(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5507573号
(P5507573)
(45)発行日　平成26年5月28日(2014.5.28)　(24)登録日　平成26年3月28日(2014.3.28)

(51)Int.Cl.　　F I
G06F 12/08 (2006.01)　　G06F 12/08 507H
G06F 11/10 (2006.01)　　G06F 11/10 330K
　　G06F 12/08 541D

請求項の数 20　(全 13 頁)

(21)出願番号　特願2011-535800 (P2011-535800)
(86)(22)出願日　平成21年12月10日 (2009.12.10)
(65)公表番号　特表2012-508425 (P2012-508425A)
(43)公表日　平成24年4月5日 (2012.4.5)
(86)国際出願番号　PCT/US2009/067530
(87)国際公開番号　WO2010/077768
(87)国際公開日　平成22年7月8日 (2010.7.8)
審査請求日　平成23年5月9日 (2011.5.9)
(31)優先権主張番号　12/317,849
(32)優先日　平成20年12月29日 (2008.12.29)
(33)優先権主張国　米国 (US)

(73)特許権者　593096712
インテル　コーポレイション
アメリカ合衆国　９５０５４　カリフォルニア州　サンタ　クララ　ミッション　カレッジ　ブールバード　２２００
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介
(74)代理人　100107766
弁理士　伊東　忠重
(72)発明者　アガルワル，ラジャト
アメリカ合衆国　９７００６　オレゴン州　ビーヴァートン　ノースウェスト　トーキングスティック　ウェイ　１６５４８
最終頁に続く

(54)【発明の名称】ポイズン・ビット・エラー検査コード手法

(57)【特許請求の範囲】
【請求項１】
少なくとも一ビットのポイズン・ビットを生成するための装置であって、前記装置は、
共有メモリに記憶する対象のキャッシュ・ラインに対応するＥＣＣ（エラー訂正コード）を生成し、前記ＥＣＣはイベント・ビットを有し、
イベント判定を生成するようイベントの生起及び非生起の一方を判定し、
前記ＥＣＣの前記イベント・ビットが前記イベント判定に対応しているかに応じて前記ＥＣＣの１つ又は複数のビットのポイズン・ビットを生成し、
前記生成されたＥＣＣの１つ又は複数のビットのポイズン・ビット及び対応するキャッシュ・ラインをメモリに記憶する
ためのロジックを含み、
前記イベント・ビットは、前記キャッシュ・ラインに対応するデータを特定のプロセッサがキャッシュしている旨を示すための動的ビット位置、及び、複数のプロセッサの何れか１つが前記データをキャッシュしている旨を示すための静的ビット位置の一方に対応する装置。
【請求項２】
請求項１記載の装置であって、前記共有メモリは前記複数のプロセッサによって共有され、前記対応するキャッシュ・ラインは前記複数のプロセッサのうちの一プロセッサによって記憶されるよう要求され、前記イベントの生起及び非生起の一方を判定するためのロジックは、前記一プロセッサがそのキャッシュにおいて更にデータをキャッシュしたかを

判定するためのロジックを含む装置。
【請求項３】
請求項１記載の装置であって、前記ＥＣＣの前記イベント・ビットが前記イベント判定に対応しているかに応じて前記ＥＣＣの１つ又は複数のビットのポイズン・ビットを生成するためのロジックは、前記イベント・ビットが前記イベント判定に対応しているかを判定するためのロジックを備え、前記イベント・ビットが前記イベント判定に対応していない場合、前記イベント判定に対応するポイズン・ビットを生成するよう前記ＥＣＣをコード化するためのロジックを含む装置。
【請求項４】
請求項１記載の装置であって、
前記ＥＣＣの１つ又は複数のビットのポイズン・ビットを生成するためのロジックは、ポイズン・マスクを使用して前記ＥＣＣをコード化するためのロジックを含む装置。
【請求項５】
請求項４記載の装置であって、
前記メモリから前記ＥＣＣを読み出し、
前記読み出されたコードを処理し、
前記イベントの前記生起を判定する
ためのロジックを更に含む装置。
【請求項６】
請求項５記載の装置であって、前記読み出されたコードをデコードして、デコードされた読み出されたコードを生成し、前記デコードされた読み出されたコードに対してコード検査を行い、前記読み出されたコードに対してコード検査を行うためのロジックを更に含む装置。
【請求項７】
請求項６記載の装置であって、前記読み出されたコードをデコードして、デコードされた読み出されたコードを生成するためのロジックは、前記読み出されたコードに対して、前記ポイズン・ビットの逆関数を施す装置。
【請求項８】
請求項５記載の装置であって、前記イベントの前記生起を判定するためのロジックは、
前記イベント・ビットを読み出して、前記イベントの前記生起及び前記非生起のうちの一方を判定する
ためのロジックを含む装置。
【請求項９】
システムであって、
複数のプロセッサと、
前記複数のプロセッサに結合された共有メモリと、
前記複数のプロセッサに結合され、
共有メモリに記憶する対象のキャッシュ・ラインに対応するＥＣＣ（エラー訂正コード）を生成し、前記ＥＣＣはイベント・ビットを有し、
イベントの生起及び非生起の一方を判定して、イベント判定を生成し、
前記ＥＣＣの前記イベント・ビットが前記イベント判定に対応しているかに応じて前記ＥＣＣの１つ又は複数のビットのポイズン・ビットを生成し、
前記生成されたＥＣＣの１つ又は複数のビットのポイズン・ビット及び対応するキャッシュ・ラインを前記メモリに記憶する
ためのメモリ・コントローラ・ロジックを有する集積回路と
を備え、前記イベント・ビットは、前記キャッシュ・ラインに対応するデータを特定のプロセッサがキャッシュしている旨を示すための動的ビット位置、及び、複数のプロセッサの何れか１つが前記データをキャッシュしている旨を示すための静的ビット位置の一方に対応するシステム。
【請求項１０】

請求項９記載のシステムであって、前記対応するキャッシュ・ラインは前記複数のプロセッサのうちの一プロセッサによって記憶されるよう要求され、前記イベントの生起及び非生起の何れかを判定するための前記メモリ・コントローラ・ロジックは、前記一プロセッサがそのキャッシュにおいて更にデータをキャッシュしたかを判定するためのロジックを含むシステム。

【請求項１１】

請求項９記載のシステムであって、前記ＥＣＣの前記イベント・ビットが前記イベント判定に対応しているかに応じて前記ＥＣＣの１つ又は複数のビットのポイズン・ビットを生成するためのロジックは、前記イベント・ビットが前記イベント判定に対応しているかを判定するためのロジックを備え、前記イベント・ビットが前記イベント判定に対応していない場合、前記イベント判定に対応するポイズン・ビットを生成するよう前記ＥＣＣをコード化するためのロジックを含むシステム。

【請求項１２】

請求項９記載のシステムであって、前記メモリ・コントローラ・ロジックは更に、

前記メモリから前記ＥＣＣを読み出し、

前記イベント・ビットを読み出して、前記イベントの前記生起及び前記非生起のうちの一方を判定するシステム。

【請求項１３】

請求項１２記載のシステムであって、前記読み出されたコードをデコードして、デコードされた読み出されたコードを生成するためのロジックを更に含むシステム。

【請求項１４】

請求項１３記載のシステムであって、前記読み出されたコードに対して、前記ポイズン・ビットの逆関数を施すためのロジックを更に含むシステム。

【請求項１５】

方法であって、

イベントの生起及び非生起の一方を判定する工程であって、前記生起及び前記非生起の前記一方がイベント判定をもたらす工程と、

コードの１つ又は複数のビットのポイズン・ビットを生成する工程であって、前記コードはイベント・ビットを有し、前記生成する工程は、

前記イベント・ビットが前記イベント判定に対応しているかを判定し、

前記イベント・ビットが前記イベント判定に対応していない場合、前記イベント判定に対応するポイズン・ビットを生成するよう前記コードをコード化すること

により、前記判定及び前記コードに応じる工程と、

前記コードをメモリに記憶する工程と

を含み、前記イベント・ビットは、キャッシュ・ラインに対応するデータを特定のプロセッサがキャッシュしている旨を示すための動的ビット位置、及び、複数のプロセッサの何れか１つが前記データをキャッシュしている旨を示すための静的ビット位置の一方に対応する方法。

【請求項１６】

請求項１５記載の方法であって、

前記メモリから前記コードを読み出す工程と、

前記イベント・ビットを読み出して、前記イベントの前記生起及び前記非生起の一方を判定する工程と

を更に含む方法。

【請求項１７】

請求項１６記載の方法であって、

前記読み出されたコードをデコードする工程を更に含む方法。

【請求項１８】

命令を記憶させた記憶媒体であって、前記命令は、マシンによって実行されると、

イベントの生起及び非生起の一方を判定する工程であって、前記生起及び前記非生起の

前記一方がイベント判定をもたらす工程と、
コードの１つ又は複数のビットのポイズン・ビットを生成する工程であって、前記コードはイベント・ビットを有し、前記生成する工程は、
前記イベント・ビットが前記イベント判定に対応しているかを判定し、
前記イベント・ビットが前記イベント判定に対応していない場合、
前記イベント判定に対応するポイズン・ビットを生成するよう前記コードをコード化することにより、前記判定及び前記コードに応じる工程と、
前記コードをメモリに記憶させる工程と
を実行させ、前記イベント・ビットは、キャッシュ・ラインに対応するデータを特定のプロセッサがキャッシュしている旨を示すための動的ビット位置、及び、複数のプロセッサの何れか１つが前記データをキャッシュしている旨を示すための静的ビット位置の一方に対応する記憶媒体。
【請求項１９】
請求項１８記載の記憶媒体であって、マシンによって実行されると、
前記メモリから前記コードを読み出す工程と、
前記イベント・ビットを読み出して、前記イベントの前記生起及び前記非生起の一方を判定する工程と
を実行させる命令を更に含む記憶媒体。
【請求項２０】
請求項１９記載の記憶媒体であって、前記読み出されたコードを処理する工程を実行させる命令は、マシンによって実行されると、
前記読み出されたコードをデコードする工程を実行させる命令を含む記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明の実施例は、ポイズン・ビット・エラー検査コード手法に関する。
【背景技術】
【０００２】
データがメモリを出入りするにつれ、各種エラーが生じ得る。最も一般的には、これは単一ビットのエラーであるが、２ビットのエラーや他のタイプのエラーも生じ得る。エラー検査は通常、データがメモリに書き込まれ、メモリから読み出されるにつれ、データのインテグリティを検査するために使用される。エラー検査は、エラー検出及び／又はエラー訂正を含み得る。エラー検出は、伝送中に雑音又は他の障害によってもたらされるエラーの存在を検出することができることをいう。エラー訂正は、更に、エラーのない元のデータを再構成することができることをいう。利用可能な手法が多く存在している。
【発明の概要】
【発明が解決しようとする課題】
【０００３】
エラー訂正コード（ＥＣＣ）は、エラー検査手法、特に、エラー訂正の一例である。例えば、ＥＣＣでは、データがメモリ・バス上で送出されるにつれ、ＥＣＣ生成ロジック、例えば、排他的論理和アルゴリズムが、ＥＣＣを生成するためにデータにわたって算出される。データ及びＥＣＣは次いで、併せてメモリに記憶される。データが、その後、メモリから（ＥＣＣとともに）アクセスされると、エラー検出及び訂正情報を生成するためにＥＣＣエラー・デコード・ロジックがデータに施される。次いで、データは、この情報を使用して訂正することが可能である。
【課題を解決するための手段】
【０００４】
本発明の実施例は、添付図面の図において限定でなく、例として示し、同じ参照符号は同様の構成要素を表す。
【図面の簡単な説明】

【０００５】
【図１】本発明の実施例によるシステムを示す図である。
【図２】本発明の実施例による方法を示す図である。
【図３】本発明の実施例による方法を示す図である。
【図４】本発明の実施例による状態図である。
【発明を実施するための形態】
【０００６】
後述の例は、例証の目的のために過ぎず、決して、本発明の実施例を限定することを意図するものでない。よって、例を詳細に説明している場合、又は１つ若しくは複数の例を記載している場合、それらの例は網羅的であると解されるものでなく、説明し、かつ／又は例証している例に本発明の実施例が限定されるものでない。
【実施例】
【０００７】
図１は、実施例によるコンピュータ・システム１００を示すブロック図である。一部の実施例では、コンピュータ・システム１００は、（最大Ｎ個の）複数のプロセッサ１０２Ａ、１０２Ｂ、１０２Ｃを含み得る。本明細書及び特許請求の範囲では、プロセッサ１０２Ａ、１０２Ｂ、１０２Ｃの特定の何れか１つは、１０２Ｎとして表し得る。本明細書及び特許請求の範囲記載の「プロセッサ」は、計算タスクを実現するためのハードウェア及びソフトウェア・リソースの何れかの組み合わせに関する。例えば、プロセッサは、予め定義された命令セットにより、データを処理するためのマシン読み出し可能な命令を実行するために中央処理装置（ＣＰＵ）又はマイクロコントローラを含み得る。プロセッサは複数の処理コアを有するマルチコア・プロセッサを含み得、あるいは、プロセッサは、マルチコア・プロセッサに含み得る処理コア（オペレーティング・システムは処理コアを、実行リソースの完全なセットを備えた個別プロセッサとして認識し得る）を表し得る。他の可能性も存在している。
【０００８】
システム１００は更に、メモリ１０６を含み得る。メモリ１０６は更に、実行されることができるマシン実行可能な命令１３２、及び／又は、アクセスされ、処理され、かつ／若しくは操作されることができるデータを記憶し得る。本明細書及び特許請求の範囲記載の「マシン実行可能な」命令は、本明細書及び特許請求の範囲では、１つ又は複数の論理演算を行うために１つ又は複数のマシンによって理解され得る表現に関する。例えば、マシン実行可能な命令１３２は、１つ又は複数のデータ・オブジェクトに対して１つ又は複数の動作を実行するためにプロセッサ・コンパイラによって解釈可能な命令を含み得る。しかし、これは、マシン実行可能な命令の例に過ぎず、本発明の実施例はこの点に限定されない。メモリ１０６は、例えば、リード・オンリー、大容量記憶装置、ランダム・アクセス・メモリ、不揮発性メモリ、及び／又は１つ若しくは複数の他のタイプのマシン・アクセス可能なメモリを含み得る。局所キャッシュ１０４Ａ、１０４Ｂ、１０４Ｃ（プロセッサに対して局所のキャッシュ）を使用してメモリが共有され、同期化されるマルチプロセッサ・システムでは、メモリに記憶されたデータは、キャッシュ・ラインとして表し得る。
【０００９】
ロジック１３０は、システム１００の何れかの部分の上、又は前述の部分内に含まれ得る。ロジック１３０は、ハードウェア、ソフトウェア、又はハードウェア及びソフトウェアの組み合わせ（例えば、ファームウェア）を含み得る。例えば、ロジック１３０は、本明細書及び特許請求の範囲記載の動作を行うための回路（すなわち、１つ又は複数の回路）を含み得る。例えば、ロジック１３０は、１つ若しくは複数のディジタル回路、１つ若しくは複数のアナログ回路、１つ若しくは複数のステート・マシン、プログラム可能なロジック、及び／又は、１つ若しくは複数のＡＳＩＣ（特定用途向け集積回路）を含み得る。ロジック１３０は、１つ又は複数の動作を行うために配線し得る。あるいは、又は更には、ロジック１３０は、前述の動作を行うために、メモリ１０６などのメモリに記憶され

たマシン実行可能な命令１３２に実施し得る。あるいは、又は更に、ロジック１３０はファームウェアで実施し得る。ロジックは、システム１００の種々の構成部分に（メモリ・コントローラ１１４などに）含み得る。ロジック１３０は、メモリ・コントローラ１１４における個別ブロックとして示しているが、メモリ・コントローラ１１４及びロジック１３０は、そうでない場合には同じブロックに含まれ得る。一般に、ロジック１３０は、本明細書及び特許請求の範囲記載の種々の構成部分により、種々の機能を行うために使用し得る。

【００１０】

ハブ・コントローラ１０８は、メモリ１０６、並びに、処理コア１０２Ａ、１０２Ｂ及び１０２Ｃそれぞれを互いに結合し得るホスト・ブリッジ／ハブ・システムを含み得る。ハブ・コントローラ１０８は、１つ又は複数の集積回路チップ（インテル（登録商標）社から市場で入手可能な集積回路チップセット（例えば、グラフィクス、メモリ、及びＩ／Ｏコントローラ・ハブ・チップセット）から選択可能なものなど）を含み得るが、他の１つ又は複数の集積回路チップを更に、又は、あるいは使用し得る。例えば、ハブ・コントローラ１０８は、入出力制御ハブ（ＩＣＨ）、メモリ制御ハブ（ＭＣＨ）、又はグラフィクス／メモリ制御ハブ（ＧＭＣＨ）を含み得る。システム１００は、ＩＣＨ及びＭＣＨなどの２つのハブ・コントローラ、又は単一のハブ・コントローラのみを含み得るが、本発明の実施例はシステム１００内のハブ・コントローラの数に限定されない。

【００１１】

ハブ・コントローラ１０８は、メモリ・バス１１２を介してメモリ１０６と通信し、システム・バス１１０を介してプロセッサ１０２Ａ、１０２Ｂ、１０２Ｃと通信し得る。ハブ・コントローラ１０８は、システム１００内の他の装置（図示せず）、並びに、メモリ１０６、及びプロセッサ１０２Ａ、１０２Ｂ、１０Ｃ間のデータの移動を管理するためにメモリ・コントローラ１１４を含み得る。あるいは、メモリ・コントローラ１１４は、プロセッサ１０２Ａ、１０２Ｂ、１０２Ｃのうちの１つ又は複数とダイ上で一体化され得る。

【００１２】

システム１００は更に、ディレクトリ・キャッシュ１１６を含み得る。ディレクトリ・キャッシュ１１６は、キャッシュ・ライン、又はメモリのブロックそれぞれについての情報を保持し得る。例えば、ディレクトリ・キャッシュ１１６は、メモリのブロックそれぞれの状態（例えば、共有されている、キャッシュされていない、又は排他的である）、どの局所キャッシュ（プロセッサ１０２Ａ、１０２Ｂ、１０２Ｃのうちの１つに対応する）が特定のキャッシュ・ラインの複製を有しているか、及びキャッシュ・ラインがダーティであるか否かを示し得る。

【００１３】

例えば、一実施例では、データに対する要求は全て、ディレクトリ・キャッシュ１１６に送出される。ディレクトリ・キャッシュ１１６は次いで、その局所キャッシュにデータを記憶したとディレクトリ・キャッシュ１１６が示す何れかの数のプロセッサ１０２Ａ、１０２Ｂ、１０２Ｃに、データに対する要求を転送し得る。例として、ディレクトリ・キャッシュ１１６は、ＮノードについてＮビットのベクトル・マップを維持し得る。ここで、各ビットは、対応するノードが、特定のキャッシュ・ラインのキャッシュされた複製を維持しているか否かを示す。

【００１４】

プロセッサ１０２Ａ、１０２Ｂ、１０２Ｃ、メモリ１０６、及びバス１１０、１１２は、例えばシステム・マザーボード１１８などの単一の回路基板に含まれ得るが、本発明の実施例はこの点に限定されない。

【００１５】

図２は、本発明の実施例による方法を示す。図２の方法はブロック２００で始まり、ブロック２０２に続き得る。ブロック２０２では、方法は、イベントの生起及び非生起の何れかを判定し、生起及び非生起の何れかがイベント判定をもたらし得る。

【００１６】
イベントは例えば、データがキャッシュされていることを含み得る。例えば、一実施例では、プロセッサ１０２Ｎはデータを修正し、これをその局所キャッシュにキャッシュし、次いで、データをもう一度、メモリ１０６などのメモリに書き込み得る。この例では、データが局所キャッシュに記憶されるとイベントが生じたと判定し得る。別の例として、イベントは、メッセージが記憶されていることを含み得る。例えば、スーパーコンピュータなどの特定のシステムでは、システム内の種々のマイクロプロセッサ上で実行している処理はメッセージを互いに転送し得る。この例では、一処理が、別の処理のためのメッセージを生成し、記憶すると、イベントが生じたと判定し得る。
【００１７】
ブロック２０４では、方法は、イベント・ビットを有するコードを処理する工程を含み得、上記処理は、イベント判定及びコードに応じる。「イベント・ビット」は、イベントの生起又は非生起を示すようセットし得る、コード内のビットを表す。イベント・ビットは、静的なビット位置（例えば、ビット位置０）を含み得るか、又は、動的なビット位置（例えば、特定のプロセッサがイベントを生成した旨を示すためのビット位置３）を含み得る。よって、イベントが生じたと判定された場合、イベント・ビットは１つの値にセットされ、さもなければ、イベントが生じていないと判定された場合、イベント・ビットは別の値にセットされる。「セット」は、イベントの生起を示すために「１」にビットをセットし、又はイベントの非生起を示すために「０」にビットをセットすることを表し得る。あるいは、「セット」は、イベントの生起を示すために「０」にビットをセットし、又はイベントの非生起を示すために「１」にビットをセットすることを表し得る。
【００１８】
この意味合いで、「イベント判定及びコードに応じてコードを処理する工程」は、ブロック２０６において、コードのイベント・ビットがイベント判定に対応しているかを判定し、否定の場合、コードがコード化され、さもなければ、そのままの状態にされる工程を表す。イベント・ビットがイベント判定と矛盾する場合には、イベント・ビットはイベント判定に対応しない。例えば、イベント・ビットが値「０」（この値は、イベントがディセーブルされたか、又は生じていない旨を表すよう構成される）を有する場合であり、かつ、イベントが生じたと判定された場合、イベント・ビットはイベント判定に対応しない。
【００１９】
逆に、イベント・ビットが、イベント判定と矛盾しない、イベントの生起を示す場合、イベント・ビットはイベント判定に対応する。例えば、イベント・ビットが値「１」（この値は、イベントが生じた旨を表すよう構成される）を有する場合であり、かつ、イベントが生じたと判定された場合、イベント・ビットはイベント判定に対応する。
【００２０】
ブロック２０８では、イベント・ビットがイベント判定に対応しない場合、コードは、イベント判定に対応するポイズン・ビットを生成するようコード化される。本明細書及び特許請求の範囲記載の「コード化」は、コードの１つ又は複数のビットのポイズニングを表す。本明細書及び特許請求の範囲記載の「ポイズニング」は、下にあるコードの一般的な効果性を変えることなく、イベントの生起又は非生起を示すためのイベント・ビット位置における１ビット又は０ビットを記憶するようポイズン・マスク（例えば、固定パターン・コード）を施す工程を表す。更に、エラー検査ロジックが、処理済みの第１のコードを生成するようコードに施され、処理済みの第２のコードを生成するよう、デコードされたコードに施されると、処理済みの第１のコード及び処理済みの第２のコードのうちの一方のみが訂正可能でない。すなわち、更なるコード化を有するデータは、メモリからもう一度読み出された場合、更なるコード化なしでデータから、比較的簡単に、区別可能であるはずである。このことは、一般的なメモリ・エラーが存在している場合にも該当し得る。このことは図３で更に詳細に説明する。
【００２１】

本発明の実施例では、例えば、そのエラー検査及び／又は訂正機能が、コード化されていないエラー検査コードと精度がほぼ同じである場合、ポイズン・マスクは、下にあるエラー検査コードの一般的な効果性を変えない。更に、好ましくは、各種エラーに対するエイリアシングが全くないか、又はほとんどない。例えば、一般的に効果的なエラー検査コードは、同じメモリ装置において２ビットのエラーに対して、又は、１ビット若しくは１ワイヤのエラーに対してエイリアシングが生じないエラー検査コードである。ここで、サイレント・データ破損（ＳＤＣ）エラー及び検出された回復不可能なエラー（ＤＵＥ）は、統計的に稀に（例えば、他の２ビット・エラーの場合、２千４百万分の１、チップキルの場合、２万２千分の１）生じる。一実施例では、コード化は、単一のビットをポイズニングするためにポイズン・マスクを施す工程を含み得る。

【００２２】

当業者が分かるように、エラー検査コードをコード化するために使用されるロジックは、エラー検査コード自体に依存する。当業者が分かるように、ＥＣＣコードなどのエラー検査コードの生成は多くの場合、労働集約的な作業であり、機密の、又はトレード・シークレットのデータである。コード化が基礎とするコードの複雑性を前提とすれば、更なるコード化を行うために使用されるロジックは、数学的解析及び試行錯誤を必要とする。しかし、満たす対象の要件の説明は、しかしながら、本発明の実施例に応じてロジックを当業者が生成することを可能にする。

【００２３】

例えば、一実施例では、データがキャッシュされている場合（ビット１が、データがキャッシュされていることを示す場合）、エラー検査コード０１０１０１００１０１１０１０００（ここで、ビット位置０は、データがキャッシュされているか否かを示すためのイベントを含む）は、ビット０でなくビット１を記憶するようビット位置０をポイズニングすることにより、イベントに応じてコード化する必要がある。この例では、コード化されたエラー検査コードは０１０１０１００１０１１０１００１をもたらす。他方で、データがキャッシュされている場合（ビット１が、データがキャッシュされていることを示す場合）生成されたエラー検査コード０１０１０１００１０１１０１００１（ビット位置０は、データがキャッシュされているか否かを示すために使用される）は、コード化しなくてよい。

【００２４】

一実施例では、イベント・ビットは、特定のプロセッサ１０２Ｎが、キャッシュされたデータを有している旨を示すための動的なビット位置に対応する。例えば、８つのプロセッサ（プロセッサ０乃至７）を有しており、キャッシング・プロセッサ１０２Ｎがプロセッサ６であるシステムでは、イベント・ビットはビット＃７に対応する。よって、エラー検査コードが０１０１０１００１００１０１０００を含む場合、そのキャッシュにプロセッサ６がキャッシュされたデータを有する旨を示すようビットをセットすることは、エラー検査コード０１０１０１００１０１１０１０００をもたらし得る。同様に、プロセッサ＃２がキャッシング・プロセッサである場合、エラー検査コードはエラー検査コード０１０１０１００１００１０１１００をもたらし得る。別の実施例では、イベント・ビットは、プロセッサ１０２Ｎのうちの何れか１つによってデータがキャッシュされている旨を示すための静的ビットに対応する。例えば、エラー検査コードが０１０１０１００１００１０１０００を含み、イベント・ビットが、プロセッサが、キャッシュされたデータを有している旨を示すためのビット０を含む場合、エラー検査コードは、プロセッサ２がキャッシュ・プロセッサであるかプロセッサ７がキャッシュ・プロセッサであるかにかかわらず、０１０１０１００１００１０１００１を含む。

【００２５】

一実施例では、コードはエラー検査コードを含み得る。エラー検査コードは例えば、エラー訂正コード（ＥＣＣ）、又はＣＲＣ（巡回冗長度コード）を含み得る。しかし、本発明の実施例は前述の例に限定されない。

【００２６】

一実施例では、処理されたコードは、対応するデータとともにメモリに記憶されたＥＣＣコードを含み得る。例えば、図４に示すように、データ４０２のキャッシュ・ラインが、メモリ１０６への書き込みに利用可能な場合、ＥＣＣ生成ロジック４０４は、データ４０２の記憶中に生じ得る特定のエラー（例えば、１ビット・エラー又は２ビット・エラー）を訂正するためにメモリ１０６にデータ４０２とともに記憶されたＥＣＣ（イベント・ビットを有する）４０６を生成するようデータ４０２に適用し得る。上述の通り、かつ、４２８の決定ブロックで示すように、ＥＣＣ４０６のイベント・ビットがイベント判定に対応しない場合、ＥＣＣ４０６はポイズン・マスク４０８を使用してコード化される。ＥＣＣ４０６のイベント・ビットがイベント判定に対応する場合、（ブロック４０８でポイズン・マスクを施すことなく）処理済みのコード４１０がデータ４０２とともにメモリ１０６に記憶される。

【００２７】

図２の方法はブロック２０６で終了し得る。

【００２８】

一実施例では、キャッシュ・ラインは複数の部分に分けることができ、エラー検査コードは、キャッシュ・ラインの部分毎に生成し得る。例えば、キャッシュ・ラインは、半分の部分２つに分け得、エラー検査コードは、半分の部分毎に生成し得る。更に、エラー検査コードをコード化する工程は、単一のイベントを推論するためにキャッシュ・ライン部分の少なくとも１つに単一のマスクを施す工程を含み得る。あるいは、キャッシュ・ラインの各部分に対応するマスクは、複数のイベントを推論するよう施し得る。このことは、エラー訂正の機能の低下につながり得るが、更に、サイレント・データ破損（ＳＤＣ）及び／又は検出された回復不可能なエラー（ＤＵＥ）に対するエイリアシングの確率を低減し得る。

【００２９】

図２の方法により、エラー検査コードを記憶した後、方法は、図３の方法に更に続き得る。図３の方法は、ブロック３００で始まり、ブロック３０２に続く。ここで、方法は、メモリからコードを読み出す工程（「コードを読み出す工程」）を含み得る。一実施例では、図４を参照すれば、方法は、対応するデータ４０２とともに、メモリ１０６から、処理済みのＥＣＣコード４１０を読み出す工程を含み得る。

【００３０】

ブロック３０４では、方法は、処理済みの複数のコードを生成するよう、読み出されたコードを処理する工程を含み得る。

【００３１】

一実施例では、「処理済みの複数のコードを生成するよう読み出されたコードを処理する工程」は、デコードされた読み出されたコードを生成するよう、読み出されたコードをデコードする工程と、処理済みの第１のコードを生成するよう、読み出されたコードに対してコード検査を行う工程と、処理済みの第２のコードを生成するよう、デコードされた読み出されたコードに対してコード検査を行う工程とを表す。読み出されたコードは、メモリから読み出されたコード化されたコードを表す。デコードされた読み出されたコードは、コードをコード化するよう施されたコードの逆数を施すことから生じるコードを表す。

【００３２】

図４を参照すれば、一実施例では、データ４０２及び処理済みのＥＣＣ４１０は、メモリ１０６から読み出される。この点で、メモリ１０６から読み出されるＥＣＣ４１０は「読み出されたコード」として表す。読み出されたコードは次いで、デコードされた読み出されたコード４１８を生成するようデコードされる（４１４）。読み出されたコード４１２は、処理済みの第１のコード４２０を生成するためにコード検査ロジック４１６にわたって実行され、デコードされた読み出されたコード４１８は、処理済みの第２のコード４２２を生成するようコード検査ロジック４１６にわたって実行される。

【００３３】

コード検査ロジック４１６は、処理済みのコードによって生成し得る。例えば、処理済みのＥＣＣがポイズンされている場合（すなわち、ポイズン・マスクがＥＣＣに施された場合）、コード検査ロジック４１６が生成され、よって、処理済みのＥＣＣ４１０がメモリ１０６から読み出されると（読み出されたコード４１２）、デコードされた読み出されたコード４１８に対して算出されるコード検査ロジック４１６は、エラーを生じないか、又は訂正可能なエラーを生じ、読み出されたコード４１２に対して算出されるコード検査ロジック４１６は、訂正可能でないエラーを生じる。

【００３４】

同様に、処理済みのＥＣＣがポイズンされていない場合、コード検査ロジック２１６が生成され、よって、処理済みのＥＣＣ２１０がメモリ１０６から読み出されると（読み出されたコード２１２）、デコードされた読み出されたコード２１８にわたって算出されるコード検査ロジック２１６は訂正可能でないエラーを生じ、読み出されたコード２１２にわたって算出されるコード検査ロジック２１６は、エラーをもたらさないか、又は訂正可能なエラーを生じる。

【００３５】

やはり、更なるコード化を行うために使用されるロジックが、このようにしてエラー検査ロジック２１６を機能させるために数学的解析、及び試行錯誤を必要とし得る。

【００３６】

ブロック３０６では、方法は、処理された複数のコードのうちの１つから、訂正可能なコードを識別する工程を含み得る。本明細書及び特許請求の範囲記載の「訂正可能なコード」は、処理済みの複数のコードのうちの１つを表し、ここで、処理済みのコードのうちの１つのみが訂正可能でないコードであり、処理済みの他のコードは、ＥＣＣ検査の結果、訂正可能なエラーを有しているか、又はエラーを何ら有していない。処理済みのコードの両方／全てが、訂正可能でないエラーをもたらしたか、又は、処理済みのコードの両方／全てが、訂正可能なエラーをもたらしたか、又はエラーを何らもたらさなかった場合、エイリアシングが生じ、訂正可能なコードを識別することが可能でない。

【００３７】

表１は、訂正可能なコードを識別することが可能であるかを判定するために使用し得る決定テーブルの例を示す。この例では、マトリクスＡは読み出された（コード化された）コードに対して行われるＥＣＣ検査ロジックに表し、マトリクスＢはデコードされたコードに対して行われるＥＣＣ検査ロジックを表す。表の最上部における列は、読み出された（コード化された）コードに対してコード検査を行うことによって生成される処理済みの第１のコードの結果の値を表し、左の列は、デコードされたコードに対してコード検査を行うことによって生成される処理済みの第２のコードの結果を表す。表が示すように、コードは、第１の処理済みのコード及び第２の処理済みのコードが回復可能でないエラーを有し、第１の処理済みのコードにエラーがないか、又は第１の処理済みのコードが訂正可能なエラーを有するというシナリオにおいて訂正可能でない。他方で、第１の処理済みのコード及び第２の処理済みのコードの一方が、訂正可能でないエラーを有し、第１の処理済みのコード及び第２の処理済みのコードの他方にエラーがないか、上記他方が訂正可能なエラーを有する場合、訂正可能なコードが存在している。この後者のシナリオでは、訂正可能なコードは、エラーがないか、又は訂正可能なエラーを有する処理済みのコードである。

【００３８】

【表１】

マトリクスA → ↓ マトリクスB	エラーなし	訂正可能なエラー	訂正可能でないエラー
エラーなし	訂正可能でないコード	訂正可能でないコード	マトリクスB
訂正可能なエラー	訂正可能でないコード	訂正可能でないコード	マトリクスB
訂正可能でないエラー	マトリクスA	マトリクスA	訂正可能でないコード

図４を参照すれば、第１の処理済みのコード４２０及び第２の処理済みのコード４２２のうちの一方が訂正可能でなく、第１の処理済みのコード４２０及び第２の処理済みのコード４２２のうちの他方が訂正可能である（か、又はエラーがない）場合、訂正可能なコード４２６は例えば、選択ロジック４２４によって識別し得る。

【００３９】

ブロック３０８では、方法は、訂正可能なコードが識別された場合、イベントの生起及び非生起の一方を判定するよう訂正可能なコードのイベント・ビットを読み出す工程を含み得る。訂正可能なコードが識別されると、そのコードのイベント・ビットは、イベントが生じたか否かを判定するために読み出し得る。例えば、ビット値０がイベントの非生起を表し、ビット値１がイベントの生起を表す場合、訂正可能なコードのイベント・ビットにおけるビット値０は、イベントが生じなかった旨を示す一方、訂正可能なコードのイベント・ビットにおけるビット値１はイベントが生じた旨を示す。

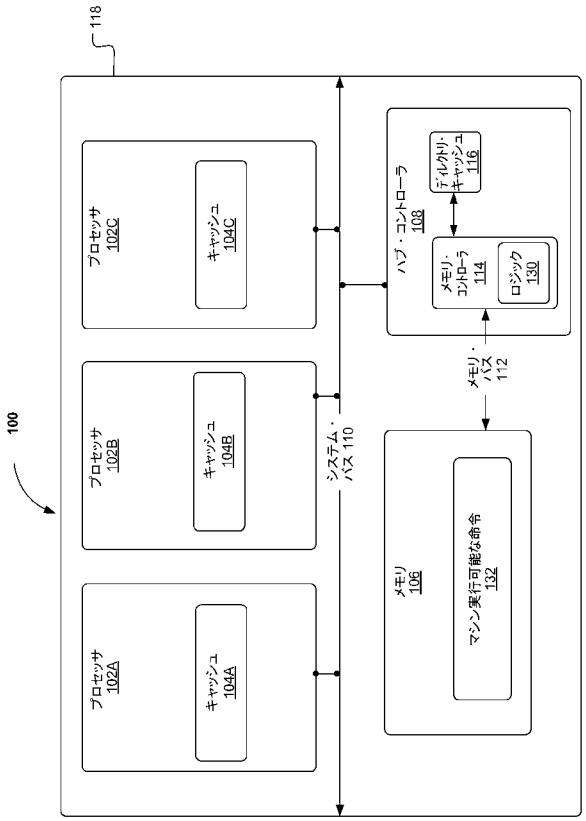
【００４０】

方法はブロック３１０で終了し得る。

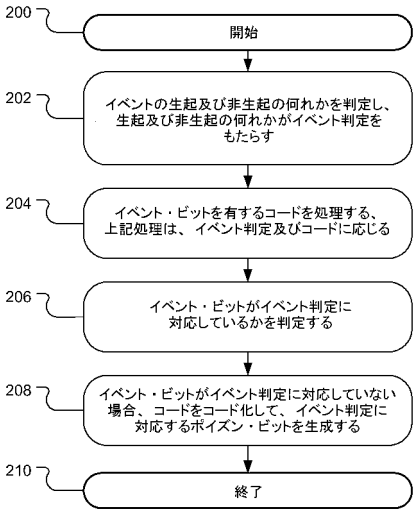
【００４１】

以下の明細書では、本発明は、その具体的な実施例を参照して説明している。しかし、それから逸脱しない限り、種々の修正及び変更を前述の実施例に対して行い得るということは明らかとなろう。よって、本願の明細書及び図面は、限定的な意味合いでなく例証的な意味合いで解されるものとする。

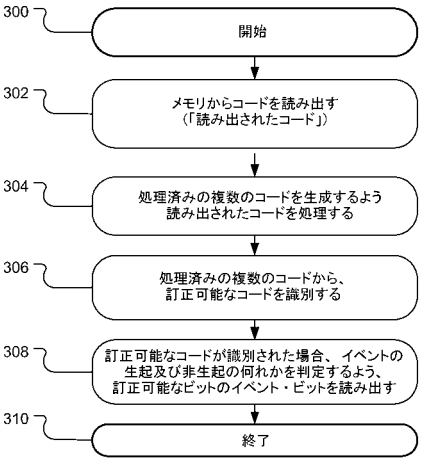
【図１】


118
100
プロセッサ 102C
キャッシュ 104C
プロセッサ 102B
キャッシュ 104B
プロセッサ 102A
キャッシュ 104A
システム・バス 110
ハブ・コントローラ 108
ディレクトリ・キャッシュ 116
メモリ・コントローラ 114
ロジック 130
メモリ・バス 112
メモリ 106
マシン実行可能な命令 132

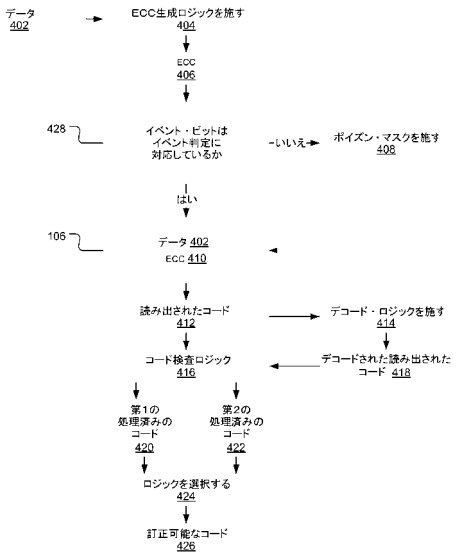
【図２】


200
開始
202
イベントの生起及び非生起の何れかを判定し、生起及び非生起の何れかがイベント判定をもたらす
204
イベント・ビットを有するコードを処理する、上記処理は、イベント判定及びコードに応じる
206
イベント・ビットがイベント判定に対応しているかを判定する
208
イベント・ビットがイベント判定に対応していない場合、コードをコード化して、イベント判定に対応するポイズン・ビットを生成する
210
終了

【図３】


300
開始
302
メモリからコードを読み出す（「読み出されたコード」）
304
処理済みの複数のコードを生成するよう読み出されたコードを処理する
306
処理済みの複数のコードから、訂正可能なコードを識別する
308
訂正可能なコードが識別された場合、イベントの生起及び非生起の何れかを判定するよう、訂正可能なビットのイベント・ビットを読み出す
310
終了

【図４】


データ 402
ECC生成ロジックを施す 404
ECC 406
428
イベント・ビットはイベント判定に対応しているか
いいえ
ポイズン・マスクを施す 406
はい
106
データ 402
ECC 410
読み出されたコード 412
デコード・ロジックを施す 414
コード検査ロジック 416
デコードされた読み出されたコード 418
第1の処理済みのコード 420
第2の処理済みのコード 422
ロジックを選択する 424
訂正可能なコード 426

フロントページの続き

(72)発明者　ハドルストン，スコット
アメリカ合衆国　９７００８　オレゴン州　ビーヴァートン　サウスウェスト　ハーネス　レーン　１３８５０
(72)発明者　ブルゼジンスキ，デニス
アメリカ合衆国　９４０８７　カリフォルニア州　サニーヴェイル　ブルックライン　ドライヴ　８２４

審査官　桜井　茂行

(56)参考文献　特開２００５－０７１２２４（ＪＰ，Ａ）
特開２００８－０４８２７８（ＪＰ，Ａ）
特開平１０－０５０００４（ＪＰ，Ａ）
特開２０００－３２２３１７（ＪＰ，Ａ）
特開平１１－００３２８８（ＪＰ，Ａ）
特開平０３－０２７４３３（ＪＰ，Ａ）
特開２００３－２１６５９６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／０８－１２／１２
Ｇ０６Ｆ　１１／１０