(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7386901号
(P7386901)

(45)発行日　令和5年11月27日(2023.11.27)

(24)登録日　令和5年11月16日(2023.11.16)

(51)国際特許分類
Ｇ０６Ｆ　17/16　(2006.01)

ＦＩ
Ｇ０６Ｆ　17/16　Ｂ

請求項の数　23　（全31頁）

(21)出願番号　特願2021-566236(P2021-566236)
(86)(22)出願日　令和2年5月1日(2020.5.1)
(65)公表番号　特表2022-531786(P2022-531786A)
(43)公表日　令和4年7月11日(2022.7.11)
(86)国際出願番号　PCT/IB2020/054137
(87)国際公開番号　WO2020/229933
(87)国際公開日　令和2年11月19日(2020.11.19)
審査請求日　令和4年10月21日(2022.10.21)
(31)優先権主張番号　16/408,575
(32)優先日　令和1年5月10日(2019.5.10)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
Ｎｅｗ　Ｏｒｃｈａｒｄ　Ｒｏａｄ，　Ａｒｍｏｎｋ，　Ｎｅｗ　Ｙｏｒｋ　１０５０４，　Ｕｎｉｔｅｄ　Ｓｔａｔｅｓ　ｏｆ　Ａｍｅｒｉｃａ
(74)代理人　100112690
弁理士　太佐 種一

最終頁に続く

(54)【発明の名称】　高性能なベクトル処理のためのアドレス生成

(57)【特許請求の範囲】
【請求項１】
バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内の前記バイナリ・データ・ベクトルにアクセスするための方法であって、前記方法が、
前記バイナリ・データ・ベクトルの開始アドレスおよび前記バイナリ・データ・ベクトルの要素の２の累乗のストライドｓを受信することと、
前記メモリ・バンクの各々について１つのｎ個のオフセットを決定することであって、前記オフセットの各々が、
複数のビットレベルのＸＯＲ関数を前記開始アドレスに適用してＺベクトルを生成することと、
マッピング・テーブルにアクセスするために前記Ｚベクトルを使用することと、
前記バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることとによって決定される、前記決定することと、
前記Ｚベクトルの２進等価値に応じて前記ｎ個のメモリ・バンク内の前記バイナリ・データ・ベクトルの部分のシーケンスを決定することと、
前記メモリ・ユニットの前記ｎ個のメモリ・バンク内の前記バイナリ・データ・ベクトルに並列にアクセスすることとを含む、方法。
【請求項２】
複数のビットレベルのＸＯＲ関数を前記開始アドレスに前記適用することが、

前記開始アドレスからｌｏｇ$_2$（ｎ）個の部分およびｌｏｇ$_2$（ｎ）ビット・サイズのＹの部分を選択することと、
前記ｌｏｇ$_2$（ｎ）個の部分の各々についてパリティ・ビットを決定し、ｌｏｇ$_2$（ｎ）次元のパリティ・ベクトルＸを生成することと、
ＸとＹの間でビットレベルのＸＯＲ演算を実行し、Ｚベクトルを生成することとを含む、請求項１に記載の方法。
【請求項３】
マッピング・テーブルにアクセスするために前記Ｚベクトルを使用することが、
前記マッピング・テーブル内のインデックスとして前記Ｚベクトルを使用することによって、ｎ＊ｎ個のエントリを含んでいる前記マッピング・テーブル内の行を選択することと、
前記マッピング・テーブルのアクセス結果を前記開始アドレスの部分と結合して、前記ｎ個のオフセットを取得することとも含む、請求項２に記載の方法。
【請求項４】
前記結合することが、
前記開始アドレスのビット・オフセットｌｏｇ２（ｎ）で開始するｌｏｇ２（ｎ）ビットを、前記決定されたｌｏｇ$_2$（ｎ）個のマッピング・テーブル・エントリに置き換え、ｎ個のオフセットを生成することを含み、各オフセットが、前記ｎ個のバンクの各１つにおいてオフセットとして使用される、請求項３に記載の方法。
【請求項５】
前記ｎ個のメモリ・バンク内の前記バイナリ・データ・ベクトルの前記部分のシーケンスを前記決定することが、
ｌｏｇ$_2$（ｎ）個の連続するマルチプレクサ段を制御することを含み、前記マルチプレクサ段の各々が、前記バイナリ・データ・ベクトルの全幅をカバーするように、複数の２入力ビット・マルチプレクサを含む、請求項１に記載の方法。
【請求項６】
前記バイナリ・データ・ベクトルの前記アクセスが読み取り動作である、請求項１に記載の方法。
【請求項７】
前記バイナリ・データ・ベクトルの前記アクセスが書き込み動作である、請求項１に記載の方法。
【請求項８】
前記開始アドレスの前記選択されたｌｏｇ$_2$（ｎ）個の部分が同じサイズに設定される、請求項２に記載の方法。
【請求項９】
前記バイナリ・データ・ベクトルが、前記複数の前記ｎ個のメモリ・バンク内の同じサイズの部分に格納される、請求項１に記載の方法。
【請求項１０】
前記開始アドレスの前記提供、前記オフセットの前記決定、ならびに前記バイナリ・データ・ベクトルの部分のシーケンスの前記決定、および部分のシーケンスの前記決定が、前記ｎ個のメモリ・バンクへの前記バイナリ・データ・ベクトルの前記書き込み動作の前に実行される、請求項７に記載の方法。
【請求項１１】
前記開始アドレスの前記提供、前記オフセットの前記決定、および前記バイナリ・データ・ベクトルの部分のシーケンスの前記決定が、前記ｎ個のメモリ・バンクに対する前記バイナリ・データ・ベクトルの前記読み取り動作の前に実行され、部分のシーケンスの前記決定が、前記ｎ個のメモリ・バンクに対する前記バイナリ・データ・ベクトルの前記読み取り動作の後に実行される、請求項６に記載の方法。
【請求項１２】
バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んで

いるメモリ・ユニット内の前記バイナリ・データ・ベクトルにアクセスするためのメモリ・アクセス・ユニットであって、前記メモリ・アクセス・ユニットが、
前記バイナリ・データ・ベクトルの開始アドレスおよび前記バイナリ・データ・ベクトルの要素の２の累乗のストライドｓを受信するように適応された受信ユニットと、
前記メモリ・バンクの各々について１つのｎ個のオフセットを決定するように適応された第１の決定モジュールであって、前記オフセットの各々が、
複数のビットレベルのＸＯＲ関数を前記開始アドレスに適用してＺベクトルを生成することと、
マッピング・テーブルにアクセスするために前記Ｚベクトルを使用することと、
前記バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることとによって決定される、前記第１の決定モジュールと、
前記Ｚベクトルの２進等価値に応じて前記ｎ個のメモリ・バンク内の前記バイナリ・データ・ベクトルの部分のシーケンスを決定するように適応された第２の決定モジュールと、
前記メモリ・ユニットの前記ｎ個のメモリ・バンク内の前記バイナリ・データ・ベクトルに並列にアクセスするように適応された基本アクセス・ユニットとを備える、メモリ・アクセス・ユニット。
【請求項１３】
前記第１の決定モジュールによって、複数のビットレベルのＸＯＲ関数を前記開始アドレスに前記適用することが、
前記開始アドレスから$\log_2$（ｎ）個の部分および$\log_2$（ｎ）ビット・サイズのＹの部分を選択することと、
前記$\log_2$（ｎ）個の部分の各々についてパリティ・ビットを決定し、$\log_2$（ｎ）次元のパリティ・ベクトルＸを生成することと、
ＸとＹの間でビットレベルのＸＯＲ演算を実行し、Ｚベクトルを生成することとを含む、請求項１２に記載のメモリ・アクセス・ユニット。
【請求項１４】
前記第１の決定モジュールによって、マッピング・テーブルにアクセスするために前記Ｚベクトルを使用することが、
前記マッピング・テーブル内のインデックスとして前記Ｚベクトルを使用することによって、ｎ＊ｎ個のエントリを含んでいる前記マッピング・テーブル内の行を選択することと、
前記マッピング・テーブルのアクセス結果を前記開始アドレスの部分と結合して、前記ｎ個のオフセットを取得することとも含む、請求項１３に記載のメモリ・アクセス・ユニット。
【請求項１５】
前記第１の決定モジュールの前記結合が、
前記開始アドレスのビット・オフセット$\log_2$（ｎ）で開始する$\log_2$（ｎ）ビットを、前記決定された$\log_2$（ｎ）個のマッピング・テーブル・エントリに置き換え、ｎ個のオフセットを生成することも含み、各オフセットが、前記ｎ個のバンクの各１つにおいてオフセットとして使用される、請求項１４に記載のメモリ・アクセス・ユニット。
【請求項１６】
前記第２の決定モジュールが、
前記ｎ個のメモリ・バンク内の前記バイナリ・データ・ベクトルの前記部分のシーケンスを前記決定するときに、
$\log_2$（ｎ）個の連続するマルチプレクサ段を制御するようにも適応され、前記マルチプレクサ段の各々が、前記バイナリ・データ・ベクトルの全幅をカバーするように、複数の２入力ビット・マルチプレクサを含む、請求項１２に記載のメモリ・アクセス・ユニット。
【請求項１７】
前記バイナリ・データ・ベクトルの前記アクセスが読み取り動作である、請求項１２に

記載のメモリ・アクセス・ユニット。
【請求項１８】
前記バイナリ・データ・ベクトルの前記アクセスが書き込み動作である、請求項１２に記載のメモリ・アクセス・ユニット。
【請求項１９】
前記開始アドレスの前記選択された$\log_2(n)$個の部分が同じサイズに設定される、請求項１３に記載のメモリ・アクセス・ユニット。
【請求項２０】
前記バイナリ・データ・ベクトルが、前記複数の前記ｎ個のメモリ・バンク内の同じサイズの部分に格納される、請求項１２に記載のメモリ・アクセス・ユニット。
【請求項２１】
前記開始アドレスの前記提供、前記オフセットの前記決定、ならびに前記バイナリ・データ・ベクトルの部分のシーケンスの前記決定、および部分のシーケンスの前記決定が、前記ｎ個のメモリ・バンクへの前記バイナリ・データ・ベクトルの前記書き込み動作の前に実行される、請求項１８に記載のメモリ・アクセス・ユニット。
【請求項２２】
前記開始アドレスの前記提供、前記オフセットの前記決定、および前記バイナリ・データ・ベクトルの部分のシーケンスの前記決定が、前記ｎ個のメモリ・バンクに対する前記バイナリ・データ・ベクトルの前記読み取り動作の前に実行され、部分のシーケンスの前記決定が、前記ｎ個のメモリ・バンクに対する前記バイナリ・データ・ベクトルの前記読み取り動作の後に実行される、請求項１７に記載のメモリ・アクセス・ユニット。
【請求項２３】
コンピュータ・プログラムであって、請求項１ないし１１のいずれか１項に記載の方法の各ステップをコンピュータに実行させるための、コンピュータ・プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本明細書に記載された実施形態例は、一般に、効率的なメモリ・アクセスに関連しており、より詳細には、バイナリ・データ・ベクトルが一部に格納される複数のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための方法に関連している。実施形態例は、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための関連するメモリ・アクセス・ユニットと、コンピュータ・プログラム製品とに、さらに関連する。
【背景技術】
【０００２】
多くのワークロードおよび最新のコンピューティングは、同一の動作を適用する基本的な処理ステップに依存することに基づき、大量のデータに対する一連の動作である。そのようなワークロードで高性能を実現するための主な課題は、通常、最も効率的な方法でメモリ・システムから実行ユニットにデータを取得すること、および処理結果を実行ユニットからメモリ・システムに返すことに関連している。
【０００３】
そのようなワークロードの多くによって必要とされる処理は、多くの場合、浮動小数点値または整数値のいずれかに対する乗算および加算の組み合わせである。例えば、深層学習の中心的処理（トレーニングおよび干渉の両方）のワークロードは、膨大な量の行列乗算動作を入力データまたはトレーニング・データあるいはその両方に適用することに基づく。ステンシル・コード（stencil codes）は、気象予測、石油およびガスの捜索などのアプリケーションに使用される多くのモデルの中核部を形成する。これらのステンシル・コードは、隣接する要素の値の何らかの加重和を通常は含む固定パターン（ステンシルと呼ばれる）に従って、誤差率要素（error rate elements）を反復的に更新する。ＦＦＴ

（Fast Fourier Transformation：高速フーリエ変換）は、科学および工学において最も広く使用されているアルゴリズムの１つであり、非常に多くのいわゆるバタフライ演算を計算することに基づき、各バタフライ演算は、乗算および加算に基づく。ステンシル計算およびＦＦＴの両方は、特定の深層学習アルゴリズム（例えば、畳み込みニューラル・ネットワーク（ＣＮＮ：convolutional neural networks）にも使用される。

【０００４】

高性能を獲得するために、実行ユニット（通常は、乗算器および加算器の組み合わせを含む）は、多くの場合、広いデータ・ベクトルのストリームとして提供される入力データに対して並列に、同じ動作または動作の組み合わせを何度も適用するベクトル（ＳＩＭＤ、単一の命令、複数のデータ）ユニットとして編成される。

【０００５】

ベクトル・ユニットを構成する処理要素のためのすべての必要なオペランド値を含んでいるメモリ・システムからデータ・ベクトルを取り出すことは、多くの場合、それらのオペランド値を読み取るために必要なアドレス・シーケンス内のいわゆる２の累乗のストライドのため、簡単ではない。この種類のアクセス・パターンは、例えば、１のインクリメント（例えば、０、１、２、３など）、８のインクリメント（例えば、０、８、１６、２４など）、または２５６のインクリメント（例えば、０、２５６、５１２、７６８、１０２４など）などを伴うアドレスによって、線形仮想アドレス空間または物理アドレス空間上にマッピングされたデータ構造にアクセスするワークロードに発生する。

【０００６】

バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための方法に関連する複数の開示が存在する。

【０００７】

文書ＵＳ　９７０５５３２　Ｂ２は、低密度パリティチェック・コードの生成の一部としての情報ビットの並列累算のための方法およびシステムを開示する。それによって、ベクトル演算を介して、連続する情報ビットが累算されることができ、累算に使用されるパリティ・アドレスは、仮想プライベート・パリティ・アドレス・マッピング（virtual and private parity address mapping）を介して連続的にされ得る。

【０００８】

文書ＵＳ　７７９３０３８　Ｂ２は、メモリ・バンク識別子およびメモリ・バンクの内部アドレスを含んでいる組み合わせアドレスへの、メモリ・システム内のメモリ・アクセスのためのアドレスのマッピングを開示する。このアドレスは、第１の部分および第２の部分に分割される。さらに、メモリ・バンク識別子は、検索行列内で検索動作を実行することによって決定される。

【０００９】

しかし、多くのアプリケーションでは、データ・ベクトル要素にアクセスするために使用されるアドレス・シーケンス内の複数の異なる２の累乗のストライドを含むさまざまな方法でデータ・ベクトルを構築する必要があるという、既知の解決策の不利益が引き続き問題になる。例えば、あるときには、データ・ベクトルは、ストライド１を伴うアドレス・シーケンスに関連するデータ要素で構成されることがあり、別のときには、データ・ベクトルは、ストライド８を伴うアドレス・シーケンスに関連するデータ要素で構成されることがある、などである。同じデータ要素が、さまざまな方法で他のデータ要素と組み合わせられてデータ・ベクトルを形成するため、１つのデータ・ベクトルを構成するすべてのデータ要素を、基礎になるハードウェアの１原理サイクル（principle cycle）でそれらのバンクから並列に読み取ることができるように、データ・ベクトルを形成するために使用されるすべてのデータ要素が個別のバンクに格納されることを保証することが、よりいっそう困難になる。本明細書において提案される概念は、この問題に対処する。

【発明の概要】

【００１０】

１つの態様によれば、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための方法が提供されてよい。この方法は、バイナリ・データ・ベクトルの開始アドレスおよびバイナリ・データ・ベクトルの要素の２の累乗のストライドｓを受信することと、メモリ・バンクの各々について１つのｎ個のオフセットを決定することとを含み、オフセットの各々は、複数のビットレベルのＸＯＲ関数を開始アドレスに適用してＺベクトルを生成し、マッピング・テーブルにアクセスするためにＺベクトルを使用し、バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることによって決定される。

【００１１】

さらに、この方法は、Ｚベクトルの２進等価値（binary equivalent value）に応じてｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定することと、メモリ・ユニットのｎ個のメモリ・バンク内のバイナリ・データ・ベクトルに並列にアクセスすることとを含む。

【００１２】

別の態様によれば、バイナリ・データ・ベクトルが一部に格納されてよい複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするためのメモリ・アクセス・ユニットが提供されてよい。メモリ・アクセス・ユニットは、バイナリ・データ・ベクトルの開始アドレスおよびバイナリ・データ・ベクトルの要素の２の累乗のストライドｓを受信するように適応された受信ユニットと、メモリ・バンクの各々について１つのｎ個のオフセットを決定するように適応された第１の決定モジュールとを備え、オフセットの各々は、複数のビットレベルのＸＯＲ関数を開始アドレスに適用してＺベクトルを生成し、マッピング・テーブルにアクセスするためにＺベクトルを使用し、バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることによって決定される。

【００１３】

さらに、メモリ・アクセス・ユニットは、Ｚベクトルの２進等価値に応じてｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定するように適応された第２の決定モジュールと、メモリ・ユニットのｎ個のメモリ・バンク内のバイナリ・データ・ベクトルに並列にアクセスするように適応された基本アクセス・ユニットとを備える。

【００１４】

バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための提案された方法は、次のような複数の利点および技術的効果提供することができる。

【００１５】

提案された方法は、特に、乗算などのような行列演算の場合に、「１回で」、すなわち、基本的にメモリ・システムの１動作サイクル内で、大きいデータ・ベクトルまたは長いデータ・ベクトルに対処することも、主として可能にすることができる。これは、重複するアドレス指定が必要とされなくてよいため、可能になることができる。高次元ベクトルの異なる次元のデータ値は、メモリ・システムまたはメモリ・ユニットの異なるメモリ・バンクにわたって常に分散されてよい。したがって、異なるメモリ・バンク、およびメモリ・バンク内の高次元ベクトルの１つの次元の１つの特定のデータ値をアドレス指定することによって、ベクトルのすべての値への最短のアクセス時間が可能になる。

【００１６】

したがって、同じ超次元ベクトルの異なる次元の異なるデータ値をアドレス指定するために、同じメモリ・バンクへの２回以上のアクセスが必要とされることがある状況が回避されることができる。これによって、メモリ・ユニット、およびしたがって、関連するコンピューティング・システムの動作サイクルを明確に節約することができる。したがって、本明細書で提案される概念によって、従来技術と比較した場合に、メモリ・システム／

ユニット内でデータ・ベクトルにアクセスするための性能上の大幅な優位性を実現することができる。

【００１７】

提案された概念は、メモリ・バンクに書き込むときにベクトルのデータ要素をシャッフルする方法、およびベクトルのデータを読み取るときにデータ要素を逆シャッフルする方法も、可能にする。したがって、書き込み－読み取りサイクルの後に、ベクトルのデータ要素は、メモリ・ユニット／システムへの書き込み動作の前のように、再び適切に格納される。

【００１８】

さらに、コンピュータ・システム設計における現在の傾向に起因して、メモリ内のデータへのアクセスおよび対応する転送が、多くのアプリケーションのシステムレベルの性能に影響を与える支配的要因になる。最先端のベクトル・ユニットによって極めて効率的に処理され得るデータ・ベクトルにメモリ内でアクセスできる速度を改善することによって、本明細書に記載された実施形態は、ある範囲のワークロードに対するコンピュータ・システムの性能を大幅に改善するのに役立つことができる。

【００１９】

以下では、本発明の概念の追加の実施形態が説明される。

【００２０】

この方法の１つの好ましい実施形態によれば、複数のビットレベルのＸＯＲ関数を開始アドレスに適用することは、開始アドレス（具体的には、ＣＰＵの線形アドレス空間内のベクトルの第１の要素のアドレス）から、log_2（ｎ）個の部分およびlog_2（ｎ）ビット・サイズのＹの部分を選択することと、log_2（ｎ）個の部分の各々についてパリティ・ビットを決定し、log_2（ｎ）次元のパリティ・ベクトルＸ（すなわち、個別のビットのグループ）を生成することと、ＸとＹの間でビットレベルのＸＯＲ演算を実行してＺベクトルを生成することとを含んでよい。それによって、アドレス・マスクによって（具体的には、アドレス・マスクによるlog_2（ｎ）＋１］によって）log_2（ｎ）個の部分およびＹの部分が選択可能であってよいということが述べられてよい。このシャッフル動作の詳細は、下で図を説明するときに、組み合わせにおいてさらに明確になるであろう。

【００２１】

方法の１つの有用な実施形態によれば、マッピング・テーブルにアクセスするためにＺベクトルを使用することは、マッピング・テーブル内のインデックスとしてＺベクトル（具体的には、その２進等価値）を使用することによって、log_2（ｎ）個のエントリを含んでいるマッピング・テーブル内の行を選択することと、マッピング・テーブルのアクセス結果を開始アドレスの部分と結合して（「マージする」と表すこともできる）ｎ個のオフセットを取得することとを含んでもよい。下で、図との関連において詳細が説明される。

【００２２】

方法の１つの有利な実施形態によれば、この結合することは、開始アドレスのビット・オフセットlog_2（ｓ）にあるlog_2（ｎ）ビットを、決定されたlog_2（ｎ）個のマッピング・テーブル・エントリに置き換え、log_2（ｎ）個のオフセットを生成することを含んでよい。それによって、各オフセットが、log_2（ｎ）個のバンクの各１つにおいてオフセットとして使用されてよい。これによって、極めて限定されたオーバーヘッドのみを伴う簡単な決定を可能にすることができる。

【００２３】

方法の１つのさらに好ましい実施形態によれば、ｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定すること（シャッフルとして表されてもよいプロセス）は、log_2（ｎ）個の連続するマルチプレクサ段を制御することを含んでよく、マルチプレクサ段の各々は、データ・ベクトルの全幅をカバーするように、複数の２入力ビット・マルチプレクサ（2-input-bit-multiplexers）を含んでいる。これによって

、重複するアドレスが必要とされなくてよいように、高次元ベクトルの個別のデータ値の位置決めを可能にすることができる。

【００２４】

より詳細には、対応するＺベクトルのビットが１に等しい場合に、第１のマルチプレクサ段が、データ・ベクトルの最上位部分を、最下位部分と交換してよい。第２のマルチプレクサ段が、第２のＺベクトルのビットの制御下で、データ・ベクトルの下位部分および上位部分に対して、同じ機能を並列に適用する、などであってよい。これらのｌｏｇ$_2$（ｎ）個のマルチプレクサ段は、メモリ・バンクの正しい位置に書き込まれ、メモリ・バンクの正しい位置からそれぞれ読み取られる、さまざまなデータ要素を交換するために使用されてよい。

【００２５】

方法の１つの許容される実施形態によれば、アクセス動作が読み取り動作であってよい。代替として、または組み合わせて、アクセス動作が書き込み動作であってよい。したがって、本発明の概念の一部として、削除操作も使用可能である。したがって、個別のメモリ・バンクのメモリ・ユニット、メモリ・デバイス内のすべての動作が、本明細書で提案される概念を使用して実行されてよい。より大きい（例えば、超次元）ベクトルの識別されたデータ要素のアドレス生成または再シャッフルのシーケンスには、差異が存在してよい。

【００２６】

方法の１つの実用的な実施形態によれば、開始アドレスの選択されたｌｏｇ$_2$（ｎ）個の部分は、同じサイズに設定されてよい。これは、２の累乗のストライドの概念の下で同様に実行されてよい。

【００２７】

方法の別の実用的な実施形態によれば、バイナリ・データ・ベクトルが、複数のｎ個のメモリ・バンク内の同じサイズの部分に格納されてよい。したがって、データ要素の等しい分散が結果として実現されてよい。

【００２８】

メモリ・バンクへの書き込み動作が対象にされる方法の１つの実施形態によれば、開始アドレスを提供することと、オフセットを決定することと、バイナリ・データ・ベクトルの部分のシーケンスを決定することと（すなわち、主概念の部分）、部分のシーケンスを決定することと（すなわち、シャッフル）が、ｎ個のメモリ・バンクへのバイナリ・データ・ベクトルの書き込み動作の前に実行されてよい。これによって、メモリ・ユニットの１サイクルのみを基本的に必要とする高速書き込み動作のため、およびデータに対する「ミラーリングされた」読み取り動作を可能にするために、ベクトルの次元値を格納することを保証することができる。

【００２９】

方法の１つの結果とし生じる実施形態によれば、開始アドレスを提供することと、オフセットを決定することと、バイナリ・データ・ベクトルの部分のシーケンスを決定することと（主概念を参照）は、ｎ個のメモリ・バンクに対するバイナリ・データ・ベクトルの読み取り動作の前に実行されてよく、部分のシーケンスを決定すること（すなわち、シャッフル）は、ｎ個のメモリ・バンクに対するバイナリ・データ・ベクトルの読み取り動作の後に実行されてよい。このステップは、大きい次元のベクトルまたは超次元ベクトルのデータ値を元のシーケンスに戻すために必要になることがある。

【００３０】

さらに、実施形態は、コンピュータまたは任意の命令実行システムによって、またはこれらに接続して使用するためのプログラム・コードを提供するコンピュータ使用可能媒体またはコンピュータ可読媒体からアクセスできる関連するコンピュータ・プログラム製品の形態を取ってよい。この説明の目的で、コンピュータ使用可能媒体またはコンピュータ可読媒体は、命令実行システム、命令実行装置、または命令実行デバイスによって、またはこれらに接続して使用するためのプログラムを格納するか、伝達するか、伝搬するか、

または運ぶための手段を含むことができる任意の装置であってよい。

【００３１】

本発明の実施形態が、さまざまな対象を参照して説明されるということに注意する必要がある。具体的には、一部の実施形態は、方法タイプの請求項を参照して説明され、他の実施形態は、装置タイプの請求項を参照して説明される。ただし、当業者は、前述の説明および以下の説明から、特に注記のない限り、対象の１つの種類に属している特徴の任意の組み合わせに加えて、異なる対象に関連する特徴間、具体的には、方法タイプの請求項の特徴と、装置タイプの請求項の特徴との間の任意の組み合わせも、本文書内で開示されると見なされるということを推測するであろう。

【００３２】

上で定義された態様およびその他の態様は、以下で説明される実施形態の例から明らかになり、実施形態の例を参照して説明されるが、実施形態はこれらに限定されない。

【００３３】

以下の図面を単なる例として参照し、好ましい実施形態について説明する。

【図面の簡単な説明】

【００３４】

【図１】バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための本発明の方法の基本実施形態のブロック図を示す図である。

【図２】標準的な行列変更のために必要な困難なメモリ・アドレス指定方式を示す図である。

【図３】提案された概念の別の応用分野を示す図である。

【図４】例示的なマッピングの実施形態を示す図である。

【図５】９ビットの要素を含んでいるデータ・ベクトルの開始アドレスの例を示す図である。

【図６】例示的なマッピング・テーブルおよびメモリ・ユニット内のメモリ・バンクのオフセットを決定することにおける次のステップの実施形態を示す図である。

【図７】メモリ・バンク内のエントリのシャッフルの決定方法を示す図である。

【図８】メモリ・ユニットを表す例示的な８つのバンクを示す図である。

【図９】ベクトル開始アドレス２５６、ストライド１としての例を示す図である。

【図１０】ベクトル開始アドレス３、ストライド８としての例を示す図である。

【図１１】提案された基本概念に従って、より実装に近い読み取りプロセスのフローチャートを示す図である。

【図１２】提案された基本概念に従って、より実装に近い読み取りプロセスのフローチャートを示す図である。

【図１３】複数のバンクを含んでいるメモリ・ユニット内のバイナリ・ベクトルのためのメモリ・アクセス・ユニットのブロック図を示す図である。

【図１４】メモリ・アクセス・ユニットが使用され得るコンピュータ・システムのブロック図を示す図である。

【発明を実施するための形態】

【００３５】

本説明の文脈において、以下の規則、用語、または表現、あるいはその組み合わが使用されてよい。

【００３６】

「バイナリ・データ・ベクトル」という用語は、コンピュータのメモリ・システム内のビットのシーケンスのワードとしてアドレス指定される値のフィールド（例えば、ビット値、整数値、浮動小数点値、文字フィールドなど）を示してよい。そのようなメモリ・システムは、通常、ＲＡＭ（random access memory：ランダム・アクセス・メモリ）であってよい。特定のアドレスでのワード長に応じて、事前に定義された数のビットがメモリ・システムまたはメモリ・ユニット（テキスト全体を通じて、両方の形態が同義語とし

て使用されてよい）のワードに格納されてよい。

【００３７】

「メモリ・ユニット」という用語は、複数のバンクに分割されることがあるコンピュータ・システムのより大きいストレージ・モジュールを示してよい。各バンクは、特定のサイズの個別にアドレス指定可能なワードに編成された、事前に定義された数のストレージ・セルを含んでよい。メモリ・ユニットのバンクのメモリ・バンクは、個別にアドレス指定可能であってよく、各メモリ・バンク内で、個別のワードがアドレス指定可能であってよい。

【００３８】

「メモリ・バンク」という用語は、ストレージ・セルのグループを示してよく、この用語の複数形は、バンク・アドレスを介してアクセスできるストレージ・セルのより大きいグループを定義する。各バンク内で、ワードとして編成された複数の個別のメモリ・セルが、個別にアドレス指定されてよい。通常、メモリ・ユニットは、複数のメモリ・バンクを含む。

【００３９】

「バイナリ・データ・ベクトルの開始アドレス」という用語は、通常、長いバイナリ・データ・ベクトルの第１の（または第０の）要素のアドレスを示してよい。

【００４０】

「２の累乗のストライド」という用語は、2^n個の要素によって間を分離されている要素のシーケンスを示してよい。ｎ＝０である場合、$2^0=1$であるため、要素０、１、２、３、．．．がアドレス指定されてよい。ｎ＝１である場合、$2^n=1$であるため、要素０、２、４、６、．．．がアドレス指定されてよい、などとなる。

【００４１】

「Ｚベクトル」という用語は、複数のビットレベルのＸＯＲ関数をアドレスに適用することによって取得された$\log_2(n)$ビットのサイズを有するベクトルを示してよい。

【００４２】

「マッピング・テーブル」という用語は、メモリ・ユニットに含まれているバンクの数と同じ数の列を含んでいるテーブルを示してよい。テーブル内の要素は、０からバンクの数－１までの整数値であってよい。

【００４３】

以下では、各図について詳細に説明する。図内のすべての命令は概略図である。まず、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための本発明の方法の実施形態のブロック図が示される。その後、さらに別の実施形態に加えて、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするためのメモリ・アクセス・ユニットの実施形態が説明される。

【００４４】

図１は、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための方法の実施形態のブロック図を示している。以下の例の精神的参照モデルとして、バイナリ・データ・ベクトルが２５６ビットの長さであること、およびメモリ・ユニットが８つのバンクを含むこと、各バンクの３２ビットの幅、ならびに６４個の位置のバンク・サイズを仮定することができる。

【００４５】

この方法は、バイナリ・データ・ベクトルの開始アドレス（具体的には、ＣＰＵのアドレス空間内の配列の開始アドレス）およびバイナリ・データ・ベクトルの要素の２の累乗のストライドｓを受信すること（１０２）を含む。

【００４６】

この方法は、メモリ・バンクの各々について１つのｎ個のオフセットを決定すること（

１０４）も含む。それによって、複数のビットレベルのＸＯＲ関数を開始アドレスに適用してＺベクトルを生成することと、マッピング・テーブルにアクセスするためにＺベクトルを使用することと（１０８）、バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることと（１１０）によって、オフセットの各々が決定される（１０６）。

【００４７】

さらに、この方法は、Ｚベクトルの２進等価値に応じてｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定することと（１１２）、メモリ・ユニットのｎ個のメモリ・バンク内のバイナリ・データ・ベクトルに並列に（具体的には、１動作サイクル内で、または同時に）アクセスすることと（１１４）を含む。

【００４８】

図２は、２００に、標準的な行列演算のために必要な困難なメモリ・アドレス指定方式を示している。この図は、単純な行列乗算の例を示しており、行列Ａの行が、要素ごとに行列Ｂの列に適用される。両方の行列の要素が、線形アドレス空間内の連続するアドレス位置に書き込まれる場合、行列Ａ内の各行は、アドレス・ストライド１を使用してアクセスされ、行列Ｂ内の各列は、行のサイズに等しいアドレス・ストライドを使用してアクセスされる。多くのメモリ・システムでは、後者のストライドが問題になる可能性があるため、通常、「行列の対角線上」の行列値を反転することを伴う、いわゆる行列転置演算が適用される。その結果、次に、ストライド１のアクセス・パターンを使用して列にアクセスし、結果として生じる行列Ｃを生成することができる。しかし、これは、行列全体を読み取って再び書き込む必要があるという犠牲を伴い、大量のメモリ帯域幅および時間を消費する可能性もある。

【００４９】

提案された概念の別の応用分野を示す別の例が、図３の３００に示されている。図３は、２Ｄ構造または３Ｄ構造に適用できるさまざまなステンシルを示している。この図から、単一のステンシルによってカバーされたすべての要素にアクセスすることが、（２Ｄ構造および３Ｄ構造の場合に）それぞれ２つ、３つの異なる２の累乗のストライドを含んでいるということがわかる。

【００５０】

長い間、２の累乗のアクセス・パターンが調査されたにもかかわらず、非効率的アクセスの問題はまだ解決されていない。１つの問題は、多くの場合、２の累乗の数のメモリ・バンクを使用して編成された多くのメモリ・システムにおいて使用される編成および卓越したアドレス・マッピングに起因して、このような種類のアクセスが、メモリ性能の低下をもたらすことが多いということである。

【００５１】

特殊なＨＰＣ（high performance computing：高性能コンピューティング）システムでは、素数に対してモジュロ演算を実行すること、または同様の方法に基づく固定された／配線接続されたマッピングが適用されることがある。これらのマッピングは、異なっているか、またはプログラムすることが不可能であるため、特定のワークロードに固有であることがあり、したがって、いまだに悪い状況／異常な状況にある可能性がある。後者のメカニズムも、アクセス待ち時間を増加させる。

【００５２】

次の図に進む前に、ベクトルの処理およびアクセスに関する次の考察について検討するべきである。

【００５３】

ｎ個のバンク（すなわち、ｎ＊ｗに等しいデータ・ベクトルのサイズであり、ｗは単一のバンクの幅またはアクセス粒度である）内のデータ・ベクトルに並列にアクセスするために、バンクごとに正しいオフセットが生成され、データ・ベクトルごとに合計ｎ個のオフセットを得る必要がある。例えば、図４のマッピング例４００に基づいてアドレス・シーケンス０、１、２、３、４、５、６、および７（ストライド１）に関連する８要素のデ

ータ・ベクトルにアクセスする必要がある場合、８つの各メモリ・バンク内で、次のオフセットでメモリ位置にアクセスする必要がある：｛０，　１，　２，　３，　４，　５，　６，　７｝。

【００５４】

示されているテーブルは、複数のメモリ・バンクから成るメモリ・システムに格納された１つのデータ構造への２の累乗のストライドのアクセス・パターン間で、完全にインターリーブすることができる、柔軟でプログラム可能なアドレス・マッピング方式を定義することによって、前述した問題に対処する。

【００５５】

このマッピング方式は、マッピングされるアドレスから抽出された２つの値によってインデックス付けされた小型の２Ｄルックアップ・テーブル（図示されていない）に基づく。２つのインデックス値のうちの１つは、選択されたアドレス・ビットのビット単位のパリティ値を計算することによって決定される。しかし、基礎になる概念についてここで説明する必要はない。この図は、完全なインターリーブの複数の２の累乗アクセス・パターンの場合、メモリ位置のシーケンスのシャッフルも必要になることがあるということのみを示しているはずである。

【００５６】

図４の縞模様でマーク付けされた「マッピングされたアドレス」は、２の累乗のストライドを伴うアライメントされたアドレス・シーケンスに関連する（例示的な８つのデータ要素で）メモリ・バンクの数に等しいサイズを有するデータ・ベクトルが、すべてのメモリ・バンク上で完全にインターリーブされており、したがって各バンク上で正確に１回のアクセスを必要として、すべてのバンクに並列にアクセスすることによって、単一のデータ・ベクトルとしてアクセスできる（すなわち、読み取るか、または書き込むことができる）ことを示している。例えば、図４では、ストライド１を伴うアライメントされたアドレス（例えば、アドレス０、１、２、３、４、５、６、７）、ストライド２を伴うアライメントされたアドレス（例えば、アドレス０、２、４、６、８、１０、１２、１４）、ストライド４を伴うアライメントされたアドレス（例えば、アドレス０、４、８、１２、１６、２０、２４、２８）、ストライド８を伴うアライメントされたアドレス（例えば、アドレス０、８、１６、２４、３２、４０、４８、５６）、ストライド１６を伴うアライメントされたアドレス（例えば、アドレス０、１６、３２、４８、６４、８０、９６、１１２）、ストライド３２を伴うアライメントされたアドレス（例えば、アドレス０、３２、６４、９６、１２８、１６０、１９２、２２４）などの各ブロックに、すべて並列にアクセスすることができる。アクセスされるデータ要素がバンク間で均等に分散されるため、バンクの再アクセスは不要である。

【００５７】

読み取り動作の場合、バンク０がデータ・ベクトルの第１の要素（マッピングされたアドレス０）を提供し、バンク１がデータ・ベクトルの第２の要素（マッピングされたアドレス１）を提供する、などとなる。同様に、書き込み動作の場合、データ・ベクトルの要素が、連続的順序で各バンクに書き込まれる。代わりに、アドレス・シーケンス０、２、４、６、８、１０、１２、および１４（ストライド２）に関連する８要素のデータ・ベクトルがアクセスされる場合、８つの各バンク内で、次のオフセットに並列にアクセスする必要がある：｛０，　８，　２，　１０，　４，　１２，　６，　１４｝。この例では、読み取り動作の場合、データ・ベクトルの第１の要素（マッピングされたアドレス０）がバンク０によって提供され、データ・ベクトルの第２の要素（マッピングされたアドレス２）がバンク２によって提供され、データ・ベクトルの第５の要素（マッピングされたアドレス８）がバンク１によって提供される、などとなる。マッピングされたアドレスの順序（０、２、．．．、１４）で要素を含んでいるべきである望ましいデータ・ベクトルを復元するために、バンクから読み取られたデータがシャッフルされなければならないということは明らかであろう。さらに、データ・ベクトルが書き込まれる場合、メモリ・バンク上で対応するアドレスがマッピングされる方法に従って各要素を正しく書き込むために、要

素もシャッフルされなければならない。

【００５８】

データ・ベクトルの要素に関連するアドレス・シーケンスに応じて、ある範囲の異なる順序が存在する可能性があり、その場合、個別のバンクのオフセットが生成される必要があり、読み取りデータおよび書き込みデータがシャッフルされる必要がある。

【００５９】

総当たりの方法は、他の要素とは無関係にデータ・ベクトル内の要素ごとにアドレスを生成し、バンク識別子およびバンク・オフセットに独立してマッピングする方法であり、合計ｎ個のアドレスの生成機能およびアドレス・マッピング機能をもたらす。次に、マッピング機能の出力（具体的には、バンク識別子）は、データ・ベクトルを書き込むためおよび読み取るための２つのｎ対ｎシャッフル・ユニットを制御し、任意の可能な順序での各データ・ベクトルの要素のシャッフルを可能にするために使用され得る。

【００６０】

しかし、性能および面積の両方のコストに関して、ｎの値と共に、この方法の複雑さが急激に増大し、総当たりの方法を、ｎの小さい値（例えば、４または８）に適用することに基本的に制限する。

【００６１】

しかし、より大きいデータ・ベクトルは、計算における必要な効率および性能を達成するのに役立つため、非常に重要である。このことは、より新しいメモリ技術（例えば、ＨＢＭ（high bandwidth memory：高帯域幅メモリ））によって提供される広いデータ・パスのために、よりいっそう強調される。

【００６２】

そのため、次の技術革新が対象とされる。

１．並列にすべてのバンクをカバーするデータ・ベクトル全体のベクトルに基づくオフセット生成。

２．メモリ・バンクに書き込むためのデータ・ベクトル要素の効率的なシャッフル。

３．メモリ・バンクから読み取った後のデータ・ベクトル要素の効率的なシャッフル。

４．パリティ・ビットの各々の計算に使用されるアドレス・ビットの選択。

【００６３】

ベクトルに基づくオフセット生成の場合、任意の２の累乗のストライドを伴うアライメントされたアドレス・シーケンスに関連するｎ個の要素を含む任意のデータ・ベクトルに関して、ｎ個のアドレスが、アドレスにおける$\log_2 n$個の隣接するビット位置に制限されることと、それらの$\log_2 n$個のビット位置のｎ個の可能な値をすべてカバーすることとの間の違いについて、以下の観察が行われ得る。

【００６４】

ビット・インターリーブ方式でアドレスからパリティ・ビットが抽出される場合、２の累乗のストライドを伴う任意のアライメントされたアドレス・シーケンスに含まれるｎ個のアドレスは、ルックアップ・テーブル内の単一の行で指定されたバンク識別子に基づいてマッピングされる。

【００６５】

図５は、９ビットの要素a_0、a_1、．．．．、a_8を含んでいるデータ・ベクトルの開始アドレス「ａ」５０２の例５００を示している。ＭＳＢ（most significant bits：最上位ビット）は、後で必要とされるベクトルＹとして使用され、$\log_2$（ｎ）のサイズ（ｎ＝８）である。ビットa_0、a_1、．．．、a_5は、マスク・ベクトルm_0、m_1、およびm_2（$\log_2$（ｎ）＝３）を使用してＡＮＤ結合され、図の右側の$\log_2$（ｎ）個のベクトル５０４を生成する。マスク・ベクトルm_0、m_1、およびm_2が、開始アドレス５０２のＹの部分の位置に「０」を含んでいることに注意する。

【００６６】

ベクトル５０４の各々について、パリティ・ビットが計算され、X_0、X_1、X_2を生成する。これは、ＸＯＲ関数を適用することとして解釈されてもよい。X_0、X_1、X_2は、

Ｘベクトルの次元として扱われる。次に、ビットレベルのＸＯＲ関数がＸおよびＹに適用され、同じサイズまたは次元（log_2（ｎ））のベクトルＺを生成する。

【００６７】

これに関して、図６は、６００で、メモリ・ユニット内のメモリ・バンクのオフセットを決定することにおける次のステップを示している。

【００６８】

マッピング・テーブルが、Ｘ値およびＹ値の特定の組み合わせに関連して格納されている値が、それらのＸ値およびＹ値のＸＯＲ積に等しいという特性を有するということに注意する。これは、やはり、バンク識別子ｉが特定の行の列ｊに発生した場合、バンク識別子ｊがその行の列ｉにも発生する、という特性をもたらす（それらの値は、交換されて示され得る）。

【００６９】

例えば、マッピング・テーブル６０２内のＹ＝６に対応する行が、バンク識別子７を（Ｘ＝１に対応する）列１に含んでいる。ここで、同じ行が、バンク識別子１を列７（Ｘ＝７）に含んでいる。これら２つの位置は、「交換」位置の対と呼ばれる。上記の「交換」特性のため、ｎ個のアドレスから抽出されたパリティ値が、ｎ個のアドレスが異なっている、log_2（ｎ）個の隣接するビット位置の値と同じ順序である場合、ここで示されるように、バンク・オフセットをルックアップ・テーブルの行から直接選択することができる。

【００７０】

ベクトルＺの２進等価値が、マッピング・テーブル６０２への行インデックスとして使用される。マッピング・テーブル６０２の行の各２進値が、開始アドレスのlog_2（ｎ）ビットを置き換える定義された位置で使用される。これが、図６の下側部分に示されている。アドレスにおいて（図５／５０２の部分ａ０、ａ１、．．．．、ａ５を比較する）、ビット・オフセット３で開始するlog_2（ｎ）ビットが、各行エントリの２進等価値に置き換えられ、例えば、図６の左下部分からわかるように、「３」が０１１（２進数）を与える。この原理が、マッピング・テーブルの選択された行内の各要素に適用される。理解可能にするために、最初の２つおよび８番目のアドレス／行エントリの組み合わせのみが示されているということに注意する。この時点で、個別のバンクのオフセット値が決定されている。

【００７１】

８ビットの例の場合、マッピング・テーブル（すなわち、ルックアップ・テーブル）が８＊８＊３ビット＝２４バイトのみを必要とするということにも注意する。そのような非常に小さいテーブル内のエントリの検索は、提案された方法が表す極めて大きい優位性にとって、無視できる程度の要件しか表さない。

【００７２】

次ステップでは（図７を比較する）、メモリ・バンク内のエントリのシャッフルを決定する必要がある。この決定は、Ｚベクトルの再呼び出しを必要とする。Ｚがバイナリ・ベクトルであるということに起因して、Ｚベクトルの要素は「０」または「１」のいずれかになる。したがって、ベクトルＺのビットによって制御される交換制御は、ビット＝１が開始アドレスでのエントリの対の交換を意味し、ビット＝０が「交換なし」を意味する、というように定義される。

【００７３】

これが、図７の下向き矢印（交換なし）または交差した矢印（交換）として示されており、これらの矢印は、図６に示されているように、ＺベクトルのＬＳＢ（least significant bit：最下位ビット）から次々に開始するlog_2（ｎ）ステップのシーケンスにおいて適用される。

【００７４】

次にこれを、図８のメモリ・ユニット８０２を表す８つのバンクの例８００において示すことができる。例えば、（ストライド１を伴う）アドレス・シーケンス６４、６５、６

６、６７、６８、６９、７０、および７１は、１に等しいＹ値を含み、したがって、マッピング・テーブル８０４内の行１（Ｙ＝１）に従ってマッピングされる。アドレス６４がバンク１（８０６を比較する）にマッピングされ（８０８を比較する）、アドレス６５がバンク０にマッピングされ、アドレス６６がバンク３にマッピングされ、アドレス６７がバンク２にマッピングされ、アドレス６８がバンク５にマッピングされ、アドレス６９がバンク４にマッピングされ、アドレス７０がバンク７にマッピングされ、アドレス７１がバンク６にマッピングされる。各オフセットは、６最下位アドレス・ビット０、１、２、３、４、５、６、および７に等しい。（前述したように、）これら８つのアドレスが異なっている$\log_2 8 = 3$つのビット位置は、アドレス・ビット位置０、１、および２にあり、これらのアドレス・ビット位置は、バンク・オフセットの３最下位ビットにも等しい。前の８つのオフセットの場合、これらのビット位置での値は、０、１、２、３、４、５、６、および７に等しい（これらは、この例では他のオフセット・ビットがたまたま０であるため、オフセット値に等しい）。ルックアップ・テーブルの内容が生成される方法のため、これらの異なる$\log_2$（ｎ）個のビット位置の値は、パリティ・ビットの値に等しくなる。

【００７５】

ルックアップ・テーブルの内容の上記の「交換」特性に基づいて、アドレスが、対応するルックアップ・テーブルの要素に基づいて特定のバンク識別子にマッピングされる場合、その要素に対応する交換位置が、異なる$\log_2$（ｎ）個のビット位置の値に等しい、その同じアドレスのパリティ・ビットの値を含むということが導き出され得る。これは、アライメントされたアドレス・シーケンスにおいて第１のアドレスのＹ値を決定することによって、オフセット・ベクトルを生成することができ、その後、異なる$\log_2$（ｎ）個のビット位置で、シーケンス内の第１のアドレスのＹ値に対応する各行の値を置換することによって、（前述したように、最下位部分を取得することによって）その第１のアドレスから取得されたオフセットのｎ個のコピーを取得することができるということを意味する。

【００７６】

この特性は、任意の２の累乗のストライドを伴うｎ個のアドレスの任意のアライメントされたシーケンスに当てはまる。これは、前述したように、ｎ個の異なるビット位置が、「アライメントされた」方法でＸフィールドのビットをカバーする（すなわち、ビット位置Ｘ０から「開始」し、その後に位置Ｘ１が続く、などとなる）、２の累乗のストライドに限定されない。これは、任意のｎ個の連続するビット位置の値が、常に、同じテーブルの行内の異なるエントリにインデックス付けする結果をもたらすからである。

【００７７】

注目すべき示唆は、アドレス「ａ」から導出されたＹ値によってインデックス付けされた行からの値を、２の累乗のストライドの$\log 2$の値に対応するｎ個のビット位置で、アドレスａから導出されたオフセット値に代入することによって、この特性に基づいて、ｎ個のバンク・オフセットの値が直接取り出され得るということである。これが、次のステップを適用することによって、すべてのバンクのオフセットを効率的に生成するためのステップを示している、図１１のフローチャートに示されている。（１）アドレス・シーケンスの第１のアドレスのみをマッピングし（すなわち、パリティを計算し、Ｙ値を導出し、オフセットを導出し）、（２）Ｙに対応する１つのテーブルの検索を実行し、（３）並列なシフトおよびビット単位のＯＲ関数を実行して、実際のバンク・オフセットを取得する。

【００７８】

フローチャートを参照する前に、マッピングの２つの追加の例が与えられてよい。図９は、ベクトル開始アドレス２５６、ストライド１（アドレス２５６、２５７、２５８、．．．、２６３）としての例９００を示している。２５６が１０００００００００ｂに等しいため、マッピング・テーブル（図６の６０２を比較する）内の行４（１００ｂ）が４　５　６　７　０　１　２　３に等しくなるように、Ｙベクトル、Ｘベクトル、およびＺベクトル

が、Ｙ＝１００ｂ、Ｘ＝０００ｂ、Ｚ＝１００ｂであると決定され得る。したがって、（それぞれ、バンク０～７の）バンク・オフセットは、４、５、６、７、０、１、２、３である。Ｚ＝１００ｂを使用するデータ・ベクトルのシャッフルは、第１の４つのエントリが第２の４つのエントリと交換されるということを意味する。

【００７９】

図１０は、ベクトル開始アドレス３、ストライド８（したがって、アドレス３、１１、１９、．．．、５９）としての例１０００を示している。開始アドレス＝３＝０００００００１１ｂであるため、Ｙ＝０００ｂ、Ｘ＝０１１ｂ、Ｚ＝０１１ｂを得る。マッピング・テーブル６０２内の行３（０１１ｂ）は、３　２　１　０　７　６　５　４に等しい。したがって、（それぞれ、バンク０～７の）バンク・オフセットは、２７（０１１０１１ｂ）、１９（０１００１１ｂ）、１１、３、５９、５１、４３、３５として決定され得る。

【００８０】

データ・ベクトルのシャッフル部分は、次によって決定される。Ｚ＝０１１ｂは、隣接するエントリの各対が、第１のステップで最初に交換され、それに続いて、２つのエントリの第１のブロックが第２のブロックと交換され、２つのエントリの第３のブロックが第４のブロックと交換されるということを意味する。その結果が、図８に、各バンクのエントリの横の縞模様マークで示されている。

【００８１】

図１１は、図１と比較した場合により実装に近い形態で、読み取りプロセスのアクセス・オフセット生成のフローチャート１１００を示している。

【００８２】

簡単に言うと、アクションのフローは次のように実行される（詳細が、さらに下でも説明される）。まず、アドレス「ａ」が取得される（１１０２）。次に、アドレス「ａ」のパリティ、Ｙフィールド、および初期オフセットが導出される（１１０４）。Ｙフィールド（またはパリティ）に基づいてマッピング・テーブル内の行にアクセスすることによって、すべてのバンクのベース・オフセットが取得される（１１０６）。次に、すべてのバンクのベース・オフセットが、対象の２の累乗のストライドに対応するビット位置にシフトされる（１１０８）。最後に、アドレス「ａ」に対して決定された初期オフセットが、マッピング・テーブルから取得されたシフトされたベース・オフセットとマージされ（１１１０）、すべてのメモリ・バンクのオフセットを取得する。

【００８３】

この簡潔なフローチャート表現は、次の段落で再び説明される。

【００８４】

データ・ベクトル要素をアドレス・シーケンス内のアドレスのメモリ・バンクに書き込むためのデータ・ベクトル要素の効率的なシャッフルは、ｎ個のメモリ・バンクにわたって任意の可能な方法でマッピングすることができ、各データ要素を正しいバンクに書き込むために対応する書き込みデータをシャッフルする必要がある、合計ｎ！個の可能性が存在する。大きい値のｎの場合、この可能性が急激に増大し、非常に大きいｎ対ｎシャッフル・ユニットが必要になる。

【００８５】

しかし、前述したマッピング方式の場合、シャッフルを大幅に簡略化することができる。このことが、上記の例示的なマッピング方式を使用してここで説明される。

【００８６】

以下では、パリティ値（すなわち、Ｘ）が、オフセット値と完全に同じように順序付けられるということを仮定する。この仮定は、異なる最下位アドレス・ビットが最下位パリティ（Ｘ）ビットによってカバーされ、異なる第２のアドレス・ビットが次のパリティ・ビットによってカバーされる、などとなる場合に当てはまる。任意の２の累乗のストライドのアクセスで、この制約をどのように満たすことができるかについて、アドレス／パリティ・ビットの選択を対象にする以下の段落でさらに説明する。

【００８７】

マッピング・テーブルが開始される方法（「交換」特性）のため、バンク・オフセット・ベクトルの生成と同様に、ルックアップ・テーブル内の単一の行によってシャッフルが決定される。次のようなマッピングの興味深い特性を、図６のマッピング・テーブルから直接理解することができる。マッピング・テーブル内の各行は、ｋ個の連続する行要素をカバーするブロックを使用して、「アライメントされた方法」で行要素を交換することによって、（Ｙ＝０に対応する）第１の行から導出されることができ、ｋは２の累乗であり、２～行サイズｎ／２の範囲内の値を有する。

【００８８】

図６の例では、２つの要素を含む４つのブロック（それぞれ、｛０，　１｝、｛２，　３｝、｛４，　５｝、｛６，　７｝）を識別し、各ブロック内すべての要素を交換することによって、Ｙ＝１に対応する行が、Ｙ＝０に対応する第１の行から導出され得る。Ｙ＝２に対応する次の行は、最初に、４つの要素を含む２つのブロック（それぞれ、｛０，　１，　２，　３｝および｛４，　５，　６，　７｝）を識別し、次に、それらの各ブロック｛０，１｝および｛２，　３｝、｛４，　５｝および｛６，　７｝の各々において２つの要素のサイズを有する２つのブロックを識別し、それらの２つの２要素ブロックの各々を４要素ブロック内で交換することによって導出され得る。４要素ブロックのレベルで、または４要素ブロック内の２要素ブロックのレベルで交換することによって、あるいはこれらの両方の組み合わせによって、残りのすべての行が第１の行から導出され得るということが、図６からわかる。

【００８９】

図６および８の例からわかる別の重要な特性は、Ｚベクトルの最上位ビットを使用して、そのマッピング・テーブルの最大のブロック・サイズのレベルで交換動作を実行する必要があるかどうかを判定することができ、Ｚベクトルの次のビットを使用して、２番目に大きいブロック・サイズのレベルで交換動作を実行する必要があるかどうかを判定することができる、などとなるということである。

【００９０】

「交換」特性のため、ここで、次のような方法で同じブロック交換の概念を用いて、前述した特性を、書き込みデータをシャッフルすることに直接使用できる。（１）（前述したように）ｎ個のアドレスのシーケンスの開始アドレスについて、Ｚ値が決定される、（２）Ｚ値の最上位ビットが設定されている場合、ｎ／２である最大ブロック・サイズのレベルで、ｎ個の書き込みデータ要素が交換される、（３）Ｚ値の次の上位ビットが設定されている場合、ｎ／４である次のブロック・サイズのレベルで、ｎ個の書き込みデータ要素が交換される、（４）などとなる。これが、図１２において説明される。４つ、８つ、または１６個のバンクを含むメモリ・システムの場合、これは、それぞれ２つ、３つ、４つの異なる両方向シャッフル機能のみが必要になり、完全なｎ対ｎシャッフル・ユニットと比較して著しい単純化であるということを意味する。

【００９１】

したがって、やはり簡単に言うと、書き込み動作は、図１２のフローチャート１２００において要約され得る。まず、アドレス「ａ」が取得される（１２０２）。ステップ１２０４で、パリティおよびＺフィールドが導出される。次に、Ｚフィールドのビット位置（$\log_2(n)$）－１に基づいて、ｎ／２のブロック・サイズの粒度で、右側のデータ要素がシャッフルされる（１２０６）。ステップ１２０８で、Ｚフィールドの位置（$\log_2(n)$）－２に基づいて、ｎ／４のブロック・サイズの粒度で、再び同じことが発生する、などとなる。このプロセスが繰り返され（破線矢印で示される）、ステップ１２１０（Ｚフィールドのビット位置０に基づく、２のブロック・サイズの粒度での、右側のデータ要素のシャッフル）を回答する。

【００９２】

新しいシャッフル機能の基本的特性は次のとおりである。

１．それぞれ２つの入力のみをシャッフルする$\log_2(n)$個のシャッフル機能（最大でｎ＊ｗ個の２入力マルチプレクサをそれぞれ必要とし、ｗは各データ要素の幅である）

のみを使用して実装され得る。
２．各シャッフル機能は、Ｚベクトルまたはパリティ機能（それぞれ、log_2（ｎ）のビット幅である）の１ビットによって制御される。
【００９３】
個別の書き込みイネーブル信号が、各書き込みデータ要素に使用される場合、各書き込みデータ要素を含む正しい書き込み信号を適用するために、それらの書き込みイネーブル信号が同じ方法でシャッフルされる必要がある。
【００９４】
３．メモリ・バンクから読み取った後のデータ・ベクトル要素の効率的なシャッフル。前述したマッピング方式の「交換」特性のため、書き込みデータに関して上で説明されたシャッフルの概念と同じ概念が、読み取りデータのシャッフルに使用され得る。
【００９５】
チップ面積またはＦＰＧＡのリソースが非常に制限されている場合、log_2（ｎ）個のシャッフラーの１セットを、データの読み取りまたは書き込みのいずれかに使用されるメモリ・ポートに使用することもできる。明らかに、そのような使用は、シャッフルされる必要がある読み取りデータ要素または書き込みデータ要素を選択するために、シャッフル段の前に入力マルチプレクサを必要とする。
【００９６】
４．パリティ・ビットの各々の計算に使用されるアドレス・ビットの選択。前に示したように、上記の書き込みデータおよび読み取りデータのシャッフルの概念は、パリティ値（Ｘ）がオフセット値と完全に同じように順序付けられるということを仮定する。この仮定は、アライメントされたアドレス・シーケンス内の異なる最下位アドレス・ビットが最下位パリティ（Ｘ）ビットによってカバーされ、異なる第２のアドレス・ビットが次のパリティ・ビットによってカバーされる、などとなることを必要とする。
【００９７】
これが当てはまらない例は、図８に示されているマッピング方式のストライド２を伴うアドレス・シーケンス０、２、４、６、８、１０、１２、および１４である。この場合、８つのアドレスの間のlog_2（ｎ）＝$log_2$８＝３つの異なるビット位置は、１、２、および３である。それらのビット位置が、同じ順序でＸビット位置に対応しないため、結果として生じるＸ値は、アドレス０のＸ＝０、アドレス２のＸ＝２、アドレス４のＸ＝４、アドレス６のＸ＝６、アドレス８のＸ＝１、アドレス１０のＸ＝３、アドレス１２のＸ＝５、およびアドレス１４のＸ＝７というように、アドレスと同じ方法で順序付けられない。これらのアドレスは、競合せずに、すべてのメモリ・バンクにわたって完全にインターリーブされるが、上記の簡略化されたシャッフル動作は不十分である。
【００９８】
この問題は、（１）追加のシャッフル層を追加すること、または（２）パリティ・ベクトルＸを生成するために使用されるアドレス・ビットの異なる選択を適用すること、のいずれかによって解決され得る。第１のオプションは、２の累乗のストライドを伴うアドレス・シーケンス内のlog_2（ｎ）個の異なるビット位置が、Ｘフィールドのビットが生成される方法を「使用して」アライメントされない、（log_2（ｎ））－１個の方法が存在する可能性があるため、（log_2（ｎ））－１個の追加の層の追加を必要とする。したがって、図８の例の場合、（log_2（ｎ））－１＝１つの追加のシャッフル層が必要になる。
【００９９】
第２のオプションは、アドレスからＸビットが導出される方法を変更する。
【０１００】
上記の例では、同じ結果を取得するために、２つのＸビット（Ｘ０およびＸ１）の位置が反転され得る。代替として、関連する２の累乗のストライドに関連するアドレス・ビット位置のみから、Ｘビットが導出され得る。これを実行することによって、前述した書き込みデータおよび読み取りデータのシャッフルの概念と同じ概念が使用され得る。この選

択は、それに応じて、パリティが計算されるアドレス・ビットを選択するために、アドレスに適用されるＸビット位置ごとにマスク・ベクトルを選択することによって行われ得る（マスクされたアドレス・ビットは、０を与え、ＸＯＲに基づくパリティ計算を変更しない）。

【０１０１】

また、このために、アライメントされたアドレス・シーケンス内の異なるビットに関連しない他のすべてのアドレス・ビット位置は、パリティ計算に含まれる必要がない（バンク・オフセットの一部でないすべてのビットがパリティ計算の対象になる必要がある、テーブルなしの方法の場合を除く）。

【０１０２】

完全性の理由から、図１３は、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするためのメモリ・アクセス・ユニット１４００の概略ブロック図を示している。メモリ・アクセス・ユニット１４００は、バイナリ・データ・ベクトルの開始アドレスおよびバイナリ・データ・ベクトルの要素の２の累乗のストライドを受信するように適応された受信ユニット１４０２を備えている。

【０１０３】

メモリ・アクセス・ユニット１４００は、メモリ・バンクの各々について１つのｎ個のオフセットを決定するように適応された第１の決定モジュール１４０４をさらに備え、オフセットの各々は、複数のビットレベルのＸＯＲ関数を開始アドレスに適用してＺベクトルを生成し、マッピング・テーブルにアクセスするためにＺベクトルを使用し、バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることによって決定される。

【０１０４】

さらに、メモリ・アクセス・ユニット１４００は、Ｚベクトルの２進等価値に応じてｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定するように適応された第２の決定モジュール１４０６と、メモリ・ユニットのｎ個のメモリ・バンク内のバイナリ・データ・ベクトルに並列にアクセスするように適応された基本アクセス・ユニット１４０８とも備えている。

【０１０５】

本発明の実施形態は、プログラム・コードを格納することまたは実行することあるいはその両方に適しているプラットフォームに関わらず、事実上、任意の種類のコンピュータと一緒に実装されてよい。図１４は、一例として、提案された方法に関連するプログラム・コードを実行するのに適しているか、またはメモリ・ユニットに関して本明細書で提案されたアドレス指定方式が実装されてよいか、あるいはその両方であるコンピューティング・システム１５００を示している。

【０１０６】

コンピューティング・システム１５００は、適切なコンピュータ・システムの一例にすぎず、コンピュータ・システム１５００が上記で示された機能のいずれかを実装されること、または実行すること、あるいはその両方を行うことができるかどうかに関わらず、本明細書に記載された本発明の実施形態の使用または機能の範囲に関してどのような制限も示唆するよう意図されていない。コンピュータ・システム１５００には、他の多数の汎用または専用のコンピューティング・システム環境または構成と共に動作できるコンポーネントが存在する。コンピュータ・システム／サーバ１５００と共に使用するのに適した周知のコンピューティング・システム、環境、または構成、あるいはその組み合わせの例としては、パーソナル・コンピュータ・システム、サーバ・コンピュータ・システム、シン・クライアント、シック・クライアント、ハンドヘルドまたはラップトップ・デバイス、マイクロプロセッサ・システム、マイクロプロセッサベース・システム、セット・トップ・ボックス、プログラマブル・コンシューマ・エレクトロニクス、ネットワークＰＣ、マイクロコンピュータ・システム、メインフレーム・コンピュータ・システム、およびこれ

らの任意のシステムまたはデバイスを含む分散クラウド・コンピューティング環境などが挙げられるが、これらに限定されない。コンピュータ・システム／サーバ１５００は、コンピュータ・システム１５００によって実行されているプログラム・モジュールなどの、コンピュータ・システムによって実行可能な命令との一般的な関連において説明されてよい。通常、プログラム・モジュールは、特定のタスクを実行するか、または特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、論理、データ構造などを含んでよい。コンピュータ・システム／サーバ１５００は、通信ネットワークを介してリンクされたリモート処理デバイスによってタスクが実行される、分散クラウド・コンピューティング環境で実行されてよい。分散クラウド・コンピューティング環境において、プログラム・モジュールは、メモリ・ストレージ・デバイスを含む、ローカルおよびリモートの両方のコンピュータ・システム・ストレージ媒体に配置されてよい。

【０１０７】

図に示されているように、コンピュータ・システム／サーバ１５００は、汎用コンピューティング・デバイスの形態で示されている。コンピュータ・システム／サーバ１５００のコンポーネントは、１つまたは複数のプロセッサまたはプロセッシング・ユニット１５０２、システム・メモリ１５０４、およびシステム・メモリ１５０４を含むさまざまなシステム・コンポーネントをプロセッサ１５０２に結合するバス１５０６を含むことができるが、これらに限定されない。バス１５０６は、メモリ・バスまたはメモリ・コントローラ、ペリフェラル・バス、アクセラレーテッド・グラフィックス・ポート、およびさまざまなバス・アーキテクチャのいずれかを使用するプロセッサまたはローカル・バスを含む、複数の種類のバス構造のいずれかのうちの１つまたは複数を表す。例として、そのようなアーキテクチャは、ＩＳＡ（Industry Standard Architecture）バス、ＭＣＡ（Micro Channel Architecture）バス、ＥＩＳＡ（Enhanced ISA）バス、ＶＥＳＡ（Video Electronics Standards Association）ローカル・バス、およびＰＣＩ（Peripheral Component Interconnects）バスを含むが、これらに限定されない。コンピュータ・システム／サーバ１５００は、通常、さまざまなコンピュータ・システム可読媒体を含む。そのような媒体は、コンピュータ・システム／サーバ１５００によってアクセスできる任意の使用可能な媒体であってよく、揮発性および不揮発性媒体、取り外し可能および取り外し不可の媒体を含む。

【０１０８】

システム・メモリ１５０４は、ランダム・アクセス・メモリ（ＲＡＭ：random access memory）１５０８またはキャッシュ・メモリ１５１０あるいはその両方などの、揮発性メモリの形態でコンピュータ・システム可読媒体を含んでよい。コンピュータ・システム／サーバ１５００は、その他の取り外し可能／取り外し不可、揮発性／不揮発性のコンピュータ・システム・ストレージ媒体をさらに含んでよい。単に例として、取り外し不可、不揮発性の磁気媒体（図示されておらず、通常は「ハード・ドライブ」と呼ばれる）に対する読み取りと書き込みを行うために、ストレージ・システム１５１２が提供されてよい。図示されていないが、取り外し可能、不揮発性の磁気ディスク（例えば、「フロッピー（Ｒ）・ディスク」）に対する読み取りと書き込みを行うための磁気ディスク・ドライブ、およびＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ、またはその他の光媒体などの取り外し可能、不揮発性の光ディスクに対する読み取りと書き込みを行うための光ディスク・ドライブが提供されてよい。そのような例では、それぞれを、１つまたは複数のデータ媒体インターフェイスによってバス１５０６に接続することができる。下で詳細に示され、説明されているように、メモリ１５０４は、本発明の実施形態の機能を実行するように構成された一連の（例えば、少なくとも１つの）プログラム・モジュールを備える少なくとも１つのプログラム製品を含んでよい。

【０１０９】

例えば、一連の（少なくとも１つの）プログラム・モジュール１５１６を含んでいるプログラム／ユーティリティがメモリ１５０４に格納されてよいが、これに限定されず、オペレーティング・システム、１つまたは複数のアプリケーション・プログラム、その他の

プログラム・モジュール、およびプログラム・データも格納されてよい。オペレーティング・システム、１つまたは複数のアプリケーション・プログラム、その他のプログラム・モジュール、およびプログラム・データまたはこれらの組み合わせの各々は、ネットワーク環境の実装を含んでよい。プログラム・モジュール１５１６は、通常、本明細書に記載された本発明の実施形態の機能または方法あるいはその両方を実行する。

【０１１０】

また、コンピュータ・システム／サーバ１５００は、キーボード、ポインティング・デバイス、ディスプレイ１５２０などの１つまたは複数の外部デバイス１５１８、ユーザがコンピュータ・システム／サーバ１５００と情報をやりとりできるようにする１つまたは複数のデバイス、またはコンピュータ・システム／サーバ１５００が１つまたは複数の他のコンピューティング・デバイスと通信できるようにする任意のデバイス（例えば、ネットワーク・カード、モデムなど）、あるいはその組み合わせと通信することもできる。そのような通信は、入出力（Ｉ／Ｏ：Input/Output）インターフェイス１５１４を介して行うことができる。さらに、コンピュータ・システム／サーバ１５００は、ローカル・エリア・ネットワーク（ＬＡＮ：local area network）、一般的な広域ネットワーク（ＷＡＮ：wide area network）、またはパブリック・ネットワーク（例えば、インターネット）、あるいはその組み合わせなどの１つまたは複数のネットワークと、ネットワーク・アダプタ１５２２を介して通信してよい。図示されているように、ネットワーク・アダプタ１５２２は、バス１５０６を介してコンピュータ・システム／サーバ１５００の他のコンポーネントと通信してよい。図示されていないが、その他のハードウェア・コンポーネントまたはソフトウェア・コンポーネントあるいはその両方を、コンピュータ・システム／サーバ１５００と併用できるということが理解されるべきである。その例として、マイクロコード、デバイス・ドライバ、冗長プロセッシング・ユニット、外部ディスク・ドライブ・アレイ、ＲＡＩＤシステム、テープ・ドライブ、およびデータ・アーカイブ・ストレージ・システムなどが挙げられるが、これらに限定されない。

【０１１１】

さらに、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするためのメモリ・アクセス・ユニット１４００が、バス・システム１５０６に取り付けられてよい。

【０１１２】

本発明のさまざまな実施形態の説明は、例示の目的で提示されているが、網羅的であることは意図されておらず、開示された実施形態に制限されない。説明された実施形態の範囲および精神を逸脱することなく多くの変更および変形が可能であることは、当業者にとって明らかである。本明細書で使用された用語は、実施形態の原理、実際の適用、または市場で見られる技術を超える技術的改良を最も適切に説明するため、または他の当業者が本明細書で開示された実施形態を理解できるようにするため選択されている。

【０１１３】

本発明は、システム、方法、またはコンピュータ・プログラム製品、あるいはその組み合わせとして具現化されてよい。コンピュータ・プログラム製品は、プロセッサに本発明の態様を実行させるためのコンピュータ可読プログラム命令を含んでいるコンピュータ可読ストレージ媒体を含んでよい。

【０１１４】

この媒体は、電子、磁気、光、電磁気、赤外線、または伝搬媒体用の半導体システムであってよい。コンピュータ可読媒体の例としては、半導体メモリまたは固体メモリ、磁気テープ、取り外し可能フロッピー（Ｒ）・ディスク、ランダム・アクセス・メモリ（ＲＡＭ：random access memory）、読み取り専用メモリ（ＲＯＭ：read-only memory）、剛体磁気ディスク、および光ディスクが挙げられる。光ディスクの現在の例は、コンパクト・ディスク読み取り専用メモリ（ＣＤ－ＲＯＭ：compact disk-read only memory）、コンパクト・ディスク読み取り／書き込み（ＣＤ－Ｒ／Ｗ：compact disk-read/write）、およびブルーレイディスクを含む。

【０１１５】
コンピュータ可読ストレージ媒体は、命令実行デバイスによって使用するための命令を保持および格納できる有形のデバイスであることができる。コンピュータ可読ストレージ媒体は、例えば、電子ストレージ・デバイス、磁気ストレージ・デバイス、光ストレージ・デバイス、電磁ストレージ・デバイス、半導体ストレージ・デバイス、またはこれらの任意の適切な組み合わせであってよいが、これらに限定されない。コンピュータ可読ストレージ媒体のさらに具体的な例の非網羅的リストは、ポータブル・フロッピー（Ｒ）・ディスク、ハード・ディスク、ランダム・アクセス・メモリ（ＲＡＭ）、読み取り専用メモリ（ＲＯＭ）、消去可能プログラマブル読み取り専用メモリ（ＥＰＲＯＭ：erasable programmable read-only memoryまたはフラッシュ・メモリ）、スタティック・ランダム・アクセス・メモリ（ＳＲＡＭ：static random access memory）、ポータブル・コンパクト・ディスク読み取り専用メモリ（ＣＤ－ＲＯＭ：compact disc read-only memory）、デジタル多用途ディスク（ＤＶＤ：digital versatile disk）、メモリ・スティック、フロッピー（Ｒ）・ディスク、パンチカードまたは命令が記録されている溝の中の隆起構造などの機械的にエンコードされるデバイス、およびこれらの任意の適切な組み合わせを含む。本明細書において使用されるとき、コンピュータ可読ストレージ媒体は、それ自体が、電波またはその他の自由に伝搬する電磁波、導波管またはその他の送信媒体を伝搬する電磁波（例えば、光ファイバ・ケーブルを通過する光パルス）、あるいはワイヤを介して送信される電気信号などの一過性の信号であると解釈されるべきではない。

【０１１６】
本明細書に記載されたコンピュータ可読プログラム命令は、コンピュータ可読ストレージ媒体から各コンピューティング・デバイス／処理デバイスへ、またはネットワーク（例えば、インターネット、ローカル・エリア・ネットワーク、広域ネットワーク、または無線ネットワーク、あるいはその組み合わせ）を介して外部コンピュータまたは外部ストレージ・デバイスへダウンロードされ得る。このネットワークは、銅伝送ケーブル、光伝送ファイバ、無線送信、ルータ、ファイアウォール、スイッチ、ゲートウェイ・コンピュータ、またはエッジ・サーバ、あるいはその組み合わせを備えてよい。各コンピューティング・デバイス／処理デバイス内のネットワーク・アダプタ・カードまたはネットワーク・インターフェイスは、コンピュータ可読プログラム命令をネットワークから受信し、それらのコンピュータ可読プログラム命令を各コンピューティング・デバイス／処理デバイス内のコンピュータ可読ストレージ媒体に格納するために転送する。

【０１１７】
本発明の動作を実行するためのコンピュータ可読プログラム命令は、アセンブラ命令、命令セット・アーキテクチャ（ＩＳＡ：instruction-set-architecture）命令、マシン命令、マシン依存命令、マイクロコード、ファームウェア命令、状態設定データ、あるいは、Ｓｍａｌｌｔａｌｋ、Ｃ＋＋などのオブジェクト指向プログラミング言語、および「Ｃ」プログラミング言語または同様のプログラミング言語などの従来の手続き型プログラミング言語を含む１つまたは複数のプログラミング言語の任意の組み合わせで記述されたソース・コードまたはオブジェクト・コードであってよい。コンピュータ可読プログラム命令は、ユーザのコンピュータ上で全体的に実行すること、ユーザのコンピュータ上でスタンドアロン・ソフトウェア・パッケージとして部分的に実行すること、ユーザのコンピュータ上およびリモート・コンピュータ上でそれぞれ部分的に実行すること、あるいはリモート・コンピュータ上またはサーバ上で全体的に実行することができる。後者のシナリオでは、リモート・コンピュータは、ローカル・エリア・ネットワーク（ＬＡＮ）または広域ネットワーク（ＷＡＮ）を含む任意の種類のネットワークを介してユーザのコンピュータに接続されてよく、または接続は、（例えば、インターネット・サービス・プロバイダを使用してインターネットを介して）外部コンピュータに対して行われてよい。一部の実施形態では、本発明の態様を実行するために、例えばプログラマブル論理回路、フィールドプログラマブル・ゲート・アレイ（ＦＰＧＡ：field-programmable gate arrays）、またはプログラマブル・ロジック・アレイ（ＰＬＡ：programmable logic arrays）を

含む電子回路は、コンピュータ可読プログラム命令の状態情報を利用することによって、電子回路をカスタマイズするためのコンピュータ可読プログラム命令を実行してよい。

【０１１８】

本発明の態様は、本明細書において、本発明の実施形態に従って、方法、装置（システム）、およびコンピュータ・プログラム製品のフローチャート図またはブロック図あるいはその両方を参照して説明される。フローチャート図またはブロック図あるいはその両方の各ブロック、ならびにフローチャート図またはブロック図あるいはその両方に含まれるブロックの組み合わせが、コンピュータ可読プログラム命令によって実装され得るということが理解されるであろう。

【０１１９】

これらのコンピュータ可読プログラム命令は、コンピュータまたはその他のプログラム可能なデータ処理装置のプロセッサを介して実行される命令が、フローチャートまたはブロック図あるいはその両方のブロックに指定される機能／動作を実施する手段を作り出すべく、汎用コンピュータ、専用コンピュータ、または他のプログラム可能なデータ処理装置のプロセッサに提供されてマシンを作り出すものであってよい。これらのコンピュータ可読プログラム命令は、命令が格納されたコンピュータ可読ストレージ媒体がフローチャートまたはブロック図あるいはその両方のブロックに指定される機能／動作の態様を実施する命令を含んでいる製品を備えるように、コンピュータ可読ストレージ媒体に格納され、コンピュータ、プログラム可能なデータ処理装置、または他のデバイス、あるいはその組み合わせに特定の方式で機能するように指示できるものであってもよい。

【０１２０】

コンピュータ可読プログラム命令は、コンピュータ、その他のプログラマブル・データ処理装置、あるいは一連の動作可能なステップをコンピュータ上、その他のプログラマブル装置上、またはコンピュータ実装プロセスを生成するその他のデバイス上で実行させる別のデバイスに読み込むこともでき、それによって、コンピュータ上、その他のプログラマブル装置上、または別のデバイス上で実行される命令は、フローチャートまたはブロック図あるいはその両方のブロックにおいて規定された機能／動作を実装する。

【０１２１】

図内のフローチャートまたはブロック図あるいはその両方は、本発明のさまざまな実施形態に従って、システム、方法、およびコンピュータ・プログラム製品の可能な実装のアーキテクチャ、機能、および動作を示す。これに関連して、フローチャートまたはブロック図内の各ブロックは、規定された論理機能を実装するための１つまたは複数の実行可能な命令を備える、命令のモジュール、セグメント、または部分を表してよい。一部の代替の実装では、ブロックに示された機能は、図に示された順序とは異なる順序で発生してよい。例えば、連続して示された２つのブロックは、実際には、含まれている機能に応じて、実質的に同時に実行されるか、または場合によっては逆の順序で実行されてよい。ブロック図またはフローチャート図あるいはその両方の各ブロック、ならびにブロック図またはフローチャート図あるいはその両方に含まれるブロックの組み合わせは、規定された機能または動作を実行するか、または専用ハードウェアとコンピュータ命令の組み合わせを実行する専用ハードウェアベースのシステムによって実装できるということにも注意する。

【０１２２】

本明細書で使用される用語は、特定の実施形態を説明することのみを目的としており、本発明を制限することを意図していない。本明細書で使用される単数形「ａ」、「ａｎ」、および「ｔｈｅ」は、特に明示的に示されない限り、複数形も含むことが意図されている。「備える」または「備えている」あるいはその両方の用語は、本明細書で使用される場合、記載された機能、整数、ステップ、動作、要素、またはコンポーネント、あるいはその組み合わせの存在を示すが、１つまたは複数のその他の機能、整数、ステップ、動作、要素、コンポーネント、またはこれらのグループ、あるいはその組み合わせの存在または追加を除外していないということが、さらに理解されるであろう。

【０１２３】

下の特許請求の範囲内のすべての手段またはステップおよび機能要素の対応する構造、材料、動作、および等価なものは、具体的に請求されるその他の請求された要素と組み合わせて機能を実行するための任意の構造、材料、または動作を含むことが意図されている。本発明の説明は、例示および説明の目的で提示されているが、網羅的であることは意図されておらず、開示された形態での発明に制限されない。本発明の範囲および精神を逸脱することなく多くの変更および変形が可能であることは、当業者にとって明らかである。本発明の原理および実際的な適用を最も適切に説明するため、およびその他の当業者が、企図されている特定の用途に適しているようなさまざまな変更を伴う多様な実施形態に関して、本発明を理解できるようにするために、実施形態が選択されて説明された。

【０１２４】

簡単に言うと、本発明の概念は、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするための方法として要約されてよく、この方法は、バイナリ・データ・ベクトルの開始アドレスおよびバイナリ・データ・ベクトルの要素の２の累乗のストライドｓを受信することと、メモリ・バンクの各々について１つのｎ個のオフセットを決定することであって、オフセットの各々が、複数のビットレベルのＸＯＲ関数を開始アドレスに適用してＺベクトルを生成し、マッピング・テーブルにアクセスするためにＺベクトルを使用し、バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることによって決定される、決定することと、Ｚベクトルの２進等価値に応じてｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定することと、メモリ・ユニットのｎ個のメモリ・バンク内のバイナリ・データ・ベクトルに並列にアクセスすることとを含む。

【０１２５】

この方法では、複数のビットレベルのＸＯＲ関数を開始アドレスに適用することは、開始アドレスから、log_2（ｎ）個の部分およびlog_2（ｎ）ビット・サイズのＹの部分を選択することと、log_2（ｎ）個の部分の各々についてパリティ・ビットを決定し、log_2（ｎ）次元のパリティ・ベクトルＸを生成することと、ＸとＹの間でビットレベルのＸＯＲ演算を実行してＺベクトルを生成することとを含んでよい。

【０１２６】

マッピング・テーブルにアクセスするためにＺベクトルを使用することは、マッピング・テーブル内のインデックスとしてＺベクトルを使用することによって、log_2（ｓ）個のエントリを含んでいるマッピング・テーブル内の行を選択することと、マッピング・テーブルのアクセス結果を開始アドレスの部分と結合してｎ個のオフセットを取得することとを含んでよい。

【０１２７】

この結合することは、開始アドレスのビット・オフセットlog_2（ｎ）にあるlog_2（ｎ）ビットを、決定されたlog_2（ｎ）個のマッピング・テーブル・エントリに置き換え、log_2（ｎ）個のオフセットを生成することを含んでよく、各オフセットは、log_2（ｎ）個のバンクの各１つにおいてオフセットとして使用される。

【０１２８】

ｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定することは、log_2（ｎ）個の連続するマルチプレクサ段を制御することを含んでよく、マルチプレクサ段の各々は、データ・ベクトルの全幅をカバーするように、複数の２入力ビット・マルチプレクサを含んでよい。

【０１２９】

アクセス動作は、読み取り動作であってよい。

【０１３０】

アクセス動作は、書き込み動作であってよい。

【０１３１】

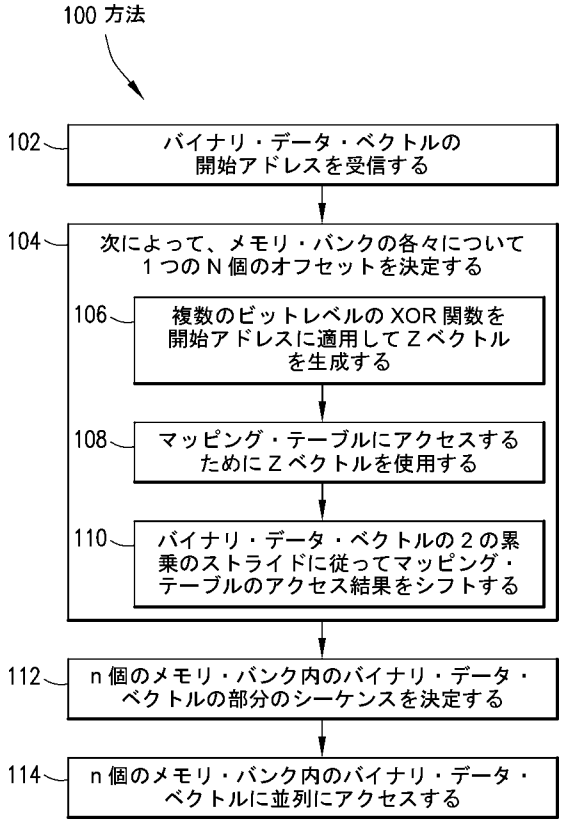
開始アドレスの選択されたlog_2（ｎ）個の部分は、同じサイズに設定されてよい。

【０１３２】
バイナリ・データ・ベクトルは、複数のｎ個のメモリ・バンク内の同じサイズの部分に格納されてよい。
【０１３３】
開始アドレスの提供、オフセットの決定、ならびにバイナリ・データ・ベクトルの部分のシーケンスの決定、および部分のシーケンスの決定は、ｎ個のメモリ・バンクへのバイナリ・データ・ベクトルの書き込み動作の前に実行されてよい。
【０１３４】
開始アドレスを提供することと、オフセットを決定することと、バイナリ・データ・ベクトルの部分のシーケンスを決定することとは、ｎ個のメモリ・バンクに対するバイナリ・データ・ベクトルの読み取り動作の前に実行されてよく、部分のシーケンスを決定することは、ｎ個のメモリ・バンクに対するバイナリ・データ・ベクトルの読み取り動作の後に実行されてよい。
【０１３５】
メモリ・ユニット内のバイナリ・データ・ベクトルにアクセスするためのメモリ・アクセス・ユニットが、バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでおり、メモリ・アクセス・ユニットは、バイナリ・データ・ベクトルの開始アドレスおよびバイナリ・データ・ベクトルの要素の２の累乗のストライドｓを受信するように適応された受信ユニットと、メモリ・バンクの各々について１つのｎ個のオフセットを決定するように適応された第１の決定モジュールであって、オフセットの各々が、複数のビットレベルのＸＯＲ関数を開始アドレスに適用してＺベクトルを生成し、マッピング・テーブルにアクセスするためにＺベクトルを使用し、バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることによって決定される、第１の決定モジュールと、Ｚベクトルの２進等価値に応じてｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定するように適応された第２の決定モジュールと、メモリ・ユニットのｎ個のメモリ・バンク内のバイナリ・データ・ベクトルに並列にアクセスするように適応された基本アクセス・ユニットとを備える。
【０１３６】
第１の決定モジュールによって複数のビットレベルのＸＯＲ関数を開始アドレスに適用することは、開始アドレスから、$\log_2$（ｎ）個の部分および$\log_2$（ｎ）ビット・サイズのＹの部分を選択することと、$\log_2$（ｎ）個の部分の各々についてパリティ・ビットを決定し、$\log_2$（ｎ）次元のパリティ・ベクトルＸを生成することと、ＸとＹの間でビットレベルのＸＯＲ演算を実行してＺベクトルを生成することとを含んでよい。
【０１３７】
第１の決定モジュールによってマッピング・テーブルにアクセスするためにＺベクトルを使用することは、マッピング・テーブル内のインデックスとしてＺベクトルを使用することによって、$\log_2$（ｓ）個のエントリを含んでいるマッピング・テーブル内の行を選択することと、マッピング・テーブルのアクセス結果を開始アドレスの部分と結合してｎ個のオフセットを取得することとを含んでよい。
【０１３８】
第１の決定モジュールの結合は、開始アドレスのビット・オフセット$\log_2$（ｎ）にある$\log_2$（ｎ）ビットを、決定された$\log_2$（ｎ）個のマッピング・テーブル・エントリに置き換え、$\log_2$（ｎ）個のオフセットを生成することを含んでもよく、各オフセットは、$\log_2$（ｎ）個のバンクの各１つにおいてオフセットとして使用される。
【０１３９】
第２の決定モジュールは、ｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定するときに、$\log_2$（ｎ）個の連続するマルチプレクサ段を制御するように適応されてもよく、マルチプレクサ段の各々は、データ・ベクトルの全幅をカバーするように、複数の２入力ビット・マルチプレクサを含む。

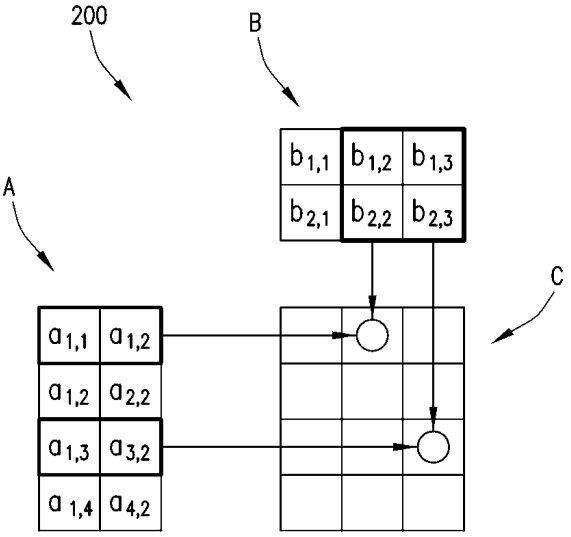
【０１４０】
アクセス動作は、読み取り動作であってよい。
【０１４１】
アクセス動作は、書き込み動作であってよい。
【０１４２】
開始アドレスの選択されたlog_2（ｎ）個の部分は、同じサイズに設定されてよい。
【０１４３】
バイナリ・データ・ベクトルは、複数のｎ個のメモリ・バンク内の同じサイズの部分に格納されてよい。
【０１４４】
開始アドレスの提供、オフセットの決定、ならびにバイナリ・データ・ベクトルの部分のシーケンスの決定、および部分のシーケンスの決定は、ｎ個のメモリ・バンクへのバイナリ・データ・ベクトルの書き込み動作の前に実行されてよい。
【０１４５】
開始アドレスを提供することと、オフセットを決定することと、バイナリ・データ・ベクトルの部分のシーケンスを決定することとは、ｎ個のメモリ・バンクに対するバイナリ・データ・ベクトルの読み取り動作の前に実行されてよく、部分のシーケンスを決定することは、ｎ個のメモリ・バンクに対するバイナリ・データ・ベクトルの読み取り動作の後に実行されてよい。
【０１４６】
バイナリ・データ・ベクトルが一部に格納される複数のｎ個のメモリ・バンクを含んでいるメモリ・コンポーネント内のバイナリ・データ・ベクトルにアクセスするコンピュータ・プログラム製品であって、コンピュータ・プログラム製品が、プログラム命令が具現化されているコンピュータ可読ストレージ媒体を備えており、プログラム命令が、１つまたは複数のコンピューティング・システムに、バイナリ・データ・ベクトルの開始アドレスおよびバイナリ・データ・ベクトルの要素の２の累乗のストライドｓを受信することと、メモリ・バンクの各々について１つのｎ個のオフセットを決定することであって、オフセットの各々が、複数のビットレベルのＸＯＲ関数を開始アドレスに適用してＺベクトルを生成することと、マッピング・テーブルにアクセスするためにＺベクトルを使用することと、バイナリ・データ・ベクトルの２の累乗のストライドに従ってマッピング・テーブルのアクセス結果をシフトすることとによって決定される、決定することと、Ｚベクトルの２進等価値に応じてｎ個のメモリ・バンク内のバイナリ・データ・ベクトルの部分のシーケンスを決定することと、メモリ・ユニットのｎ個のメモリ・バンク内のバイナリ・データ・ベクトルに並列にアクセスすることとを実行させるように、１つまたは複数のコンピューティング・システムまたはコントローラによって実行可能である、コンピュータ・プログラム製品。

【図面】

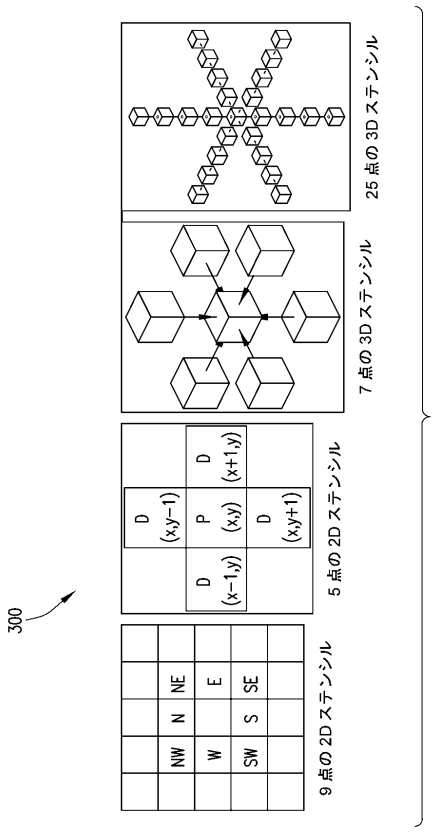
【図１】



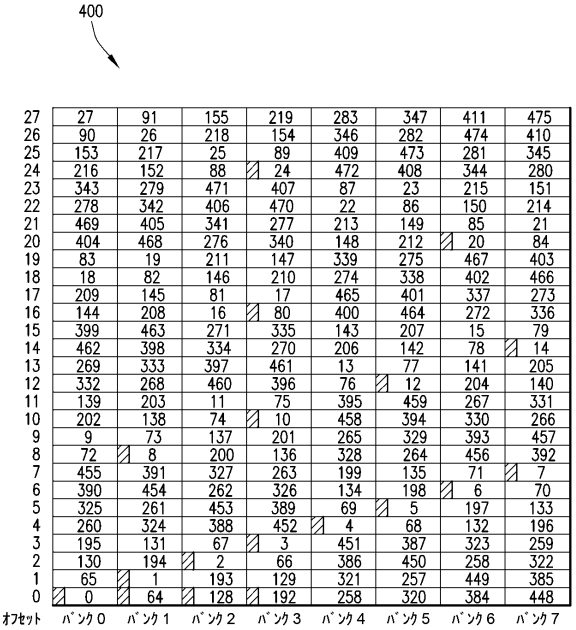
【図２】



【図３】

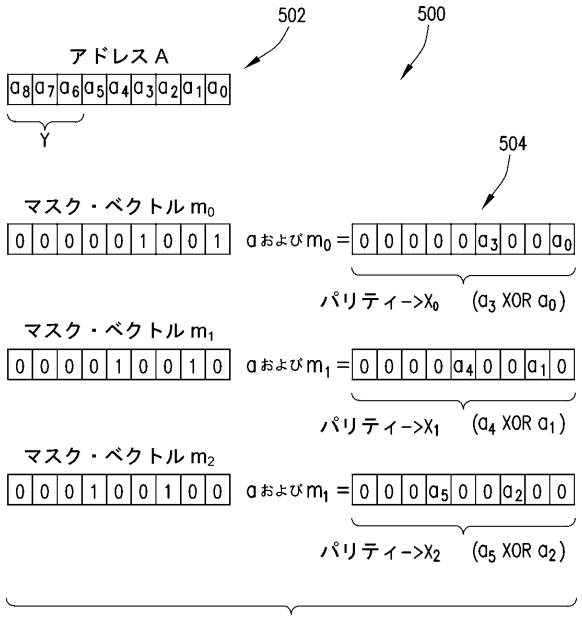


【図４】

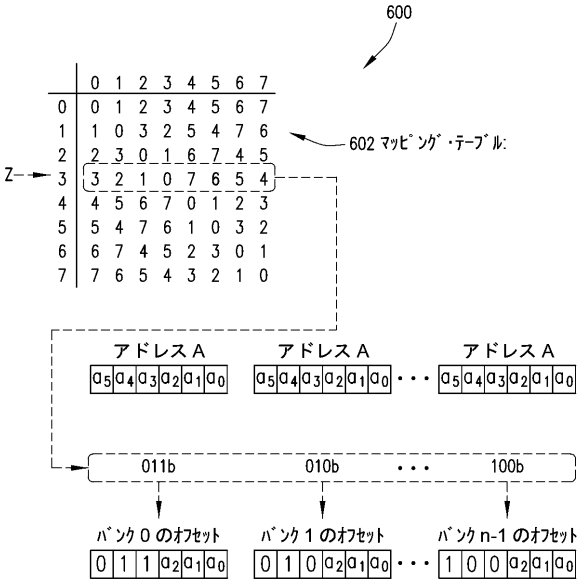


オフセット	バンク０	バンク１	バンク２	バンク３	バンク４	バンク５	バンク６	バンク７
27	27	91	155	219	283	347	411	475
26	90	26	218	154	346	282	474	410
25	153	217	25	89	409	473	281	345
24	216	152	88	24	472	408	344	280
23	343	279	471	407	87	23	215	151
22	278	342	406	470	22	86	150	214
21	469	405	341	277	213	149	85	21
20	404	468	276	340	148	212	20	84
19	83	19	211	147	339	275	467	403
18	18	82	146	210	274	338	402	466
17	209	145	81	17	465	401	337	273
16	144	208	16	80	400	464	272	336
15	399	463	271	335	143	207	15	79
14	462	398	334	270	206	142	78	14
13	269	333	397	461	13	77	141	205
12	332	268	460	396	76	12	204	140
11	139	203	11	75	395	459	267	331
10	202	138	74	10	458	394	330	266
9	9	73	137	201	265	329	393	457
8	72	8	200	136	328	264	456	392
7	455	391	327	263	199	135	71	7
6	390	454	262	326	134	198	6	70
5	325	261	453	389	69	5	197	133
4	260	324	388	452	4	68	132	196
3	195	131	67	3	451	387	323	259
2	130	194	2	66	386	450	258	322
1	65	1	193	129	321	257	449	385
0	0	64	128	192	258	320	384	448

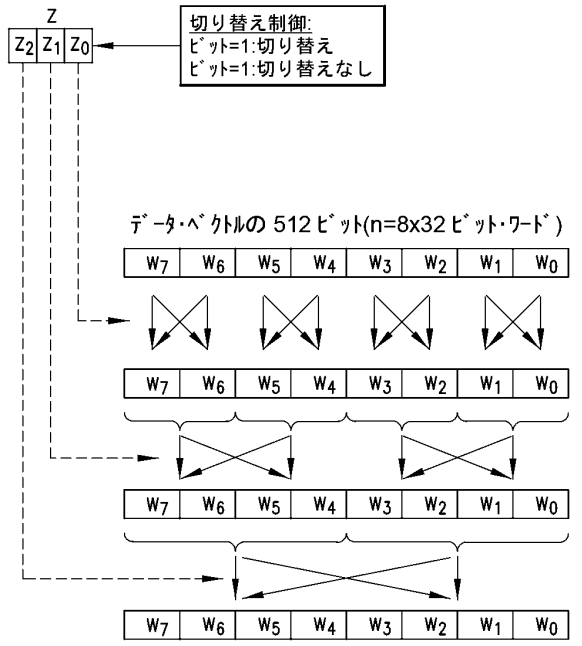
【図５】


502
500
アドレス A
a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0
Y
504
マスク・ベクトル m_0
0 0 0 0 0 1 0 0 1
a および m_0 = 0 0 0 0 0 a_3 0 0 a_0
パリティ->X_0 (a_3 XOR a_0)
マスク・ベクトル m_1
0 0 0 0 1 0 0 1 0
a および m_1 = 0 0 0 0 a_4 0 0 a_1 0
パリティ->X_1 (a_4 XOR a_1)
マスク・ベクトル m_2
0 0 0 1 0 0 1 0 0
a および m_1 = 0 0 0 a_5 0 0 a_2 0 0
パリティ->X_2 (a_5 XOR a_2)

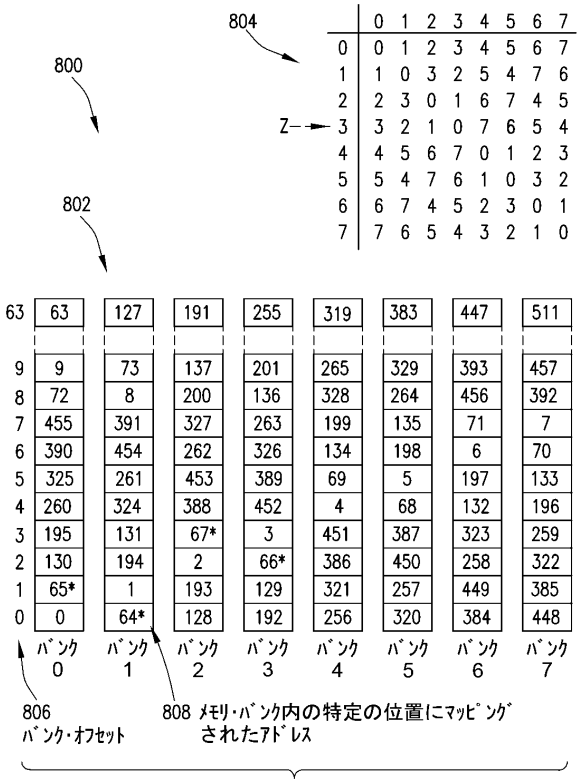
【図６】


600
602 マッピング・テーブル:
Z
アドレス A
a_5 a_4 a_3 a_2 a_1 a_0
011b
010b
100b
バンク 0 のオフセット
バンク 1 のオフセット
バンク n-1 のオフセット
0 1 1 a_2 a_1 a_0
0 1 0 a_2 a_1 a_0
1 0 0 a_2 a_1 a_0

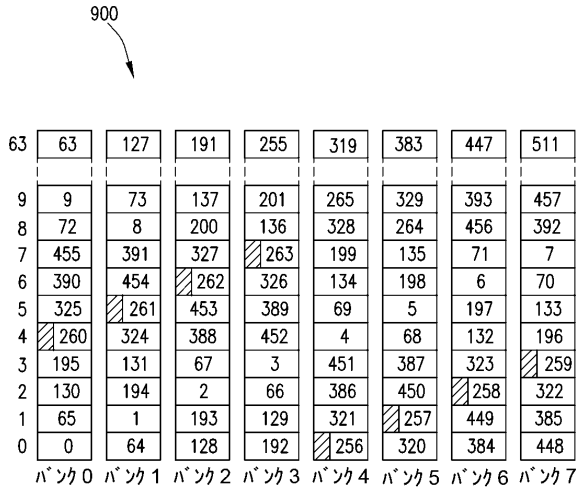
【図７】


Z
Z_2 Z_1 Z_0
切り替え制御:
ビット=1:切り替え
ビット=1:切り替えなし
データ・ベクトルの 512 ビット(n=8x32 ビット・ワード)
W_7 W_6 W_5 W_4 W_3 W_2 W_1 W_0

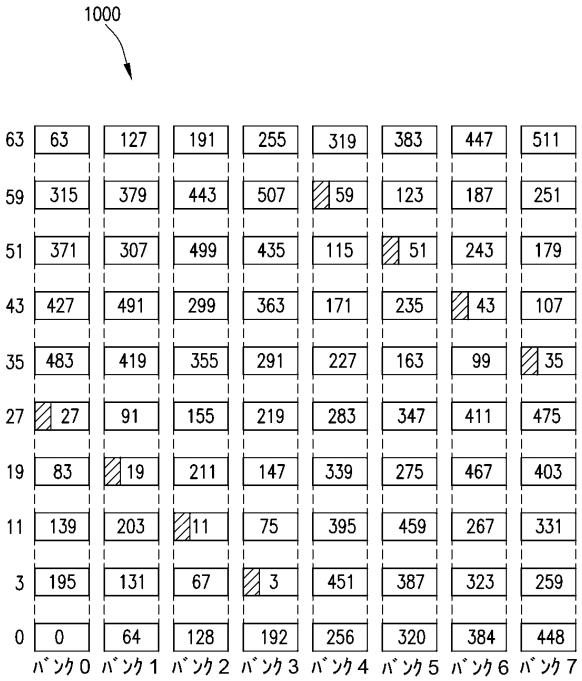
【図８】


804
800
802
Z
806
バンク・オフセット
808 メモリ・バンク内の特定の位置にマッピングされたアドレス
バンク 0
バンク 1
バンク 2
バンク 3
バンク 4
バンク 5
バンク 6
バンク 7

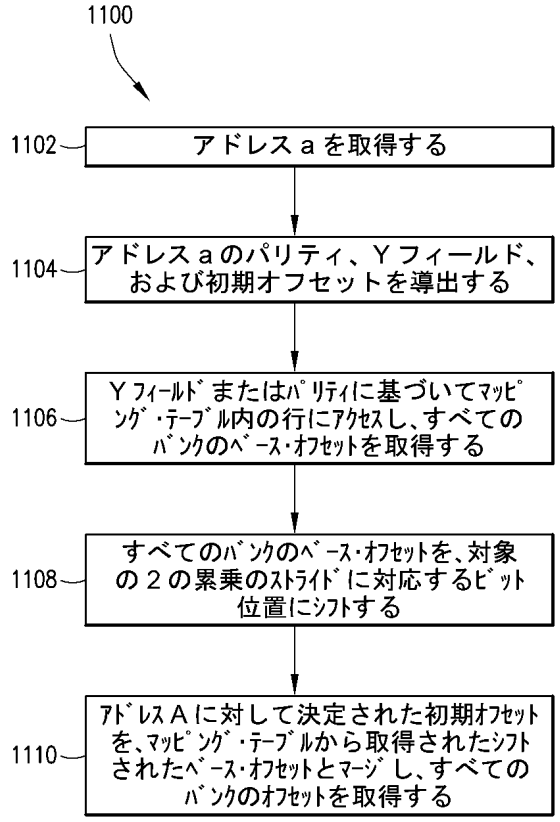
【図９】



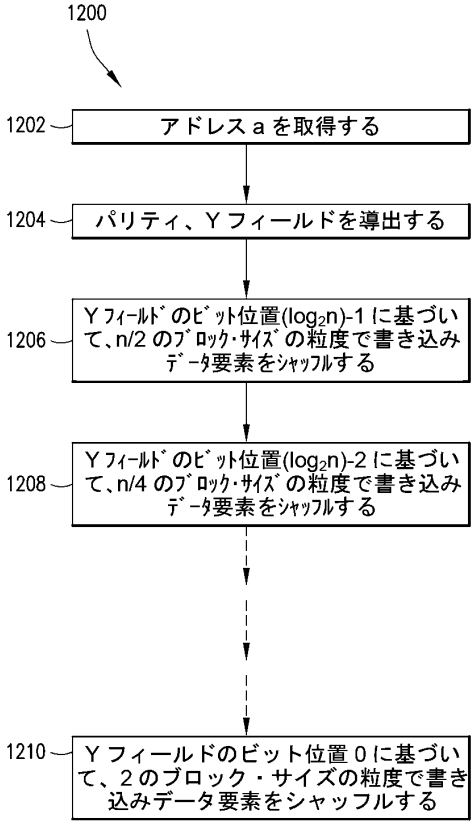
【図１０】



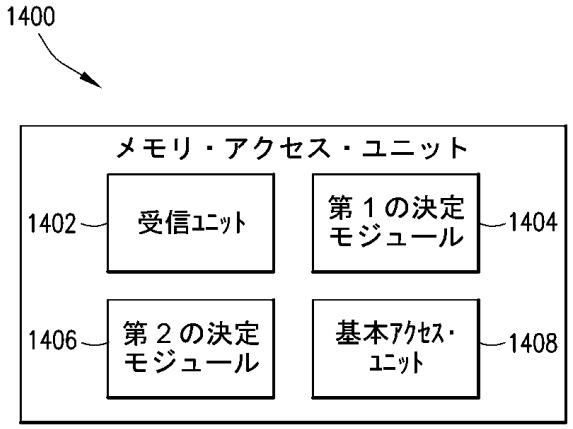
【図１１】



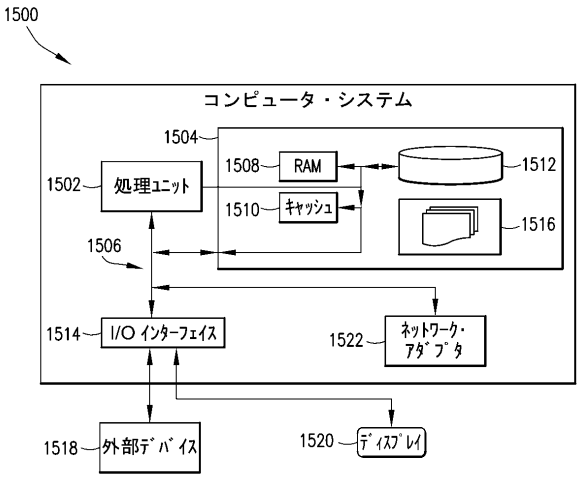
【図１２】



【図１３】


1400
メモリ・アクセス・ユニット
1402
受信ユニット
第１の決定モジュール
1404
1406
第２の決定モジュール
基本アクセス・ユニット
1408

【図１４】


1500
コンピュータ・システム
1504
1502
処理ユニット
1508
RAM
1512
1510
キャッシュ
1516
1506
1514
I/O インターフェイス
1522
ネットワーク・アダプタ
1518
外部デバイス
1520
ディスプレイ

フロントページの続き

(72)発明者　ファン ルンテレン、ヤン
スイス８８０３ リュシュリコン ゾイマーシュトラーセ ４

審査官　漆原 孝治

(56)参考文献　特開平０２－１２７７６８（ＪＰ，Ａ）
特開２００２－０７３４１２（ＪＰ，Ａ）
特開２０００－１６３３１６（ＪＰ，Ａ）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１７／１６