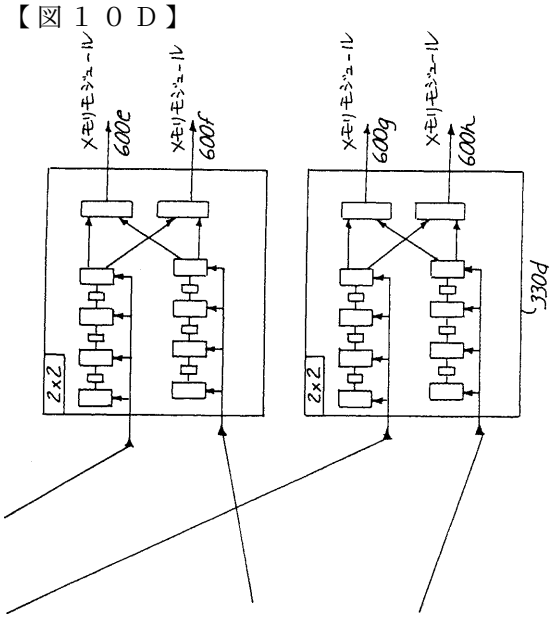
【図１０Ｂ】



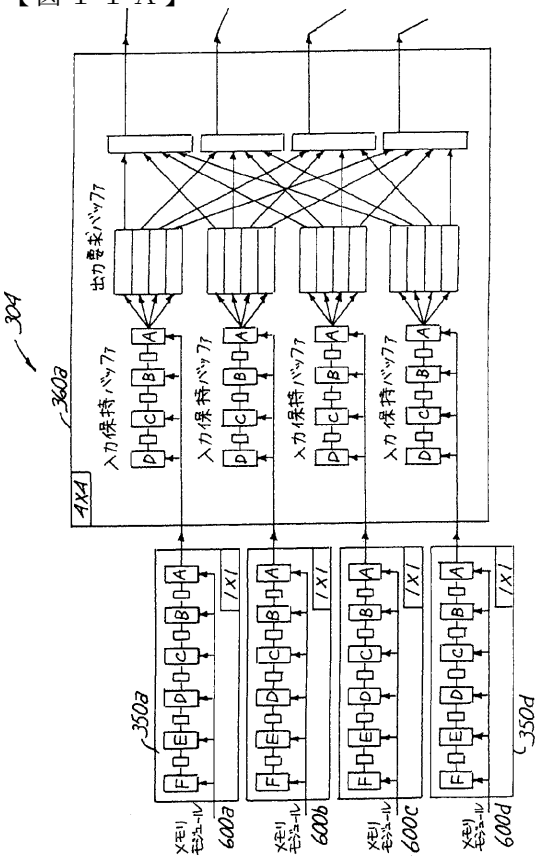
【図１０Ｃ】



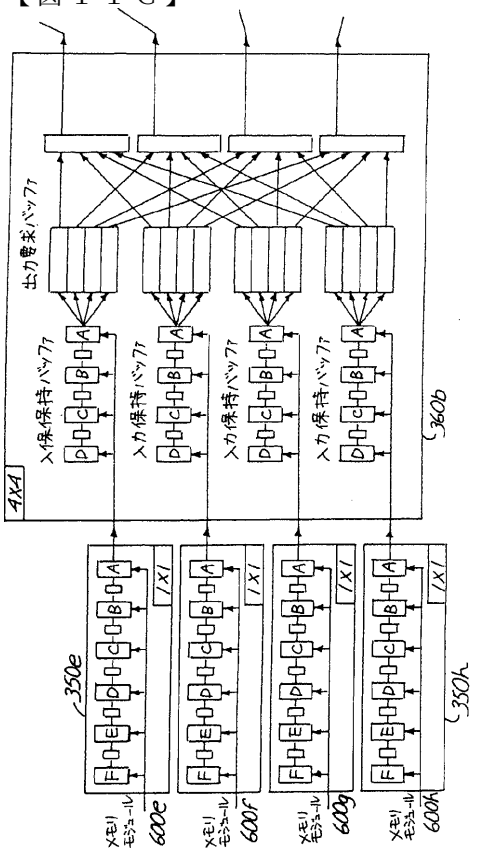
【図１０Ｄ】



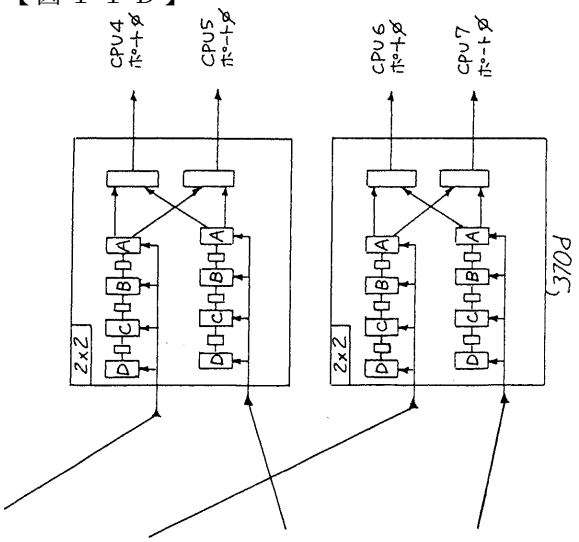
【図１１Ａ】



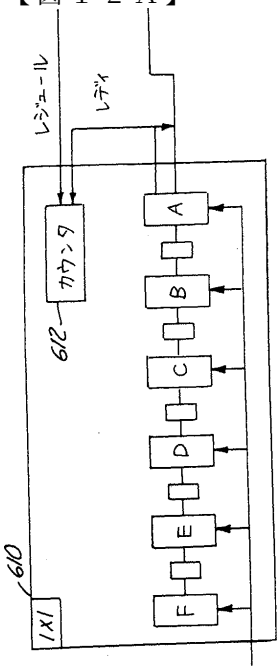
【図１１Ｃ】



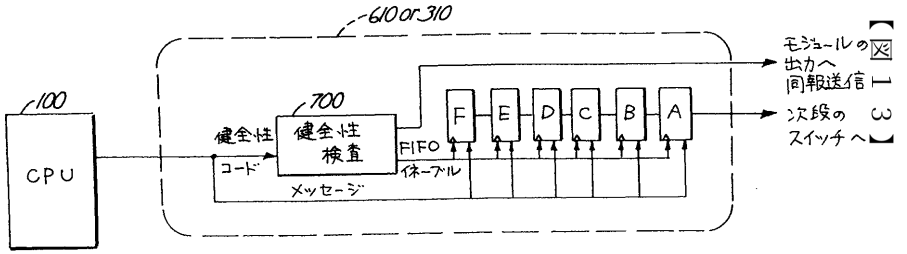
【図１１Ｄ】



【図１２Ａ】



【図１３】

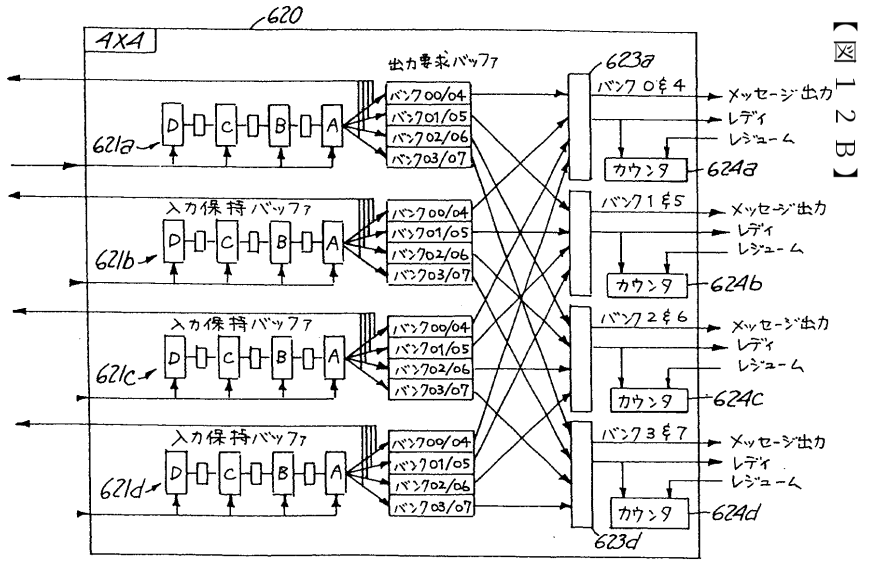


【図１４】

(--強制的に0とされるビット)

モジュール数			セクション構成			アドレスビットの割り付け			
プロセッサ	ネットワーク	メモリ	CPU当りのセクション数	セクション当りのサブセクション数	総バンク数	セクション	サブセクション	バンク	チップ・アドレス
1から4へ	0	4	8	1	64	00,01,02	--,--,--	03,04,05,--	06 - 31
1から4へ	0	4	8	1	32	00,01,02	--,--,--	03,04,--,--	05 - 30
1から4へ	0	4	4	1	32	00,01,--	--,--,--	02,03,04,--	05 - 30
1から4へ	0	4	4	1	16	00,01,--	--,--,--	02,03,--,--	04 - 29

【図１２Ｂ】

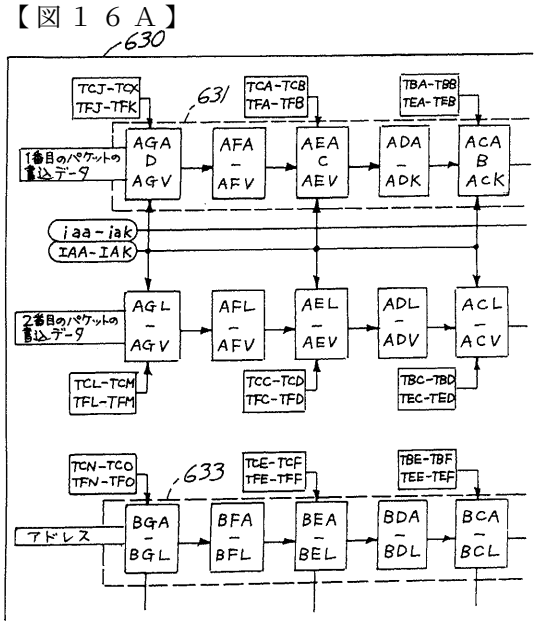


【図１５】

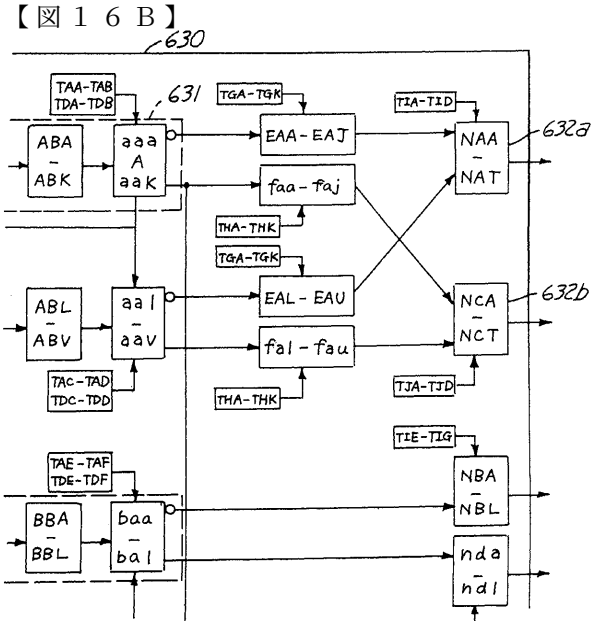
(--強制的に0とされるビット)

モジュール数			セクション構成			アドレスビットの割り付け			
プロセッサ	ネットワーク	メモリ	CPU当りのセクション数	セクション当りのサブセクション数	総バンク数	セクション	サブセクション	バンク	チップ・アドレス
8	16	8	8	2	128	00,01,02	03,--,--	04,05,06,--	07 - 32
8	16	8	8	2	64	00,01,02	03,--,--	04,05,--,--	06 - 31
8	16	8	4	2	64	00,01,--	02,--,--	03,04,05,--	06 - 31
8	16	8	4	2	32	00,01,--	02,--,--	03,04,--,--	05 - 30
8 16	16	16	8	4	256	00,01,02	03,04,--	05,06,07,--	08 - 33
8 16	16	16	8	4	128	00,01,02	03,04,--	05,06,--,--	07 - 32
8 16	16	16	4	4	128	00,01,--	02,03,--	04,05,06,--	07 - 32
8 16	16	16	4	4	64	00,01,--	02,03,--	04,05,--,--	06 - 31
8 32	16	32	8	8	512	00,01,02	03,04,05	06,07,08,--	09 - 34
8 32	16	32	8	8	256	00,01,02	03,04,05	06,07,--,--	08 - 33
8 32	16	32	4	8	256	00,01,--	02,03,04	05,06,07,--	08 - 33
8 32	16	32	4	8	128	00,01,--	02,03,04	05,06,--,--	07 - 32

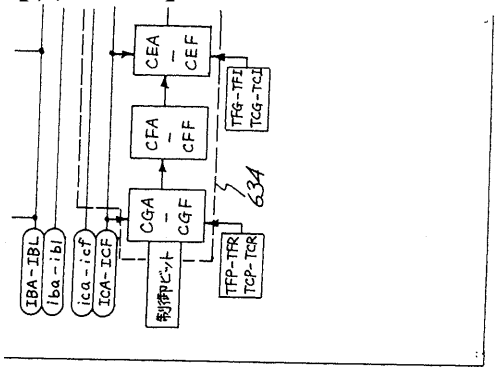
【図１６Ａ】



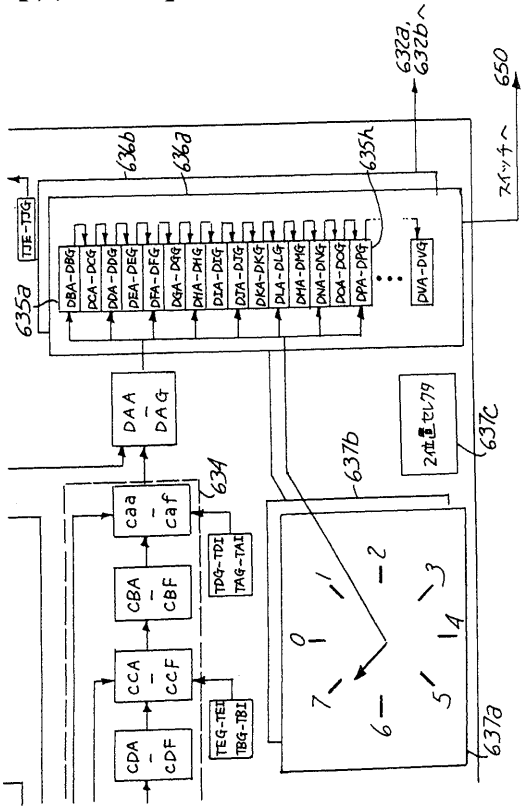
【図１６Ｂ】



【図１６Ｃ】



【図１６Ｄ】



フロントページの続き

(72)発明者　ロッギー、ピーター・ジー
アメリカ合衆国　５４７２９　ウィスコンシン州　チッペワ・フォールズ、ピー・オー・ボックス２１２、ルート３番

審査官　鳥居　稔

(56)参考文献　特開平０１－２９８４５９（ＪＰ，Ａ）
特開平０４－３２８６５３（ＪＰ，Ａ）
特表平０６－５０２０３３（ＪＰ，Ａ）
欧州特許出願公開第００３３９２７０（ＥＰ，Ａ１）
欧州特許出願公開第００５５３１５８（ＥＰ，Ａ１）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 15/173
H04Q
H04L