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

特許第5960517号
(P5960517)

(45)発行日　平成28年8月2日(2016.8.2)　(24)登録日　平成28年7月1日(2016.7.1)

(51)Int.Cl.		F I		
G06F 12/02	(2006.01)	G06F	12/02	510A
G06F 12/00	(2006.01)	G06F	12/00	560A
		G06F	12/00	597U

請求項の数 17　外国語出願　(全 37 頁)

(21)出願番号	特願2012-138094 (P2012-138094)
(22)出願日	平成24年6月19日 (2012.6.19)
(65)公開番号	特開2013-25793 (P2013-25793A)
(43)公開日	平成25年2月4日 (2013.2.4)
審査請求日	平成27年5月14日 (2015.5.14)
(31)優先権主張番号	61/507,659
(32)優先日	平成23年7月14日 (2011.7.14)
(33)優先権主張国	米国 (US)
(31)優先権主張番号	13/334,599
(32)優先日	平成23年12月22日 (2011.12.22)
(33)優先権主張国	米国 (US)

(73)特許権者　506076606
アバゴ・テクノロジーズ・ジェネラル・アイピー（シンガポール）プライベート・リミテッド
シンガポール国シンガポール７６８９２３，イーシュン・アベニュー・７・ナンバー１
(74)代理人　100087642
弁理士　古谷　聡
(74)代理人　100121061
弁理士　西山　清春

最終頁に続く

(54)【発明の名称】フラッシュメディアコントローラの内部のメタデータハンドリング

(57)【特許請求の範囲】

【請求項１】

フラッシュメディアコントローラ内のフラッシュメモリのページに格納されたメタデータを処理するための方法であって、

コンテキストごとに前記メタデータを定義するステップであって、該コンテキストはページごとに定義される、ステップと、

前記メタデータのサイズが定義済みの閾値と同じか該閾値よりも小さいときには、前記コンテキストの構造（以下、コンテキストの構造をコンテキスト構造という）内に完了したメタデータを格納するステップと、

前記メタデータのサイズが前記定義済みの閾値より大きいときには、前記コンテキスト内にメタデータポインタを定義するステップ

を含む方法。

【請求項２】

誤り訂正コードを用いて前記メタデータを保護するステップを更に含む、請求項１に記載の方法。

【請求項３】

前記メタデータは管理データを含む、請求項１に記載の方法。

【請求項４】

前記定義済みの閾値は、ページ当たりのバイト数を指定する、請求項１に記載の方法。

【請求項５】

フラッシュプログラムサイクルにおいて、前記コンテキスト構造からの前記メタデータは、フラッシュターゲットに格納され、読み取りサイクルにおいて、前記フラッシュターゲットから読み取られたメタデータは、前記コンテキスト構造に格納されることからなる、請求項１に記載の方法。

【請求項６】

前記メタデータポインタは、フラッシュプログラムサイクル中に外部システムメモリからメタデータを取り出すために、及び、前記コンテキストによって指定されたメタデータポインタによって指示される前記外部システムメモリ内の位置にフラッシュ読み取りコマンドからのメタデータを格納するために、アドレスポインタを提供することからなる、請求項１に記載の方法。

【請求項７】

フラッシュプログラムサイクルにおいて、フラッシュターゲットに格納されるメタデータの第１の部分は前記コンテキスト構造から得られ、フラッシュターゲットに格納される前記メタデータの第２の部分は、前記コンテキストによって指定されたメタデータポインタによって指示される外部システムメモリ内の位置から得られ、

読み取りサイクルにおいて、前記フラッシュターゲットから読み取られた前記メタデータの前記第１の部分は、前記コンテキスト構造に格納され、前記第２の部分は、前記コンテキストによって指定されたメタデータポインタによって指示される前記外部システムメモリ内の位置に格納されることからなる、請求項１に記載の方法。

【請求項８】

ページサイズがホストサイズの倍数であるときには、前記メタデータは複数のホストユーザデータセクタのそれぞれに分散される、請求項１に記載の方法。

【請求項９】

複数のフラッシュメディアデバイスを備えるフラッシュメモリであって、複数のページとして編成されるフラッシュメモリと、

前記フラッシュメモリの１以上のページにメタデータを格納するように構成されたフラッシュメディアコントローラ

を備える装置であって、

前記フラッシュメディアコントローラは、

(i)ページごとにコンテキストを定義し、及び、コンテキストごとに前記メタデータを定義し、

(ii)前記メタデータのサイズが定義済みの閾値と同じか該閾値よりも小さいときには、前記コンテキストの構造（以下、コンテキストの構造をコンテキスト構造という）内に完了したメタデータを格納し、

(iii)前記メタデータのサイズが前記定義済みの閾値より大きいときには、前記コンテキスト内にメタデータポインタを定義する

ことからなる、装置。

【請求項１０】

前記フラッシュメディアコントローラは、誤り訂正コードを用いて前記メタデータを保護する、請求項９に記載の装置。

【請求項１１】

前記メタデータは管理データを含む、請求項９に記載の装置。

【請求項１２】

メタデータのサイズが前記定義済みの閾値と同じか該閾値よりも小さい場合には、

フラッシュプログラムサイクルにおいて、前記フラッシュメディアコントローラは、フラッシュターゲットに前記コンテキスト構造からのメタデータを格納し、

読み取りサイクルにおいて、前記フラッシュメディアコントローラは、前記フラッシュターゲットから読み取られたメタデータを前記コンテキスト構造に格納する

ことからなる、請求項９に記載の装置。

【請求項１３】

前記定義済みの閾値は、ページ当たりのバイト数を指定する、請求項９に記載の装置。
【請求項１４】
前記メタデータポインタは、フラッシュプログラムサイクル中に外部システムメモリからメタデータを取り出すために、及び、前記コンテキストによって指定されたメタデータポインタによって指示される前記外部システムメモリ内の位置にフラッシュ読み取りコマンドからのメタデータを格納するために、前記フラッシュメディアコントローラによって使用されるアドレスポインタを提供する、請求項９に記載の装置。
【請求項１５】
フラッシュプログラムサイクルにおいて、フラッシュターゲットに格納されるメタデータの第１の部分は、前記コンテキスト構造から得られ、フラッシュターゲットに格納される前記メタデータの第２の部分は、前記コンテキストによって指定されたメタデータポインタによって指示される外部システムメモリ内の位置から得られ、
読み取りサイクルにおいて、前記フラッシュターゲットから読み取られたメタデータの前記第１の部分は、前記コンテキスト構造に格納され、前記第２の部分は、前記コンテキストによって指定されたメタデータポインタによって指示される前記外部システムメモリ内の位置に格納される
ことからなる、請求項９に記載の装置。
【請求項１６】
前記フラッシュメディアコントローラは、前記フラッシュメモリのページサイズがホストサイズの倍数であるときには、複数のホストユーザデータセクタのそれぞれにメタデータを分散するように更に構成される、請求項９に記載の装置。
【請求項１７】
コンテキストごとにメタデータを定義するための手段であって、該コンテキストはページごとに定義されることからなる、手段と、
前記メタデータのサイズが定義済みの閾値と同じか該閾値より小さい場合には、前記コンテキストの構造内に完了したメタデータを格納するための手段と、
前記メタデータのサイズが前記定義済みの閾値より大きい場合には、前記コンテキスト内にメタデータポインタを定義するための手段
を備えるフラッシュメモリシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本願は、２０１１年7月１４日に出願された米国仮特許出願第６１／５０７，６５９号明細書の利益を主張し、その全体が参照によって援用される。
【０００２】
本願は、同時係属中の出願である、２０１１年１２月２１日に出願された米国特許出願第１３／３３２，８４９号に関連し、その全体が参照によって援用される。
【０００３】
本発明は、一般に、フラッシュメディアシステムに係り、特に、フラッシュメディアコントローラの内部のメタデータをハンドリングするための方法及び／又は装置に関する。
【背景技術】
【０００４】
フラッシュメモリシステムは、機械的な遅延をハードドライブと関連付けないので、フラッシュメモリは、マスストレージ環境において魅力的である。従って、フラッシュメモリシステムは、より高い性能、対応してより低いコスト、電力、廃熱、及びスペース使用を許可する。しかしながら、フラッシュメモリは、一定の技術的な制限のためにそのような環境において伝統的に用いられていない。第１の技術的問題は、機械的なハードドライブのシーケンシャルアクセスの１０分の１と遅い書き込み速度である。より遅い書き込み速度は、書き込みに先立って長い消去サイクルなしにＮＡＮＤフラッシュデバイスでデータの上書きをすることができないという実情による。消去サイクルが直接に書き込みの性

能に影響を与えるので、大部分のフラッシュ設計は新規の位置に書き込みデータを移動させて、後まで消去を遅らせる。使用中のシステムでは、プロセッサが使用していないフラッシュページを消費し、新規のそれを作成するために停止しなければならないまで、遅れた消去サイクルは増すことができ、著しくシステム性能に影響を与える。第２の技術的問題は、シングルレベルセル（「ＳＬＣ」）デバイスのための１００，０００の消去サイクル、及びマルチレベルセル（「ＭＬＣ」）デバイスのための１０，０００のサイクルのフラッシュメモリページのそれぞれのための指定限界である。消去サイクルの限定された数は、予測不能のデータストリームが複数の消去に従うメモリの一定の高使用エリアにおいて生じるデータセンターのための特定の問題をもたらす。第３の問題は、データ損失である。データ損失は、妨害されたセルに隣接するメモリセルの読み取り又は書き込みによって、データの損失ビットを招くことにつながる読み取り妨害又はプログラム妨害を含むフラッシュメモリに影響を与える様々なファクタの結果として生じる場合がある。フラッシュメモリセルの状態は、時間の経過の結果として予測不能の方式においても変化する。

【０００５】

フラッシュ技術では、フラッシュ管理機能はファームウェアにおいて実現される。フラッシュ管理機能は、フラッシュバッファ管理、欠陥管理、及びウェアレベリングを含む。全ての管理機能は、ファームウェアによって利用される暫定的なデータ又は他の情報の格納のためにフラッシュデバイスのいくらかのストレージを利用する。フラッシュページに格納されるファームウェアによって利用される暫定的なデータ及び情報は、一般に、ここにメタデータと指称される。

【発明の概要】

【発明が解決しようとする課題】

【０００６】

フラッシュメディアコントローラの内部のメタデータをハンドリングするための方法及び／又は装置を実現することは望ましいだろう。

【課題を解決するための手段】

【０００７】

本発明は、フラッシュメディアコントローラの内部のフラッシュメモリのページに格納されたメタデータをハンドリングする方法に関する。方法は、一般に、（ｉ）ページベースごとにコンテキストが定義されるコンテキストベースごとにメタデータを定義し、（ｉｉ）メタデータのサイズが定義済みの閾値以下であるとき、コンテキストの構造の内部に完了したメタデータを格納し、及び（ｉｉｉ）メタデータのサイズが定義済みの閾値より大きいとき、コンテキストの内部にメタデータポインタを定義することを含む。

【０００８】

本発明の目的、特徴、及び利点は、（ｉ）ページベースごとにコンテキストが定義されるコンテキストベースごとにメタデータを定義し、（ｉｉ）メタデータサイズが定義済みの閾値以下であるとき、コンテキスト構造に完了したメタデータ情報を格納し、（ｉｉｉ）メタデータサイズが定義済みの閾値を超えるとき、コンテキストの内部のメタデータにポインタを定義し、（ｉｖ）ホストユーザデータを有するメタデータを分散し、及び／又は（ｖ）誤り訂正符号化、完全性チェッキング、及び訂正を有するメタデータを保護するフラッシュメディアコントローラの内部のメタデータをハンドリングするための方法及び／又は装置を提供することを含む。

【０００９】

本発明のこれら及び他の目的、特徴、及び利点は、以下の明細書、添付された特許請求の範囲、及び図面から明らかになる。

【図面の簡単な説明】

【００１０】

【図１】チップ（ＳＯＣ）コンテキストのシステムにおいて実現されたフラッシュメディアコントローラを図示するブロック図である。

【図２】本発明の実施の形態によるフラッシュメディアコントローラ（ＦＭＣ）アーキテ

クチャの一例を図示するブロック図である。
【図３】本発明の実施の形態によるフラッシュレーンコントローラアーキテクチャの一例を図示するブロック図である。
【図４】図３のコンテキストマネージャモジュールのサブモジュールの一例を図示する図である。
【図５】図３のダイ管理モジュールのサブモジュールの一例を図示する図である。
【図６】図３のフラッシュ動作マネージャモジュールのサブモジュールの一例を図示する図である。
【図７】図３のデータフローマネージャモジュールのサブモジュールの一例を図示する図である。
【図８】図３のコンテキストマネージャモジュールの実施の形態の一例を図示する図である。
【図９】図３のフラッシュ動作マネージャの実施の形態の一例を図示する図である。
【図１０】本発明の実施の形態によるフラッシュメディアコンテキストレイアウトの一例を図示する図である。
【図１１】本発明の実施の形態によるフラッシュページのリージョンパーティションの一例を図示する図である。
【図１２Ａ】本発明の実施の形態によるＦＭＣフラッシュページ構造の一例を図示する図である。
【図１２Ｂ】本発明の実施の形態によるＦＭＣフラッシュページ構造の一例を図示する図である。
【図１３】本発明の実施の形態による処理を図示する流れ図である。
【発明を実施するための形態】
【００１１】
１つの実施の形態では、本発明によるシステムは、小型コンピュータシステムインタフェース（「ＳＣＳＩ」）に基づいた、ＳＡＳ（「シリアルアタッチドＳＣＳＩ」）、ＦＣ（「ファイバチャンネル」）、及びＦＣ－ＡＬ（「ファイバチャネルアービトレーテッドループ）を含む様々なマスストレージプロトコル、並びにシリアルＡＴＡ（「ＳＡＴＡ」）プロトコルで動作するように設計される。当業者はマスストレージプロトコルに精通しているため、これらプロトコルはここで更に説明されない。特定のプロトコルがどこで呼び出されるかを除いて、ここに開示されたシステムと方法は、用いられる特定のプロトコルに依存せず、プロトコルの全てで正確に動作するように設計される。更に、本発明の実施の形態によるシステム及び方法は、エンドユーザ等の他のアプリケーションのためのプロトコルと同様にエンタープライズレベルのアプリケーションのためのプロトコルを含む現在利用しているか又は開発されているいずれかの他の同様のプロトコルの利用のために適しているだろう。ここに説明されたシステムは、フラッシュページ（例えば、メタデータ）に格納されるファームウェアによって利用される暫定的なデータ及び情報等のファームウェアデータのハンドリングのための新規な方法及び／又はアーキテクチャを含む。
【００１２】
図１を参照すると、本発明の実施の形態によるフラッシュメディアコントローラを有して実現されたシステム１００のブロック図が示される。一例では、システム（又はアーキテクチャ）１００は、ブロック（又は回路）１０２、複数のブロック（又は回路）１０４ａ～１０４ｎ、複数のブロック（又は回路）１０６ａ～１０６ｎ、ブロック（又は回路）１０８、ブロック（又は回路）１１０、ブロック（又は回路）１１２、ブロック（又は回路）１１４、及びブロック（又は回路）１１６を備える。回路１０２～１１６は、ハードウェア、ファームウェア、ソフトウェア、ハードウェアの組み合わせ、ファームウェア、及び／又はソフトウェア、又は他の実装として実現されたモジュール及び／又はブロックを表わす。
【００１３】
一例では、ブロック１０２は、本発明の実施の形態によるフラッシュメディアコントロ

ーラ（ＦＭＣ）を実現する。ブロック１０４ａ～１０４ｎは、フラッシュストレージデバイス又はコンポーネントの第１の数として実現される。ブロック１０４ａ～１０４ｎは、ブロック１０２の第１のフラッシュレーンに連結される。ブロック１０２の第１のフラッシュレーンは、ブロック１０４ａ～１０４ｎのそれぞれに独立したチップイネーブル（ＣＥ）信号を提供するように構成される。ブロック１０６ａ～１０６ｎは、フラッシュストレージデバイス又はコンポーネントの第２の数として実現される。ブロック１０６ａ～１０６ｎは、ブロック１０２の第２のフラッシュレーンに連結される。ブロック１０２の第２のフラッシュレーンは、ブロック１０６ａ～１０６ｎのそれぞれに独立したチップイネーブル（ＣＥ）信号を提供するように構成される。ＦＭＣ１０２は、２つのフラッシュレーンインスタンスで図示されるが、特定の実施の形態のその設計基準を満たすために、より多くのフラッシュレーンが実現されるであろうことは当業者に明らかである。フラッシュコンポーネント１０４ａ～１０４ｎ及び１０６ａ～１０６ｎは、１つ以上のダイを備える単一のフラッシュパッケージとして実現される。フラッシュコンポーネント１０４ａ～１０４ｎ及び１０６ａ～１０６ｎは、ＮＡＮＤ及び／又はＮＯＲフラッシュデバイスを用いて実現される。ブロック１０２は、ＮＡＮＤフラッシュ及び／又はＮＯＲフラッシュのための適切なフィジカルレイヤサポート（ＰＨＹ）を含む。

【００１４】

ブロック１０８は、ブロック１０２に連結される外部のＦＭＣプロセッサ（ＦＡＲＭ）を実現する。ブロック１１０は、ブロック１０２に静的ランダムアクセスメモリ（ＳＲＡＭ）及び／又は動的ランダムアクセスメモリ（ＤＲＡＭ）を連結するように構成されるメモリコントローラを実現する。ブロック１１２は、１つ以上のＳＲＡＭデバイスとして実現される。ブロック１１４は、１つ以上のＤＲＡＭデバイスとして実現される。ブロック１１６は、ブロック１１０及びブロック１１４を連結するダブルデータレートフィジカルレイヤ（ＰＨＹ）インタフェースを実現する。一例では、ブロック１０２、１０８、１１０、１１２、１１４及び１１６は、チップ（ＳＯＣ）アーキテクチャのシステムを実現する。

【００１５】

ブロック１０２は、様々なアプリケーションがフラッシュデバイス１０４ａ～１０４ｎ及び１０６ａ～１０６ｎを用いるのを支援するように構成されたソフトＩＰブロックとして実現される。ここに用いられたように、用語のソフトＩＰブロックは、一般に、ソフトウェア（例えば、ＨＤＬコード、ＲＴＬコード等）で提供される集積回路のビルディングブロックを表わす。ブロック１０２は、一般に、フラッシュデバイスを有する複数のフラッシュインタフェースをサポートする。ブロック１０２は、一般に、プロセッサ（例えば、ＡＲＭ）を含まない。しかしながら、ブロック１０２は、一例では、外部プロセッサ１０８にブロック１０２を連結するように構成されたインタフェース（例えば、３２ビットＡＨＢ等）を実現する。ブロック１０２は、一般に、ブロック１０４ａ～１０４ｎ及び１０６ａ～１０６ｎによって構成されたフラッシュメディアマスストレージアレイの管理をハンドリングするように構成される。一例では、ブロック１０２は、付属する複数の独立したフラッシュコンポーネントを有する単一のフラッシュデータレーンに関連する大部分の管理機能を実行する多数例示されたフラッシュレーンコントローラ（ＦＬＣ）を利用する。ブロック１０２の機能は、ブロック１０２がフラッシュアクセスに関してほとんど理解しないという意味ではいくぶん一般的である。ブロック１０２は、一般に、フラッシュアウェアレーンをまとめ上げて単一のハードウェアエンティティを作ることに関する。一例では、ブロック１０２を実現するソフトＩＰは、アプリケーションのために出来る限り最大のレーンをサポートするためにパラメータ化する。例えば、１つの実施の形態では、レーンの数は２である。別の実施の形態では、その数は８である。

【００１６】

一例では、ブロック１０２は、以下に含まれる特徴をサポートする：（ｉ）２つのフラッシュレーン；（ｉｉ）フラッシュレーンのそれぞれでの８つまでのチップイネーブル信号（ＣＥ）；（ｉｉｉ）非同期ノーマルモード、非同期拡張モード、トグル１．０、ＯＮ

ＦＩ２．１、ＯＮＦＩ２．３、及びトグル２．０を含むフラッシュインタフェース；（ｉｖ）専門のＥＣＣ又は複数のレーンの間で共有されたＥＣＣは、設定可能なハードウェアである（例えば、ブロック１０２を実現するソフトＩＰブロックの特徴をパラメータ化する）；（ｖ）フラッシュインタフェースでの８ビットデータ；（ｖｉ）トグル２．０でのフラッシュインタフェース又はＯＮＦＩ２．３でのフラッシュインタフェースの仕様の２００ＭＨｚまでのＤＤＲレート；（ｖｉｉ）部分的な読み取りコマンド、（ｖｉｉｉ）ランダム読み取りコマンド；（ｉｘ）フラッシュ書き込み／読み取りのＣＲＣストリップ／インサートオプション；（ｘ）４Ｋバイトデータのための６４ビットまでの訂正；（ｘｉ）５１２、２Ｋ、４Ｋバイトデータでの設定可能なｎビット訂正（最大ｎ＝６４）；（ｘｉｉ）レジスタプログラミングのための３２ビットＡＨＢインタフェース；（ｘｉｉｉ）外部メモリ（例えば、ＤＲＡＭ又はＳＲＡＭ）でのコンテキストコマンドの記憶；（ｘｉｖ）フラッシュレーンコントローラのカットスルーバッファ；（ｘｖ）優れたパフォーマンスを提供するための独立したフラッシュ読み書きデータパス；（ｘｖｉ）フラッシュユニットナンバー（ＦＵＮ）ごとにインオーダ状態の伝達；（ｘｖｉｉ）フラッシュレーンごとにデータパスのための１つの読み取り及び１つの書き込みバッファコントローラ（ＢＣ）インタフェースのサポート；（ｘｖｉｉｉ）コンテキスト訂正のための読み取りＢＣインタフェースのサポート；（ｘｉｘ）コンテキストアップデートのための書き込みＢＣインタフェースのサポート；（ｘｘ）コンテキストフリーリソースポインタ（ＣＦＲＰ）のための読み取り／書き込みＢＣインタフェースのサポート。

【００１７】

図２を参照すると、本発明の実施の形態によるフラッシュメディアコントローラ（ＦＭＣ）アーキテクチャの一例を図示する図１のブロック１０２のより多くの個別ブロック線図が示される。一例では、ブロック１０２は、バッファコントローラ（ＢＣ）インタフェース、フラッシュデバイスインタフェース、及びプロセッサインタフェース（例えば、３２ビットＡＨＢ等）の三大機能インタフェースを実現する。バッファコントローラ（ＢＣ）インタフェースは、ブロック図の左側及び左上に図示される。一例では、７つのバッファコントローラインタフェース（例えば、３つの読み取りインタフェースＢＣ＿ＲＤ＿Ｉ／Ｆ、３つの書き込みインタフェースＢＣ＿ＷＲ＿Ｉ／Ｆ、及び１つの読み取り／書き込みインタフェースＢＣ＿ＲＤ／ＷＲ＿Ｉ／Ｆ）が実現される。フラッシュデバイスインタフェースは、ブロック図の右側に図示される。一例では、２つのフラッシュレーンインタフェース（例えば、ＦＬＡＳＨ＿Ｉ／Ｆ＿０及びＦＬＡＳＨ＿Ｉ／Ｆ＿１）が実現される。３２ビットＡＨＢインタフェースは、ブロック図の右上に図示される。３２ビットＡＨＢインタフェースは、一例では、レジスタのプログラム、状態の読み取り、及びブロック１０２の内部のレジスタの診断に用いられる。

【００１８】

ブロック１０２は、一般に、ブロック（又は回路）１５０、ブロック（又は回路）１５２、複数のブロック（又は回路）１５４ａ～１５４ｎ、複数のブロック（又は回路）１５６ａ～１５６ｎ、複数のブロック（又は回路）１５８ａ～１５８ｎ、ブロック（又は回路）１６０、ブロック（又は回路）１６２、ブロック（又は回路）１６４、ブロック（又は回路）１６６、ブロック（又は回路）１６８、ブロック（又は回路）１７０、複数のブロック（又は回路）１７２ａ～１７２ｎ、及び複数のブロック（又は回路）１７４ａ～１７４ｎを備える。回路１５０から１７４ａ～１７４ｎは、ハードウェア、ファームウェア、ソフトウェア、ハードウェアの組み合わせ、ファームウェア、及び／又はソフトウェア、又は他の実装として実現されるモジュール及び／又はブロックを表わす。ブロック１５０は、プロセッサインタフェースロジック（ＰＩＬ）を実現する。ブロック１５２は、データＤＭＡマネージャ（ＤＤＭ）を実現する。ブロック１５４ａ～１５４ｎは、フラッシュバスコントローラ（ＦＢＣ）を実現する。ブロック１５６ａ～１５６ｎは、フラッシュレーンコントローラ（ＦＬＣ）を実現する。ブロック１５８ａ～１５８ｎは、データ転送パス（ＤＴＰ）を実現する。ブロック１６０は、コンテキストフェッチアービタ（ＣＡ）を実現する。ブロック１６２は、コンテキストフリーポインタリソース（ＣＦＰＭ）を実現

する。ブロック１６４は、消費コンテキストマネージャ（ＣＣＭ）を実現する。ブロック１６６は、コンテキスト訂正ポート（ＣＲＰ）を実現する。ブロック１６８は、コンテキストアップデートポート（ＣＵＰ）を実現する。ブロック１７０は、コンテキストポインタリストポート（ＣＰＬＰ）を実現する。ブロック１７０は、一般に、オプションである。ブロック１７２ａ～１７２ｎは、データＤＭＡ読み取りインタフェースポート（ＤＤＲＩＰ）を実現する。ブロック１７４ａ～１７４ｎは、データＤＭＡ書き込みインタフェースポート（ＤＤＷＩＰ）を実現する。そして、ブロック１７２ａ～１７２ｎ及び１７４ａ～１７４ｎは、一般に、データＤＭＡインタフェースポート（ＤＤＩＰ）を構成する。

【００１９】

一例では、ブロック１５０は、ブロック１０８からブロック１０２の指定可能なリソースにインタフェースを提供する（例えば、ＡＭＢＡ　ＡＨＢライトインタフェースを介して）。ブロック１５０は、構成のために全ての指定可能なリソース、ダイレクトインタフェース、及びブロック１５６ａ～１５６ｎの内部に存在しないブロック１０２のサブモジュールの状態レジスタにインタフェースを提供する。ブロック１５０は、同様にそれぞれのブロック１５６ａ～１５６ｎの内部に存在する指定可能なリソースにインタフェースを提供する。更に、ブロック１５０は、プロセッサファームウェアはブロック１６８を介してシステムバッファの中への記憶のためのブロック１０２に実メディアコンテキストを書き込むコンテキスト構造バッファ（ＣＣＢ）を含む。一例では、ブロック１５０は以下の特徴を含む：ブロック１０８に対する３２ビットＡＭＢＡ　ＡＨＢライトスレーブインタフェース、入力クロック（例えば、ＨＣＬＫ）のいくらか分割値（又は同じ）システムクロック（例えば、ＳＹＳ＿ＣＬＫ）、ブロック１０２の全てのプロセッサ指定可能スペースと同様に全ての構成及び状態レジスタに対するアクセス、システムバッファに格納されるコンテキストを構築するためにプロセッサファームウェアによって用いられるコンテキスト構造バッファ（ＣＣＢ）、指定可能リソースのアクセスがプロセッサアクセスポート（ＰＡＰ）によってハンドリングされ、ブロック１０２で複数のサブモジュールによって用いられるレジスタを含んでいる場合、ブロック１５６ａ～１５６ｎのそれぞれに分配されるプロセッサインタフェース。ブロック１５０は、全てのレジスタの復号化を実行し、ブロック１５６ａ～１５６ｎにおいて論理的に格納されない全ての指定可能なリソースのために読み取りデータを多重化する。

【００２０】

ブロック１５２は、一般に、２つのデータ転送、フラッシュプログラム（例えば、バッファからフラッシュデバイスへのデータトランザクション）のための１つ、及びフラッシュ読み取り（例えば、フラッシュデバイスからバッファへのデータトランザクション）のための別のものを管理する。ＤＭＡデータパスは、一般に、ブロック１５６ａ～１５６ｎからそれぞれのブロック１５８ａ～１５８ｎ、データＤＭＡインタフェースポート（ＤＤＩＰ）ブロック１７２ａ～１７２ｎ及び１７４ａ～１７４ｎを通じて別々の３２ビットの読み書きデータバスを備える。ブロック１５８ａ～１５８ｎは、ＥＣＣ機能を含む。ＤＭＡデータ転送は、一般に、ブロック１０２の他のサブブロック（又はポートブロック）によって対応するコンテキストに対するマルチプルアクセスを含むイベントのシーケンスを備える。一例では、ＤＭＡ転送は、ＦＬＣ要求、検索コンテキスト動作、データ転送、及びＦＬＣ完了フェーズを含む。

【００２１】

ＦＬＣ要求ステップにおいて、データ転送は、それぞれの要求ラインをレイズするブロック１５６ａ～１５６ｎの１つから開始する。検索コンテキスト動作では、対応するコンテキストはコンテキスト訂正ポート（ＣＲＰ）インタフェース１６６を介してバッファコントローラから検索される。データ転送は、コンテキストがＤＤＩＰに送信されると共に返答をし又はしない間に、ＤＤＩＰ、ＤＴＰ、及びＦＬＣブロックの間で発生する。ＦＬＣ完了フェーズでは、選択されたブロック１５６ａ～１５６ｎに対する完了ラインは転送の終わりを示すためにレイズされる。ＤＤＭ１５２は、コンテキストを検索し、データトランザクションを容易にするためにＤＴＰブロックに入力を提供するために作用する。

【００２２】
ブロック１５４ａ～１５４ｎは、一般に、フラッシュレーンのそれぞれの１セットのＮＡＮＤフラッシュデバイスに対するローレベルのインタフェースシグナリングを実行する。一般に、フラッシュレーンコントローラ（ＦＬＣ）１５６ａ～１５６ｎのそれぞれのために１つのフラッシュバスコントローラ（ＦＢＣ）１５４ａ～１５４ｎがある。ブロック１５４ａ～１５４ｎは、一般に、与えられたタイプ（例えば、非同期、ＯＮＦＩ２．０同期、ＯＮＦＩ２．３同期、三星トグル１．０、三星トグル２．０等）のための異なるタイミングモードと同様にいくつかのインタフェースタイプのために、フラッシュインタフェースプロトコルのサイクルのそれぞれのタイミングを管理する。サイクルのタイミングは、一例では、内部タイミングレジスタのグループに格納されたタイミングカウントを介して制御される。ブロック１５４ａ～１５４ｎのコアロジックは、一般に、ブロック１０２の残りとは異なるクロックドメインで動作する。一般に、タイミングレジスタセットのみが、ブロック１５６ａ～１５６ｎの残りと同一のクロックドメインに属する。同期ロジックは、一般に、ＦＢＣが静止しているときに限り（例えば、未解決の動作がない）、レジスタが書き込まれるため、レジスタが静的なものとして扱われるので、これらのレジスタとＦＢＣコアとの間に必要でない。
【００２３】
ブロック１５６ａ～１５６ｎは、一般に、ダイのそれぞれに対するコマンドのスケジューリングを実行する。ブロック１５６ａ～１５６ｎは、フラッシュレーンのそれぞれでのコマンドのシーケンシングを管理する。ブロック１５６ａ～１５６ｎは、ダイをプログラムすると共に状態を監視するファームウェアを通じて制御と状態レジスタを提供する。ブロック１５６ａ～１５６ｎのそれぞれは、コンテキスト管理及びダイ管理を含む。ブロック１５６ａ～１５６ｎは、一般に、コンテキストの処理のための役割を担う。
【００２４】
ブロック１５８ａ～１５８ｎのそれぞれは、データトラフィックを送り、ブロック１５４ａ～１５４ｎ、オプションの内部ＥＣＣエンコーダ／デコーダ、及びデータＤＭＡインタフェースポート（ＤＤＩＰ）のそれぞれの１つの間においてデータフローのためのインタフェースのそれぞれのフロー制御を有効にする。一例では、内部ＥＣＣエンコーダ／デコーダは、ブロック１５８ａ～１５８ｎの内部で実現される。また、ブロック１５８ａ～１５８ｎのそれぞれは、単一のＥＣＣエンコーダ／デコーダモジュールを共有するように構成される。ブロック１５８ａ～１５８ｎは、データＤＭＡマネージャ（ＤＤＭ）モジュール１５２のそれぞれとデータＤＭＡインタフェースポート（ＤＤＩＰ）ブロック１７２ａ～１７２ｎ及び１７４ａ～１７４ｎのそれぞれとの両方によって転送のそれぞれのためにプログラムされる。ブロック１５８ａ～１５８ｎのそれぞれは、全二重動作モードで動作する独立したフラッシュ読み書きパスを含む。ブロック１５８ａ～１５８ｎは、リージョンのそれぞれの内部の現在のｄｗｏｒｄカウントと同様にデータ転送の間に現在のリージョンカウントを保持する。ブロック１５８ａ～１５８ｎは、一般に、ＤＤＩＰ、ＥＣＣエンコーダ＆デコーダ、及びＦＬＣブロックの間のフロー制御変換を実行する。ブロック１５８ａ～１５８ｎは、転送のそれぞれのために訂正可能なＥＣＣエラー和の動作を保持し、転送の終わりでブロック１５２に対する終値を送信する。ブロック１５８ａ～１５８ｎは、ＥＣＣエンコーダ＆デコーダのプログラムのために用いられるＦＭＣレジスタを含む。レジスタは、ブロック１５０からのレジスタインタフェースを介してアクセスされる。ＥＣＣモジュールは、一般に、４Ｋバイト以上のデータの６４ビット訂正ができる。しかしながら、他のレベルの訂正は、特定の実施の形態のその設計基準を満たすために実現される。一例では、デコーダゲートカウントは４１５Ｋゲートであり、エンコーダゲートカウントは７５Ｋゲートである。
【００２５】
ブロック１６０は、一般に、ブロック１５６ａ～１５６ｎからコンテキストのための要求を受け取り、システムバッファ（例えば、バッファコントローラを介してアクセスされたＤＲＡＭ）から要求されたコンテキストを検索し、その後、ブロック１５６ａ～１５６

ｎにコンテキストを伝達するための役割を担う。訂正は、事実上、コンテキスト訂正アクセスポート（ＣＲＰ）１６６に対する要求を介して実行される。コンテキストは、ＦＭＣの制御の基本ユニットである。コンテキストは、一般に、コマンドを実行するためのＦＬＣによって、及びシステムバッファに又はそのシステムバッファから関連データ転送（ＤＭＡ）を実行するためのＦＭＣによって必要とされる全ての情報を含む。ＦＬＣは、完全に自主的に動く。従って、ＦＬＣは、ファームウェアによって構築されたコンテキストのリンクリストを含むシステムバッファへのバッファコントローラを介するアクセスのためのアービトレーションを必要とする。ブロック１６０は、一般に、ブロック１６６に対する要求の開始と同様にアービトレーションを提供する。その後、ブロック１６０は、ＦＬＣデスティネーションのそれぞれに検索されたコンテキストを率直に送る。ブロック１６２は、一般に、フリーポインタがファームウェアに使用可能な単一のポイントを提供するために、ブロック１０２のサブブロックとして実現される。

【００２６】

ブロック１６４は、一般に、完了したコンテキストが完了の後にファームウェアによって検査される単一のポイントを提供するために、ブロック１０２のサブブロックとして実現される。ブロック１６４は、一般に、複数のＦＬＣソースの間でアービトレーションを実行する。ＦＬＣは、コンテキストポインタに関連するパス／フェイルＥＣＣ状態を提供する。ブロック１６４は、コンテキストがフェッチされたならば、コンテキスト状態フィールドをアップデートし、その後、ファームウェアに対するコンテキストを送信する。ファームウェアが完了したコンテキストを読み取るためのより長い時間を要し、ブロック１６４の内部の内部メモリが満たされる場合には、ブロック１６４は、現在の報告されたコンテキストの後に待ち行列に入れられる完了したコンテキストを格納するためにバッファを用いる。

【００２７】

ブロック１６６～１７４ｎは、一般に、ポートインタフェースを実現する。ポートインタフェースは、バッファコントローラによって通信するために用いられる。一例では、ＱＢＦＩＦＯブロックは、ポートインタフェースの内部で実現される。以下のポートインタフェースもポートインタフェースの一部として実現される：コンテキスト訂正ポート（ＣＲＰ）１６６、コンテキストアップデートポート（ＣＵＰ）１６８、コンテキストポインタリストインタフェースポート（ＣＰＬＩＰ）１７０（オプション）、データＤＭＡ読み取りインタフェースポート（ＤＤＲＩＰ）１７２ａ～１７２ｎ、及びデータＤＭＡ書き込みインタフェースポート（ＤＤＷＩＰ）１７４ａ～１７４ｎ。一例では、ブロック１０２のインタフェース信号は、４大インタフェースにグループ化される：ＡＨＢインタフェース、バッファコントローラインタフェース、ＮＡＮＤ及び／又はＮＯＲフラッシュフィジカルレイヤ（ＰＨＹ）インタフェース、及びミソレニアス（ＭＩＳＣ）インタフェース。バッファコントローラインタフェースは、（ｉ）レーン０＆レーン１のためのＤＤＩＰ　ＢＣ書き込みインタフェース、（ｉｉ）レーン０＆レーン１のためのＤＤＩＰ　ＢＣ読み取りインタフェース、（ｉｉｉ）ＣＲＰ　ＢＣ読み取りインタフェース、（ｉｖ）ＣＵＰ　ＢＣ書き込みインタフェース、及び（ｖ）ＣＰＬＩＰ　ＢＣ読み取り／書き込みインタフェース。

【００２８】

一例では、ブロック１０２は、３つのクロックで実現される。ブロック１０２の大多数のロジックは、システムクロック（例えば、ＳＹＳ＿ＣＬＫ）と呼ばれるクロックドメインで動作する。システムクロックは、ＡＨＢクロックである。システムクロックは、一般に、ＦＭＣプロセッサ（ＦＡＲＭ）１１２の動作周波数の２分の１の周波数を有する。第２のクロックは、フラッシュクロック（例えば、ＦＢＣ＿ＣＬＫ）と呼ばれる。フラッシュバスコントローラ（ＦＢＣ）１５４ａ～１５４ｎは、完全にフラッシュクロックドメインで動作する。一例では、先入れ先出しバッファ（ＦＩＦＯ）は、クロックＦＢＣ＿ＣＬＫとＳＹＳ＿ＣＬＫとの間の周波数を管理するために、ブロック１５４ａ～１５４ｎのデータフローマネージャ（ＤＭ）モジュールで実現される。第３のクロックは、バッファコ

ントローラクロック（例えばＢＣ＿ＣＬＫ）である。ＢＣを有する全てのインタフェースポートは、バッファコントローラクロックドメインで動作する。バッファリングエレメント（例えば、ＱＢＦＩＦＯ）は、バッファコントローラクロックＢＣ＿ＣＬＫとシステムクロックＳＹＳ＿ＣＬＫとの間で実現される。

【００２９】

図３を参照すると、本発明の実施の形態によるフラッシュレーンコントローラアーキテクチャの一例を図示するブロック２００の図が示される。ブロック２００は、一例では、図２のブロック１５４ａ～１５４ｎ及び１５６ａ～１５６ｎを実現するために用いられる。一例では、ブロック（又は回路）２００は、ブロック（又は回路）２０２、ブロック（又は回路）２０４、ブロック（又は回路）２０６、ブロック（又は回路）２０８、ブロック（又は回路）２１０、ブロック（又は回路）２１２、及びブロック（又は回路）２１４を備える。回路２０２～２１０は、ハードウェア、ファームウェア、ソフトウェア、ハードウェアの組み合わせ、ファームウェア、及び／又はソフトウェア、又は他の実装として実現されるモジュール及び／又はブロックを表わす。ブロック２０２は、一例では、コンテキスト処理コーディネータ（ＣＰＣ）を実現する。ブロック２０４は、一例では、コンテキストマネージャ（ＣＭ）を実現する。ブロック２０６は、一例では、ダイ管理モジュール（ＤＭＭ）を実現する。ブロック２０８は、一例では、フラッシュ動作マネージャ（ＦＯＭ）を実現する。ブロック２１０は、一例では、プロセッサアクセスポート（ＰＡＰ）を実現する。ブロック２１２は、一例では、フラッシュバスコントローラ（ＦＢＣ）を実現する。ブロック２１４は、一例では、データフローマネージャ（ＤＦＭ）を実現する。

【００３０】

ブロック２０２は、ブロック２００に及びそのブロックからのコンテキスト情報のフローをアシストする。コンテキストフローは、ブロック２０４によって開始される。ブロック２０２は、コンテキストを取得するか解放するための要求に応えることに主として関する。コンテキストを取得するために、ブロック２０２は、ブロック２０４による新規のコンテキストのための要求に応える。最初に、ブロック２０２は、ブロック２０６に対する要求を開始する。ブロック２００によって管理されたダイの間にアービトレートし、ブロック２０２に選択されたダイ又は論理装置番号（ＬＵＮ）のためのコンテキストを転送する。その後、ブロック２０２は、システムバッファからコンテキストを検索することを試みるコンテキストフェッチアービタ（ＣＦＡ）（例えば、図２のブロック１６０）に対するフェッチをもたらす。

【００３１】

フェッチされたならば、コンテキストは、ブロック２０２に伝達される。ブロック２０２は、コンテキストでいくらかの解釈を実行し、ブロック２０４にコンテキストを転送する。ブロック２０６がコンテキストの実行を開始するために使用可能なダイ（ＬＵＮ）を有さないならば、ブロック２０６は、使用可能なダイの不足をブロック２０２に通知し、ブロック２０２は、ブロック２０４に使用可能なダイの不足への返答を通信する。ブロック２０２は、完了したコンテキストの解放においてもブロック２００をアシストする。再び、このフローを開始するのはブロック２０４であり、消費コンテキストマネージャ（ＣＣＭ）（例えば、図２のブロック１６４）を実現するブロックに対する解放メッセージをもたらすのはブロック２０２である。解放メッセージがＣＣＭによって受信され動作が開始されたとき、ブロック２０２は、ブロック２０４に通知し、その後、ブロック２０４は、コンテキスト処理の実行を保持する。

【００３２】

ブロック２０２は、一般に、コンテキストのいくらかの解釈を実行する。具体的には、ブロック２０２は、コンテキストがプロセッサ制御モード（ＰＣＭ）コンテキストか否かを判断する目的でコンテキストを解釈する。ＰＣＭコンテキストが受信されるとき、コンテキストフェッチ（追加）は停止するべきである。その後、ブロック２０２は、ブロック２０４がＰＣＭコンテキストを実行し始めることを待ち、プロセッサ制御モードが完了す

るとき、「スタンダード」動作を再開する。プロセッサ制御モードインターバルの間に、ブロック２０２は、フェッチされたコンテキストが４つのｄｗｏｒｄフラッシュコンテキストの代わりに完全な１５のｄｗｏｒｄコンテキストか否かを判断し、ブロック２０２は「スタンダード」動作のブロック２０４にそれらを送信する。

【００３３】

ブロック２０４は、一例では、コンテキスト状態マシン（ＣＳＭ）、コンテキストフェッチマネージャ（ＣＦＭ）、コンテキスト解放エンジン（ＣＤＥ）、及びコンテキストインタープリタ（ＣＩ）を備える。ブロック２０４は、一般に、ブロック２００によって活動的に処理されるコンテキストを管理する役割を担う。ブロック２０４は、一般に、アクティブなコンテキストの「ブックキーピング」を実行する。コンテキストは、システムバッファに対するフラッシュトランザクション及びＤＭＡを実行するためにフラッシュメディアコントローラ（ＦＭＣ）によって必要とされる全ての情報を提供するデータ構造である。ブロック２０４は、フラッシュレーンコントローラのレベルでコンテキストを管理し、従って、フラッシュトランザクションに関連するように、コンテキスト管理に主として関する。ブロック２０４は、フラッシュレーンのフラッシュダイに対するコマンド及びデータ転送を実行するために、ブロック２０８によって用いられる情報を保持する。

【００３４】

ブロック２０６は、一般に、ブロック２００の動作のために必要とされるダイに基づいた情報を保持するための役割を担う。ブロック２０６は、ダイ管理テーブルのダイごとに情報を管理し、コンテキストテーブルへの待ち行列となるためのアクセスのためにダイの間にアービトレートする。ブロック２０６は、一例では、ダイ状態をアップデートするためにダイ状態マシンを含む。ブロック２０６は、マルチダイ動作を実行／モニタリングする。ブロック２０６は、一般に、ＲＥＡＤ、ＣＯＰＹＢＡＣＫ　ＲＥＡＤ／ＣＯＰＹＢＡＣＫ　ＷＲＩＴＥ、ＢＬＯＣＫ　ＥＲＡＳＥ、ＰＡＧＥ　ＰＲＯＧＲＡＭを含み、これらに限定されないフラッシュコマンド、及びＲＥＡＤ　ＩＤ、ＲＥＡＤ　ＰＡＲＡＭＥＴＥＲ　ＰＡＧＥ、ＧＥＴ　ＦＥＡＴＵＲＥＳ、ＳＥＴ　ＦＥＡＴＵＲＥＳ、ＳＹＮＣＨＲＯＮＯＵＳ　ＲＥＳＥＴ、及びＲＥＳＥＴを含み、これらに限定されないターゲットレベルコマンドのための役割を担う。

【００３５】

ブロック２０８は、一般に、フラッシュレーンに適用されたフラッシュ動作のそれぞれのシーケンシングをハンドリングする。１つのブロック２０８は、一般に、フラッシュメディアコントローラのフラッシュレーンコントローラ（ＦＬＣ）のそれぞれのために実現される。ブロック２０８は、ブロック２０４のコンテキストテーブルのコマンドの間にアービトレートし、ブロック２１２にコマンドを適用する。一例では、ブロック２０８は、本来、三星ＮＡＮＤフラッシュデバイスにおいて知られたいくつかの特定の（かつ類似した）コマンドと同様にＯＮＦＩ２．０コマンドリストからの大部分の共通コマンドをサポートする。更に、他の既存の及び将来のコマンドは、ナノシーケンサ（図９～１１に関してより詳細に説明される）を介してサポートされる。本来、サポートされたコマンドは、プロセッサインターベンションなしに実行されるが、一般に他のコマンドはいくらかのレベルのプロセッササポートを用いる。

【００３６】

フラッシュコマンドは、ブロック２０８によって制御された実際のフラッシュダイに連続的に適用される原子の「サイクル」に分割される。典型的には、フラッシュコマンドが長いウェイト時間（例えば、データがチップから読み取られるために使用可能な前に、ページ読み取りは２５μｓを要する）を含んでいるので、「コマンドサイクル」は、しばしばフラッシュレーン上の異なるダイに「バックツーバック」実行され、従って、累積的なウェイト時間を効果的に削減できる。ブロック２０８は、一般に、フラッシュ「サイクル」のそれぞれが適用されるとき、ダイの状態をアップデートすることによってフラッシュダイを管理する。その後、ブロック２０８は、「サイクル」が次に実行されるべき（又はありうる）であるかを判断するべきための最新のコンテキストテーブルを読み取る。ＮＡ

ＮＤフラッシュ動作は、一般に、１つ以上のフラッシュサイクルで構成される。４つのタイプのフラッシュサイクルが一般にある：コマンド、アドレス、データ出力（フラッシュデバイスについては、例えば、読み取り）、及びデータ入力（フラッシュデバイスについては、例えば、書き込み）。サイクルタイプは、大雑把に言ってブロック２０８とブロック２１２との間に定義された動作タイプに解釈する。

【００３７】

ブロック２１０は、一般に、ブロック２００の内部のアドレス可能なリソースにＦＭＣ１００のＡＨＢライトスレーブインタフェースからのプロセッサアクセスを提供するインタフェースブロックを実現する。ここでアドレスされた大部分のリソースは、全てのコンフィギュレーション信号がグローバルレベル（共有コンフィギュレーションレジスタブロックの一部としての）で送信されるとき、主として診断目的のためのアクセス可能である。例えば、フラッシュレーンデータバッファに対する完全なアクセスは、ブロック２１０によって使用可能である。アクセスは単に迅速な検証の足がかりとして提供される。しかしながら、フラッシュレーンデータバッファに対するアクセスは、内部テーブルに対する直接アクセスを必要とするファームウェアパッチもサポートする。そのようなアクセスは、ブロック２１０によって提供される。

【００３８】

ブロック２１０の特徴は次のものを含む：ＡＨＢライトスレーブプロトコルに従い、ＦＭＣにおいてプロセッサインタフェースロジック（ＰＩＬ）によってバッファされるシンプルアクセスインタフェース；リソース、コンテキストテーブル、コンテキストキャッシュ、及びダイ管理テーブルをレジストするために提供される読み取り書き込みアクセス；ブロック２１４に位置されるフラッシュレーンデータバッファメモリリソースに提供される読み取り書き込みアクセス。大部分のコンフィギュレーションレジスタは、一般に、ブロック２００に入力として提供されるが、ブロック２１０は、一般に、レーンごとにコンフィギュレーションレジスタを追加するための能力をサポートする。同様に、大部分の状態及び割り込みレジスタは、一般に、ブロック２００の外部に生成されるが、状態及び割り込みレジスタアクセスはサポートされる。ブロック２１０の主要なロジックグループは次のものを含む：インタフェースマネージャ（ＩＦ＿ＭＧＲ）、データフローマネージャインタフェース（ＤＭ＿ＩＦ）、レジスタブロックデコーダ（ＲＥＧ＿ＤＥＣ）、レジスタブロックマルチプレクサ（ＲＥＧ＿ＭＵＸ）、割り込みハンドラ（ＩＮＴ＿ＨＮＤ）、及びＦＬＣグローバルレジスタ（ＧＬＯＢ＿ＲＥＧＳ）。

【００３９】

図４を参照すると、図３のコンテキストマネージャモジュール２０４のサブモジュールを図示する図が示される。一例では、ブロック２０４は、コンテキストテーブル（ＣＴ）２２０、コンテキスト状態マシン（ＣＳＭ）２２２、コンテキストキャッシュ（ＣＣ）２２４、及びコンテキスト待ち行列コントローラ（ＣＱＣ）２２６を含む。ブロック２０４は、一般に、フラッシュレーンコントローラに対する動作のフェーズをステージし実行し、フラッシュレーンで全てのアクティブなコンテキストのプライオリティオーダリングを保持し、フラッシュレーンでコンテキストのそれぞれの状態を保持し、完全なトランザクションを実行するために必要とされるコンテキストのテンポラリオンチップストレージの最低量を提供し（例えば、コンテキストキャッシュを介して）、実行が進行中のコンテキストのそれぞれのバッファポインタを保持し、及びコンテキスト状態マシン（ＣＳＭ）２２２を用いてコンテキストの次の状態を決定することによって、コンテキストのそれぞれのためにエージェンシーを提供する。最小のコンテキスト情報は、コンテキストテーブル（ＣＴ）２２０に保持される。コンテキストテーブル２２０は、一般に、現在実行されているコンテキストの優先待ち行列を提供する。コンテキスト待ち行列コントローラ（ＣＱＣ）２２６は、コンテキストテーブル２２０から完了したコンテキストを除去し、かつギャップを削減するためにコンテキストテーブル２２０を圧縮するように構成される。

【００４０】

図５を参照すると、図３のダイ管理モジュール２０６のサブモジュールを図示する図が

示される。一例では、ブロック２０６は、ダイ状態マシン２３０、ダイサービスアービタ２３２、及びダイ管理テーブル２３４を備える。

【００４１】

図６を参照すると、図３のフラッシュ動作マネージャ（ＦＯＭ）２０８のサブモジュールを図示する図が示される。一例では、ブロック２０８は、４つのサブモジュール、コマンドアービタ（ＣＡ）２４０、データ転送アービタ（ＤＴＡ）２４２、フラッシュ動作フォーマッタ（ＦＯＦ）２４４、及びナノシーケンサ２４６に分割される。コマンドアービタ２４０は、一般に、適用するためのコマンドのためにコンテキストテーブルをスキャンし、その後、フラッシュバッファコントローラ（ＦＢＣ）に信号を送信するためにフラッシュ動作フォーマッタ（ＦＯＦ）２４４によって通信する。全ての「コマンド」の部分が実行され、フラッシュが「データフェーズ」のための準備ができたならば、データ転送アービタ２４２は、ＦＢＣとデータフローマネージャ（ＤＭ）２１４との間の転送を開始する。最後に、ナノシーケンサ２４６は、本来、コマンドシーケンスがサポートされてなくても、フラッシュが必要とするあらゆるコマンドシーケンスを適用するために特別の「ソフトコンテキスト」を解釈する。

【００４２】

図７を参照すると、図３のデータフローマネージャ２１４のサブモジュールを図示する図が示される。データフローマネージャ２１４は、一般に、フラッシュレーンデータバッファメモリリソースを提供する。一例では、フラッシュレーンデータバッファメモリリソースは、カットスルーバッファ２５０及び２５２を備える。一例では、カットスルーバッファ２５０及び２５２は、プログラマブルなサイズで実現される。例えば、バッファ２５０及び２５２のサイズは、帯域幅仕様とマッチするように調節される。一例では、バッファ２５０及び２５２は、静的ランダムアクセスメモリ（ＳＲＡＭ）を備える。しかしながら、他のタイプのメモリは、特定の実施の形態のその設計基準を満たすために実現される。一般に、２つのカットスルーバッファは、フラッシュレーンごとに実現される。

【００４３】

図８を参照すると、図３のコンテキストマネージャ（ＣＭ）２０４の実施の形態の一例を図示する図が示される。コンテキストマネージャ（ＣＭ）２０４は、一般に、フラッシュレーンコントローラ（ＦＬＣ）のそれぞれによってアクティブに処理されているコンテキストを管理する役割を担う。ＣＭ２０４は、一般に、アクティブなコンテキストの「ブックキーピング」を実行する。先に述べたように、コンテキストは、システムバッファに対するフラッシュトランザクション及びＤＭＡを実行するためにフラッシュメディアコントローラ（ＦＭＣ）１０２によって用いられる全ての情報を提供するデータ構造である。ＣＭ２０４は、ＦＬＣのレベルでコンテキストを管理し、従って、フラッシュトランザクションに関連するコンテキスト管理に主として関する。ＣＭ２０４は、フラッシュレーンのフラッシュダイに対するコマンド及びデータ転送を実行するためにフラッシュ動作マネージャ（ＦＯＭ）によって用いられる情報を保持する。

【００４４】

ＣＭ２０４は、一般に、（ｉ）フラッシュレーンコントローラのそれぞれに対する動作のフェーズをステージし実行し、するように構成される。（ｉｉ）フラッシュレーンのそれぞれで全てのアクティブなコンテキストのプライオリティオーダリングを保持し、（ｉｉｉ）フラッシュレーンのそれぞれでコンテキストのそれぞれの状態を保持し、（ｉｖ）完全なトランザクションを実行するために用いられるコンテキストのテンポラリオンチップストレージ（例えば、コンテキストキャッシュ２２４を介して）の最低量（又は最小限に抑えた量）を提供し、（ｖ）実行が進行中のコンテキストのそれぞれのバッファポインタを保持し、（ｖｉ）コンテキスト状態マシン（ＣＳＭ）２２２を用いてコンテキストの次の状態を決定することによって、コンテキストのそれぞれのためにエージェンシーを提供し、及び（ｖｉｉ）現在実行されているコンテキスト（例えば、コンテキストテーブル２２０）の優先待ち行列に最小のコンテキスト情報を保持する。コンテキスト待ち行列コントローラ２２６は、一般に、コンテキストテーブル２２０から完了したコンテキストを

除去し、かつギャップを削減するためにコンテキストテーブル２２４を圧縮するように構成される。

【００４５】

コンテキスト待ち行列コントローラ（ＣＱＣ）２２６は、コンテキストテーブル（ＣＴ）２２０で修正を実行するロジックブロックである。ＣＴ２２０は、一例では、待ち行列に入れられたコンテキストごとに１つのエントリに編成されるレジスタのブロックとして実現される。ＣＱＣ２２６は、優先待ち行列として編成されるテーブルで動作を実行するブロックである。ＣＱＣ２２６は、一般に、コンテキスト処理を開始し実行し、コンテキストテーブルの処理を実行する役割を担う。メイン処理は、一般に、追加、待機、修正、解放、及び圧縮の処理を含む。処理は、ＣＱＣ２２６によってステージされ実行される。

【００４６】

追加フェーズは、新規のコンテキストがＦＭＣによってフェッチされ、それらのコンテキストのためのエントリがコンテキストテーブル２２０に追加されるフェーズである。ＣＱＣ２２６は、ＣＰＣ２０２によって送信されたフラッシュコンテキスト及びコンテキスト情報の内容を検査し、内容とコンテキスト情報に基づいたエントリを追加し、作成する。一例では、コンテキストテーブルエントリは、コンテキストテーブルエントリがアクティブか否かを示すビット（又はフラグ）、コンテキスト状態を表わす値、コンテキストキャッシュインデックスを表わす値、フラッシュ動作を送信する値、フラッシュダイを表わす値、コンテキストポインタ、データ転送を無効にするべきか否かを示すビット（又はフラグ）、及びプレーンアドレスを表わす値を備える。新規なエントリは、一般に、「アクティブな」ビットセット（例えば、ロジック「１」）によって開始し、「コンテキスト状態」は値「ＱＵＥＵＥＤ」に設定する。フラッシュ動作がイリーガルの場合、初期状態は値「ＩＬＬＥＧＡＬ」に設定され、コンテキストテーブルエントリは解放フェーズの間に除去される。他のフィールドは、一般に、ＣＱＣ２２６によって提供されるコンテキストと情報によって決定される。新規なエントリは、一般に、圧縮されたコンテキストテーブル２２０の末尾に追加される。従って、ＣＱＣ２２６は、一般に、コンテキストテーブル２２０の深度を知っている。

【００４７】

ＣＱＣ２２６は、一般に、ＣＱＣ２２６がもはや未解決のデータ転送が完了するのを待っておらず、ＣＱＣ２２６が与えられたフラッシュ動作サイクルの間に少なくとも１つ追加の動作を試みたとき、「追加」フェーズを完了する。ＣＱＣ２２６は、コンテキストテーブル２２０又はコンテキストキャッシュ２２４に使用可能なスペースがもはやないときも、「追加」フェーズを完了する。

【００４８】

コンテキストマネージャ２０４は、完全なフラッシュ動作サイクルの間に待つことを強制されるか、又は強制されない。コンテキストマネージャ２０４は、一般に、最小のフラッシュ動作期間（例えば、フラッシュ動作期間レジスタを介して）を強制するための能力を有する。そのような最小の期間は、例えば、フラッシュレーンがＰＲＯＧＲＡＭ又はＥＲＡＳＥのコマンドの後にポーリングすることを除いて主として使用されていないケースのために望ましい。そのようなインスタンスでは、コンテキストフェーズは、追加又は解放がないとき、実行するべきための非常に短時間を要する。従って、レーンが連続的に使用中であるフラッシュダイをポーリングする状態においてレーンが存在するための傾向があり、その結果としてその消費電力が保証されないときフラッシュインタフェースで電力を消費する。ＣＱＣ２２６は、一般に、所定時間が完了する（例えば、時間は「フラッシュ動作タイマー」レジスタにおいて指定される）まで存続する。所定時間が完了したとき、ＣＱＣ２２６は「修正」フェーズに入る。

【００４９】

ＣＱＣによって開始される次のフェーズは、一般に、「修正」フェーズである。修正フェーズでは、コンテキストテーブル２２０は、フラッシュ動作マネージャ（ＦＯＭ）によって及びデータパス転送からの結果によって実行されたフラッシュ動作に基づいて修正さ

れる。アップデートは、一般に、コンテキストの状態に関連し、従って、一般に、コンテキスト状態マシン（ＣＳＭ）２２２によって開始される。状態アップデートが発生するとき、ＣＳＭ２２２はＣＱＣ２２６に最新の状態及びコンテキストテーブルインデックスを送信する。その後、ＣＱＣ２２６は、コンテキストテーブル２２０のエントリをアップデートする。ＦＯＭがフラッシュインタフェース処理のサイクルを終えるとき修正フェーズは終わる。ＦＯＭは、信号（例えば、ＦＯＭ＿ＣＭ＿ＦＬＡＳＨ＿ＰＲＯＣ＿ＣＭＰＬＴ）のアサートによって、フラッシュインタフェース処理が終わったことをコンテキストマネージャ２０４に通知する。修正フェーズが完了したならば、ＣＱＣ２２６は、解放、圧縮、及びコンテキストテーブル２２０のコンテキストの追加を実行する。この時間の間に、コンテキストテーブル２２０はＦＯＭに対してアクセス不能である。ＣＱＣ２２６は、コンテキストテーブルがエントリを読み取ったこと、及びコンテキストキャッシュに読み取られたデータが特定のクロックサイクルの間に有効であることをＦＯＭに示す信号（例えば、ＣＭ＿ＦＯＭ＿ＣＴ＿ＶＡＬＩＤ）をデアサートすることによってＦＯＭに対してアクセス不能なコンテキストテーブル２２０を強制する。

【００５０】

修正フェーズが完了したとき、ＣＰＣ２０２は「解放」アクションを開始する。解放アクションは、ＣＱＣ２２６が実行を終えたエントリ有産者を捜すコンテキストテーブル２２０を探索するモードにＣＱＣ２２６を至らせる。ＣＱＣ２２６は、エントリがコンテキストの状態の実行を終えたか否かの判断に基づく。コンテキストが「完了」状態であるとき、コンテキストはＣＱＣ２２６によって解放される。一例では、コンテキストは、ＣＱＣ２２６がコンテキストの完了状態に関してデータパスからの通知を待つ状態である。例えば、読み取り動作の場合には、コンテキストはＤＡＴＡ＿ＴＲＡＮＳＦＥＲ＿ＤＯＮＥ状態にあり、ＥＣＣチェックの結果を待つ。この場合には、ＣＱＣ２２６は一時的に解放処理を停止し、状態がデータパスから復帰されるのを待つ。この時間の間に、ＣＱＣ２２６は「追加」を許可する。しかしながら、待機状態が復帰されれば、コンテキストはＣＱＣ２２６によって解放され、消費されたコンテキストレコードはＣＰＣ２０２に（結局、消費コンテキストマネージャ（ＣＣＭ）１６４に）転送される。

【００５１】

ＣＱＣ２２６がコンテキストを解放したとき、ＣＱＣ２２６はコンテキストテーブル２２０の通信エントリのための「アクティブ」ビットを除去する。ＣＱＣ２２６がコンテキストテーブル２２０ごとにコンテキストを調査するまで、その処理は継続する。ＣＱＣ２２６がコンテキストテーブル２２０のアクティブなコンテキストの終わりに到達するとき、解放フェーズは完了する。

【００５２】

ＣＱＣ２２６によって解放されたコンテキストは除去されたテーブルエントリのそれぞれにおいて「アクティブ」ビットを有する。ホールを満たすまでテーブルをシフトするための機構なしに、アクティブなエントリはコンテキストテーブル２２０に支出される（又は細分化される）。支出されたコンテキストは、コンテキストテーブルをスキャンすることを困難にし、「追加」フェーズをより複雑にする。コンテキストテーブル２２０が優先待ち行列としてそのキャラクタを保持することを保証するために、コンテキストテーブル２２０は圧縮される。圧縮処理において、ＣＱＣ２２６がコンテキストを解放するとき、ＣＱＣ２２６は１つの位置によって解放されたエントリの後に直ちに全てのエントリをシフトする。処理が完了するとき、全てのアクティブなエントリはプライオリティオーダーのリストのフロントにあり、全ての「ホール」は除去される。他のアクションの場合であるとき、圧縮処理が完了されるとき、ＣＱＣ２２６は「完了」セマフォ（又はビット）をアサートする。最終の圧縮フェーズの終わりで、ＣＱＣ２２６は追加フェーズを開始する。

【００５３】

ＣＱＣ２２６は、一般に、プロセッサ制御モードを知っている。プロセッサ制御モードでは、全てのＣＭ２０４はスタンダード動作を停止し、フラッシュ動作マネージャ２０８

の内部のナノシーケンサ２４６によって実行される「ソフトコンテキスト」によってＦＬＣの動作が本質的に駆動されるモードで進行する。ソフトコンテキストは、スタンダードフラッシュコンテキストとは異なるサイズである。一例では、ソフトコンテキストは完全な１５の３２ビットダブルワードを備えるが、「フラッシュコンテキスト」、ＦＬＣによって実行された完全なメディアコンテキストの部分は、一般に、ちょうど４つの３２ビットダブルワードを備える。

【００５４】

プロセッサ制御モード（ＰＣＭ）は、一般に、「フラッシュ動作」フィールドがＰＲＯＣＥＳＳＯＲ＿ＣＯＮＴＲＯＬ＿ＭＯＤＥに設定されるコンテキストがコンテキスト待ち行列のトップに現われるとき開始する。一般に、ＣＱＣ２２６がＰＣＭコンテキストを待ち行列に入れれば、ＣＱＣ２２６がスタンダードコンテキストの訂正を停止するとき、コンテキストテーブル２２０にＰＣＭコンテキストの後ろにアクティブなエントリはない。ＰＣＭが開始するとき、ＣＱＣ２２６は信号（例えば、ＣＭ＿ＣＰＣ＿ＰＲＯＣ＿ＣＮＴＬ＿ＭＯＤＥ）を介してＣＰＣ２０２に通知する。通知に応じて、ＣＰＣ２０２は、ＰＣＭコンテキストに与えられた位置で見つけられた「ソフトコンテキスト」をフェッチする。ＦＯＭに送信されるものの観点から、ＰＣＭコンテキストがコンテキストテーブル２２０の他のアクティブなエントリの後ろにある間に、ＦＯＭは、一般に、コンテキストテーブル２２０のＰＣＭコンテキストの存在についての情報を有さない。ＣＭ２０４がソフトコンテキストを実行し始めるＦＯＭのための準備ができるまで、コンテキストテーブル２２０のＰＣＭコンテキストエントリは、ＦＯＭに「アクティブ」ビットを０として送信する。

【００５５】

ＦＯＭがソフトコンテキストを読み取り始めるとき、ソフトコンテキストが格納されるコンテキストキャッシュ２２４によって動作がＦＯＭ２０８に送信されるとき、ＣＱＣ２２６は動作をスヌープする。動作がＤＭＡコンテキスト（例えば、プリフェッチデータ、セット読み取りデータバッファ、又はコンテキストポインタの解放）を含んでいるとき、ＣＱＣ２２６は、コンテキストテーブル２２０の現在未使用のストレージを選出し、トラッキングのためのコンテキストテーブルにポインタを配置する。それらのＤＭＡコンテキストが完了するとき、ＦＯＭ２０８はコンテキストマネージャ２０４に通知し、その後、コンテキストマネージャ２０４は通常の方法でコンテキストを解放する。

【００５６】

スヌープしている間に、ＣＱＣ２２６は、「ネクストソフトコンテキストのフェッチ」動作を捜す。ＣＱＣ２２６がそれを見つけるとき、ＣＱＣ２２６はＣＰＣ２０２に信号（例えば、ＣＭ＿ＣＱＣ＿ＰＣＭ＿ＮＥＸＴ＿ＣＯＮＴＥＸＴ）をアサートし、ＣＰＣ２０２はネクストソフトコンテキストをフェッチする。ＦＯＭ２０８がソフトコンテキスト実行が完了したことをＣＭ２０４に通知するとき、ＦＯＭ２０８はＦＯＭ／ＣＭコマンドインタフェースでＣＭ２０４に通知する。その後、ＣＱＣ２２６は、ＣＰＣに信号（例えば、ＣＭ＿ＣＰＣ＿ＰＲＯＣ＿ＣＮＴＬ＿ＭＯＤＥ）をデアサートし、スタンダード動作は継続する。一例では、ＣＭ２０４がプロセッサ制御モードに入っており、現在ソフトコンテキストを受信するための準備ができていること示すためのレベルとして信号ＣＭ＿ＣＰＣ＿ＰＲＯＣ＿ＣＮＴＬ＿ＭＯＤＥがアサートされる。

【００５７】

ＣＱＣ２２６の別の重要な機能は、タイムアウト状況をモニタすることにある。一例では、ＣＱＣ２２６は、同一のコンテキストテーブルエントリがコンテキストテーブル２２０のトップ（例えばエントリ０で）に属するシステムクロック（ＳＹＳ＿ＣＬＫ）サイクルの数を数えるように構成されたカウンタを含む。カウント値がプログラマブルな「タイムアウト」カウンタの値に到達する場合、コンテキストテーブル２２０のトップのエントリはタイムアウトを有すると見なされる。エントリがタイムアウトを有すると見なされるとき、エントリは、コンテキストテーブル２２０から除去され、コンテキストポインタは、消費コンテキストインタフェースのコンテキスト処理コーディネータ（ＣＰＣ）２０２

にリターンされる。

【００５８】

コンテキストのためのリターン状態は２つの起こり得る「タイムアウト」状態の１つである。第１のケースでは、タイムアウトは、可能性としてフラッシュレーンの別のダイが使用中で、Ｒ／Ｂラインを押し下げている状況である。この場合には、状態は、タイムアウトが別のダイのタイムアウトであることを示す。第２のケースでは、コンテキストのためのダイはカルプリットであると認識される。ここで、異なる状態はダイがカルプリットであることを示してリターンされる。

【００５９】

コンテキストテーブル２２０は、本質的にエントリの記憶媒体である。コンテキストテーブルの深度は、パラメータ化することができる。例えば、レーンごとに１６ダイをサポートすることができるチップの場合には、１６のエントリが実現される。１つを超える動作が１つのダイごとに管理されるならば、深度を増加させることは有利である。コンテキストテーブル２２０は、最小の機能を有する。大部分のより複雑な処理は、コンテキストテーブル２２０でＣＱＣ２２６によって実行される。しかしながら、コンテキストテーブル２２０は、複数の読み取りインタフェース、及び読み取りインタフェースのそれぞれのための多重化ロジックで実現される。一例では、コンテキストテーブル２２０は、ＦＯＭ２０８に対するインタフェース、及び読み取りアクセシビリティのためのコンテキスト状態マシン（ＣＳＭ）２２２に対するインタフェースによって実現される。コンテキストテーブル２２０は、ＣＱＣ２２６に対する読み取りインタフェースも有する。コンテキストテーブル２２０は、プロセッサによってもアクセスされる。

【００６０】

コンテキストテーブル２２０は、同様にテーブルの圧縮フェーズのために用いられる「シフト」性能を有する。それに加えて、ＣＱＣ２２６はシンプルな書き込みインタフェースを用いてコンテキストテーブル２２０をアップデートする。一例では、コンテキストテーブル２２０は、フリップフロップで実現される。コンテキストテーブル２２０がフリップフロップで実現されるとき、読み取りアクセスのために必要なアービトレーションはない。コンテキストテーブル２２０が約１０００を超えるフリップフロップのサイズに増加する場合、コンテキストテーブル２２０はレジスタファイル又はＳＲＡＭで実現されるが、追加の管理及びアクセスのアービトレーションも実現される。

【００６１】

コンテキストキャッシュ２２４は、コンテキストテーブル２２０に類似する別のコンテキストデータストレージエレメントである。コンテキストキャッシュ２２４は、一般に、エントリのパラメータ化することができる数を含む。一例では、エントリの数は８である。しかしながら、エントリの他の数は、特定の実施の形態のその設計基準を満たすために実現される。例えば、完全なパイプライン動作のために事実上必要とされるよりエントリの数は１つ又は２つ以上に設定される。数は、一般に、プロセッサ制御モードの完全な「ソフトコンテキスト」のための十分なスペースを許可するために十分に大きくされる。先に述べたように、完全なコンテキストは１５の３２ビットダブルワードを備える。完全なメディアコンテキストのサブセットは「フラッシュコンテキスト」と指称される。フラッシュコンテキストは、一般に、完全なメディアコンテキストの第１の４つのダブルワード（又はｄｗｏｒｄｓ）である。フラッシュコンテキストの４つのｄｗｏｒｄｓは、一般に、ファームウェアによって指定された完全な動作を実行するためにＦＬＣによって用いられる全ての情報を含む。スタンダード動作（例えば、ＦＬＣがプロセッサ制御モードにないとき）の間に、フラッシュコンテキストの第１の２つのｄｗｏｒｄｓのみがコンテキストキャッシュ２２４に格納される。フラッシュコンテキストの残りは、一般に、コンテキストテーブル２２０に格納される。

【００６２】

コンテキストキャッシュ２２４は、一般に、エントリのそれぞれの状態を保持する。一例では、状態は、エントリがＦＲＥＥ又はＵＳＥＤか否かを示すビットを備える。一例で

は、そのような８ビットは、コンテキストキャッシュ２２４で実現される。フラッシュコンテキストがコンテキストキャッシュ２２４で位置に書き込まれるとき、位置の状態はＵＳＥＤになる。ＣＱＣ２２６がその位置をクリアすることを許可する状態変更についての情報を取得するとき、位置の状態はＦＲＥＥにリターンする。スタンダード動作の間に、コンテキストキャッシュ２２４は、コンテキストキャッシュ２２４が状態ビットに基づいてフリーエントリのためのスペースを有することをＣＱＣ２２６に通知する。フリーロケーションがあるならば、ＣＱＣ２２６は自由にＣＰＣ２０２からコンテキストを要求することができる。ＣＰＣ２０２が新規のフラッシュコンテキストをフェッチしたとき、ＣＰＣ２０２はデータの３２ビットダブルワードのバーストとしてコンテキストキャッシュ２２４にフラッシュコンテキストを送信する。データが有効なとき、信号（例えば、ＣＰＣ＿ＣＭ＿ＥＮＱ＿ＣＴＸ＿ＶＡＬＩＤ）がアサートされる。コンテキストキャッシュ２２４は、フリーロケーションにデータを書き込む。コンテキストキャッシュ２２４は、ＣＰＣ２０２が単に１つのフラッシュコンテキストを書き込むだろうと予期する。

【００６３】

コンテキストテーブル２２０のトップのエントリがＰＲＯＣＥＳＳＯＲ＿ＣＯＮＴＲＯＬ＿ＭＯＤＥ動作として示されるときに入るプロセッサ制御モードでは、コンテキストキャッシュ２２４は完全に自由である。プロセッサ制御モードでは、コンテキストキャッシュ２２４は、ＣＰＣ２０２からソフトコンテキストを受信することを予期する。コンテキストキャッシュ２２４は、ソフトコンテキストが１５のｄｗｏｒｄｓを含むことも予期する。本質的に、コンテキストキャッシュ２２４は、ＣＰＣ２０２によって送信された全てのデータを受け取って、スレーブとして行動する。コンテキストキャッシュ２２４にデータの適正量を書き込むことはＣＰＣ２０２の責務である。コンテキストキャッシュ２２４は、ＦＯＭ２０８によってアクセス可能であり、フラッシュユニットで実コマンドを実行するとき、完全なフラッシュコンテキスト情報を用いる。ＦＯＭ２０８は、３２ビットダブルワードにアドレスを提供し、コンテキストキャッシュ２２４は、以下のクロックサイクルで要求されたダブルワードによって応答する。プロセッサ制御モードの間に、コンテキストキャッシュ２２４からの読み取られた応答は、動作の内容に基づいたアクションを実行するコンテキスト待ち行列コントローラ（ＣＱＣ）２２６によってスヌープされる。コンテキストキャッシュ２２４は、コンテキストテーブル２２０であるときも、プロセッサインタフェースによってアクセス可能である。

【００６４】

コンテキスト状態マシン（ＣＳＭ）２２２は、一般に、エントリの現在状態に基づいてコンテキストテーブル２２０のコンテキストのそれぞれの実行状態、及びＦＯＭ２０８によって実行されている動作又はデータパス動作の状態のいずれかを決定するように構成される。修正フェーズでは、ＣＳＭ２２２は、ＦＯＭ２０８がコマンドを適用するか、結果をリターンするごとにＣＱＣ２２６によって呼び出される。ＦＯＭコマンド通知インタフェース及びＦＯＭコンテキストテーブル読み取りインタフェースの内容は、一般に、次の状態を決定するためにＣＳＭ２２２のために必要とされる全ての情報を提供する。

【００６５】

解放フェーズでは、ＣＱＣ２２６がコンテキストテーブル２２０をスキャンし、コンテキストテーブルエントリがアクション（例えば、ＴＲＡＮＳＦＥＲ＿ＤＡＴＡ状態又はＰＲＥＦＥＴＣＨ＿ＤＡＴＡ状態）を待っている状態であるコンテキストテーブルエントリにエンカウントするとき、ＣＳＭ２２２はＣＱＣ２２６によって呼び出される。ＴＲＡＮＳＦＥＲ＿ＤＡＴＡ状態又はＰＲＥＦＥＴＣＨ＿ＤＡＴＡ状態にエンカウントするとき、ＣＱＣ２２６はデータ転送の状態に関するデータパスからの情報（例えば、ＤＭ、ＤＤＭ、又はＤＴＰのいずれか）を待つ。どちらにしても、ＣＳＭ２２２は、一般に、懸案のコンテキストテーブルエントリのための次の状態を決定するために呼び出される。ＣＳＭ２２２は、コンテキストテーブルエントリが完了状態（例えば、ＣＯＭＰＬＥＴＥＤ又はＣＯＭＰＬＥＴＥＤ　ＷＩＴＨ　ＥＲＲＯＲ）に移行するときに、ダイ管理モジュール２０６に通知するための役割も担う。

【００６６】
図９を参照すると、図６のフラッシュ動作マネージャ（ＦＯＭ）２０８の実施の形態の一例を図示するブロック図が示される。一例では、ブロック２０８は、５つのサブモジュールにより実現される。例えば、ブロック２０８は、ブロック（又は回路）２４０、ブロック（又は回路）２４２、ブロック（又は回路）２４４、ブロック（又は回路）２４６、及びブロック（又は回路）２４８を備える。回路２４０から２４８は、ハードウェア、ファームウェア、ソフトウェア、ハードウェアの組み合わせ、ファームウェア及び／又はソフトウェア、又は他のインプリメンテーションとして実現されるモジュール及び／又はブロックを表わす。ブロック２４０は、一例では、コマンドアービタ（ＣＡ）を実現する。ブロック２４２は、一例では、データ転送アービタ（ＤＴＡ）を実現する。ブロック２４４は、一例では、フラッシュ動作フォーマッタ（ＦＯＦ）を実現する。ブロック２４６は、一例では、ナノシーケンサを実現する。ブロック２４８は、一例では、制御状態マシン（ＦＯＭＣＳＭ）を実現する。

【００６７】
データ転送アービタ２４２は、一般に、データフローマネージャ２１４にフラッシュ動作マネージャ２０８を接続する。フラッシュ動作フォーマッタ２４４は、一般に、フラッシュバスコントローラ２１２にフラッシュ動作マネージャ２０８を連結する。制御状態マシン２４８は、一般に、コンテキストマネージャ２０４にフラッシュ動作マネージャ２０８を連結する。コマンドアービタ２４０は、一般に、フラッシュ動作フォーマッタ（ＦＯＭ）２４４と制御状態マシン２４８との間に接続される。データ転送アービタ２４２は、一般に、フラッシュ動作フォーマッタ２４４と制御状態マシン２４８との間に連結される。ナノシーケンサ２４６は、一般に、フラッシュ動作フォーマッタ２４４と制御状態マシン２４８との間に連結される。コマンドアービタ２４０は、一般に、適用するためのコマンドのためのコンテキストマネージャのコンテキストテーブルをスキャンし、その後、フラッシュバスコントローラ（ＦＢＣ）２１２に信号を送信するためにフラッシュ動作フォーマッタ（ＦＯＦ）２４４と通信する。

【００６８】
図１０を参照すると、本発明の実施の形態によるフラッシュメディアコンテキストレイアウト３００の一例を図示する図が示される。フラッシュトランザクションのそれぞれは、一般に、コンテキストによって表わされる。コンテキストは、フラッシュバンクを有するトランザクションを実行するためにシステムハードウェアによって用いられる全ての情報を含み、及び／又はシステムバッファの位置に又はその位置からデータを移動させるデータ構造である。コンテキストは、一般に、ファームウェアによって構成され、コンテキストの内容が事実上位置付けられる場所にバッファコントローラ（ＢＣ）へのポインタとしてフラッシュレーンコントローラ（ＦＬＣ）に送信される。ファームウェアは、フラッシュ動作のより大きなハードウェア自動化を許可するためにこれらのコンテキスト（例えば、コンテキストリスト）のリンクリストを構成することを決定する。一般に、管理されたフラッシュユニット（例えば、ダイ、ＬＵＮ等）ごとに１つのコンテキストリストがある。

【００６９】
一般に、通常動作の間に、完全なコンテキストの部分のみがフラッシュトランザクションを実行するためにＦＬＣによって用いられる。従って、ＦＬＣによって用いられる部分のみが、ＦＬＣのコンテキストキャッシュに格納される。通常動作の間にＦＬＣのコンテキストキャッシュに格納される完全なコンテキストの部分は、一般に「フラッシュコンテキスト」と指称される。フラッシュメディアコンテキストレイアウト３００から観察することができるとき、フラッシュコンテキストは、一例では、完全な１５のダブルワードコンテキストの第１の４つのダブルワードのみを備える。

【００７０】
一例では、本発明の実施の形態によって実現されたコンテキストは、以下の内容及び／又はフィールドを備える：フラッシュ動作フィールド（フラッシュローアドレスフィール

ド）、チャンク記述ポインタ／コピーバックローアドレスフィールド、コンフィギュレーションビットフィールド、ネクストコンテキストポインタフィールド、ＤＭＡスキップマスクフィールド、データバッファポインタフィールド、オルタネートデータバッファポインタフィールド、ロジカルブロックアドレス（ＬＢＡ）フィールド、状態フィールド、フラッシュユニット数（ＦＵＮ）フィールド、メタデータ又はコンフィギュレーションデータポインタのいずれかを含むフィールド、メタデータ、及び／又はフラッシュコンフィギュレーションデータ、メタデータ又はフラッシュコンフィギュレーションデータのいずれかを含むフィールド、６４ビットＬＢＡモードが用いられる場合にメタデータ又はロジカルブロックアドレス（ＬＢＡ）の上部のいずれかを含むフィールド、及びコンテキストのために誤り訂正符号化（ＥＣＣ）を含むフィールド。フラッシュ動作フィールドは、動作コードを備える。一例では、動作コードは８ビットを備える。ハードウェアに実行するための動作を通信するために、動作コードはファームウェアによって生成される。一般に、動作は、フラッシュアレイに少なくとも１つのアクセスを含む。ファームウェアに使用可能な基本動作コードは、例えば、ＲＥＳＥＴ、ＳＹＮＣＨＲＯＮＯＵＳ＿ＲＥＳＥＴ、ＲＥＡＤ＿ＩＤ、ＲＥＡＤ＿ＰＡＲＡＭＥＴＥＲ＿ＰＡＧＥ、ＧＥＴ＿ＦＥＡＴＵＲＥＳ、ＳＥＴ＿ＦＥＡＴＵＲＥＳ、ＲＥＡＤ＿ＰＡＧＥ、ＰＲＯＧＲＡＭ＿ＰＡＧＥ、ＥＲＡＳＥ＿ＢＬＯＣＫ、ＣＯＰＹＢＡＣＫ＿ＲＥＡＤ／ＰＲＯＧＲＡＭ、ＲＥＡＤ＿ＳＴＡＴＵＳ、ＲＥＡＤ＿ＳＴＡＴＵＳ＿ＥＮＨＡＮＣＥＤ、ＰＲＯＣＥＳＳＯＲ＿ＣＯＮＴＲＯＬ＿ＭＯＤＥ、ＭＵＬＴＩＰＬＡＮＥ　ＰＲＯＧＲＡＭ　ＰＡＧＥ、及びＭＵＬＴＩＰＬＡＮＥ　ＰＲＯＧＲＡＭ　ＥＮＤを含む動作を表わす値を備える。

【００７１】

フラッシュローアドレスフィールドは、フラッシュメモリにアクセスのページのローアドレスを含む。一例では、フラッシュローアドレスフィールドは、ページアドレスと連結されたブロックアドレスを含む。フラッシュローアドレスは、動作の３つの（３）ローアドレスサイクルの間にフラッシュアレイに送信される。消去動作のために、フラッシュメモリのアクセスのブロックアドレスのみが送信される。ちょうど１バイトのアドレスを用いるＲＥＡＤ＿ＩＤ動作のために、バイトは、一例では、フラッシュローアドレスの左端のバイト（ＭＳＢ）に配置される。フラッシュ動作がＰＲＯＣＥＳＳＯＲ＿ＣＯＮＴＲＯＬ＿ＭＯＤＥであるとき、フラッシュローアドレスフィールドもソフトコンテキストポインタとして用いられる。プロセッサ制御モード動作に遭遇するとき、ＦＬＣはソフトコンテキストのシステムバッファアドレスとしてフラッシュローアドレスフィールドの値を用いる。

【００７２】

ＤＭＡスキップマスクフィールドは、システムバッファに又はそのシステムバッファからデータを移動させるとき、ビットの状態（例えば、「１」又は「０」）に基づいて、ページのセクタ（リージョン）が転送又はスキップされることを許可するビットを含む。一例では、ＤＭＡスキップマスクフィールドは、アクティブロースキップマスク又はフォワードマスク（例えば、転送を示す「１」、及びページのそれぞれのスキップを示す「０」）として実現される。ＤＭＡスキップマスクフィールドは、少数のセクタのトランザクションのために共通の読み取り／修正／書き込み（ＲＭＷ）動作のために主としてｏを用いる。例えば、８Ｋページデバイスでは、ページは、通常、１６の（１６）セクタに分割される。ＤＭＡスキップマスクフィールドの特定のビットが１であるとき、対応するセクタは、一例では、システムバッファに書き込まれるか、又はシステムバッファから読み取られる。ビットが０であるとき、対応するセクタは、システムバッファへの書き込み又はシステムバッファからの読み取りが省略される。一例では、データが２つの個別のソースから書き込まれることを許可するモードもある：未変更のページデータを含むバッファチャンク、及び最新のページデータを含むバッファチャンク。それらの場合のために、１つの状態（例えば、「０」）はセクタが最新のページデータ含むチャンクから書き込まれることを示すために用いられ、別の状態（例えば、「１」）は、セクタが未変更のページデータ含むチャンクから書き込まれることを示すために用いられる。ビット位置も重要である

。例えば、左端のビット（ＭＳＢ）は、一般に、最初、又はページにおける最小の番号のセクタに対応するビットである。右端のビット（ＬＳＢ）は、一般に、ページにおける最大の番号のセクタに対応するビットである。ページで１６未満のリージョンがある場合、構わず右端のビットが処理される。しかしながら、他の配置は特定の実施の形態のその設計基準を満たすために実現される。一例では、ボーズチャウドゥーリーホッケンガム（ＢＣＨ）誤り訂正コード（ＥＣＣ）が実現されるとき、マスクの細分性は１つの（１）情報ワードである。

【００７３】

ネクストコンテキストポインタフィールドは、ファームウェアによって構築されたコンテキストリスト（例えば、コンテキストのリンクリスト）のネクストコンテキストへのポインタを備える。「ネクストコンテキストポインタ」がＦＬＣにファームウェアによってプログラムされる「リストポインタの終わり」と等しい場合、ハードウェアは探索の終わりに達したと仮定する。ハードウェアは、「リストポインタの終わり」値がリンクリストを横断し続ける前に変化するのを待つ。ハードウェアによるリスト探索を同様に停止する「ヌルポインタ」値も定義される。

【００７４】

一例では、チャンク記述子ポインタ／コピーバックローアドレスフィールドは、一般に、コピーバックコマンドのために書き込まれるためにキャッシュされたデータのために、フラッシュメモリのページのローアドレスを提供する。ローアドレスは、ブロックアドレスとページアドレスの連結である。別の一例では、コピーバックフラッシュローアドレスフィールドは、非コピーバックコンテキストのための未使用のフィールドと共有される。例えば、チャンク記述子ポインタ／コピーバックローアドレスフィールドは、バッファ割付けマネージャ（ＢＡＭ）アシストフィールドを提供するために用いられる。ＢＡＭアシストフィールドは、バッファ割付けマネージャ（ＢＡＭ）によって管理されたデータバッファの内部のデータチャンクのための記述子に対するポインタを備える。

【００７５】

チャンク記述子フィールドは、キャッシュ管理のために用いられる。チャンク記述子は、チャンクアドレス（例えば、システムバッファの）、有効ビット、ダーティビット、転送未解決カウント、状態、ＬＢＡ、及びバッファ管理に用いられるポインタを含む。ＤＭＡが完了した後、チャンク記述子ポインタは、一般に、フラッシュメディアコントローラ（ＦＭＣ）のＤＭＡマネージャによってＢＡＭに送信される。チャンク記述子ポインタは、フラグ（例えば、ＢＡＭルックアップ要求ビット）が設定された場合、ＤＭＡを開始するため、バッファデータポインタを取得するためにＢＡＭに送信される。チャンク記述子ポインタ／ＢＡＭアシストフィールドは、一般的なＢＡＭアシストフィールドとしてファームウェアによってセットアップされ、コンテンツは、ＢＡＭシーケンサのスペシフィックプログラミングに依存するファームウェアによって決定される。一般に、チャンク記述子ポインタ／ＢＡＭアシストフィールドは、ハードウェアに完全にトランスペアレントである。

【００７６】

データバッファポインタフィールドは、一般に、フラッシュメディアからバッファコントローラに／バッファコントローラからフラッシュメディアに送信される実データにポインタを提供する。データバッファポインタフィールドは、コンテキストが最初にフェッチされるとき、コンテキストに送信するか、又はＢＡＭルックアップ要求ビットが設定される場合にＢＡＭルックアップによるＢＡＭによってデータバッファポインタフィールドは満たされる。データバッファポインタフィールドは、一般に、チャンクの第１のバイトのアドレス、必ずではなくチャンクの第１の有効なＬＢＡを提供する。

【００７７】

オルタネートデータバッファポインタフィールドは、フラッシュメディアからバッファコントローラに／バッファコントローラからフラッシュメディアに送信されるデータに対するポインタである。オルタネートデータバッファポインタフィールドは、動作がバッフ

ァから又はそのバッファに複数のリソース又はデスティネーションを用いるとき用いられる。オルタネートデータバッファポインタは、読み取り／修正／書き込み（ＲＭＷ）動作のために主に用いられる。ホストがフルページよりも小さい動作を実行するとき、ホストは、ページをアップデートするためにＲＭＷ動作を終える。オルタネートデータバッファポインタは、読み取り動作がテンポラリチャンクで古いページデータを格納するために用いられる。その後、書き込み動作が実行されるとき、スキップマスクは未変更のページデータを含むメディアチャンク又はアップデートするためのデータを含むホストチャンクのいずれかからセクタ（又はリージョン）の書き込みソースを選ぶために用いられる。

【００７８】

メタデータフィールドは、一般に、フラッシュページに関連するメタデータ（例えば、ＬＢＡ等の管理情報、順序番号付、不良ブロックインジケータ等）を備える。全てのメタデータがコンテキストの内部で適合するであろうことが一般に期待される。そうでなければ、メタデータバッファポインタは、メタデータがシステムバッファに格納されるか、システムバッファから検索されるように、メタデータ自体の場所のコンテキストに含まれる。書き込みのために、メタデータフィールドは、一般に、ファームウェアによって占められ、ページへのハードウェアによって挿入される。読み取りには、読み取りページからファームウェアがメタデータの内容を要求することが大抵の場合望ましい。これらの場合のために、コンテキストは、コンテキストにページからメタデータを読み込むように構成される。一例では、メタデータのバイトのサイズは、メタデータサイズレジスタによって決定される。

【００７９】

フラッシュコンフィギュレーションデータフィールドは、一般に、メタデータフィールドとバイト位置を共有し、一般に、限定されたバイト数が転送される（例えば、ＲＥＡＤ＿ＩＤ、ＧＥＴ＿ＦＥＡＴＵＲＥＳ、ＳＥＴ＿ＦＥＡＴＵＲＥＳ、ＲＥＡＤ＿ＳＴＡＴＵＳ等）動作のために用いられる。そのような転送に関連するバイト数は一般に小さいので（例えば、ＳＥＴ＿ＦＥＡＴＵＲＥＳとＧＥＴ＿ＦＥＡＴＵＲＥＳは４バイトを用いる；ＲＥＡＤ　ＩＤは一般に５バイトを用いる；ＲＥＡＤ＿ＳＴＡＴＵＳはわずか１バイトを用いる）、データは別々のバッファ位置にコンテキストに単に転送される。そのような転送に関連するデータがコンテキストに割り当てられたスペースより大きくなる場合、フラッシュコンフィギュレーションデータポインタフィールドが用いられる。ＧＥＴ＿ＦＥＡＴＵＲＥＳとＳＥＴ＿ＦＥＡＴＵＲＥＳコマンドは、常に４バイトを用いることが想定される。ＲＥＡＤ＿ＩＤコマンドのために必要とされるバイト数は、一例では、ＲＥＡＤ　ＩＤコマンドレジスタのための複数の読み取りバイトから取得される。フラッシュコンフィギュレーションデータフィールドがコンフィギュレーションデータのために用いられるとき、７つのダブルワードの全てはハードウェアによってアップデートされ、ファームウェアは単に適切なバイトを読み取る必要がある。

【００８０】

メタデータバッファポインタフィールドは、ページメタデータ（例えば、フラッシュページのユーザデータに含まれる管理データ）の位置のアドレスを備える。メタデータバッファポインタフィールドは、全てのメタデータがコンテキスト自体に格納されることが予想されるとき、多くのアプリケーションのために省略される。メタデータバッファポインタフィールドは、メタデータが外部システムバッファから検索されるとき用いられる。メタデータバッファポインタフィールドは、一般に、メタデータがコンテキストにではなく外部に格納される場合にのみ用いられる。一例では、コンフィギュレーションビットは、メタデータが外部に又はコンテキストの内部に格納されるか否かを指定するように実現される。一例では、外部システムバッファは、フラッシュメディアコントローラ（ＦＭＣ）を含むチップの外側にあるメモリ（例えば、ＤＤＲ　ＲＡＭ）として実現される。別の一例では、外部システムバッファは、ＦＭＣ　ＩＰの外側にあるオンチップＲＡＭとして実現される。なお別の一例では、外部システムバッファは、ＦＭＣ　ＩＰであるメモリを備える。一般に、外部システムバッファは、コンテキストの外側にあるあらゆるストレージ

である。

【００８１】

フラッシュコンフィギュレーションデータポインタフィールドは、一般に、取得されるか又はフラッシュユニットに書き込まれる必要があるコンフィギュレーションデータの位置のアドレスを提供する。ＲＥＡＤ＿ＩＤ、ＧＥＴ＿ＦＥＡＴＵＲＥＳ、及びＳＥＴ＿ＦＥＡＴＵＲＥＳコマンドに関連するフラッシュコンフィギュレーションデータは、メタデータがデータトランザクションのためにコンテキストフィールドに割り当てられるとき、コンテキスト自体に格納される。しかしながら、メタデータの場合のように、データサイズがコンテキストのフラッシュコンフィギュレーションデータのために割り当てられたスペースより大きくなる場合、システムバッファのデータにポインタはアクセスのために用いられる。

【００８２】

ロジカルブロックアドレス（ＬＢＡ）フィールドは、一般に、ページにおける第１のデータリージョンのＬＢＡを提供する。ＬＢＡは、一般に、システムバッファとフラッシュのセクタのそれぞれのためのデータ保護に符号化される。ページがＬＢＡの隣接するグループであるので、グループの第１のＬＢＡのみがコンテキストの一部として用いられる。ＬＢＡは、セクタページのためのバッファＣＲＣをシードするために用いられる。ＬＢＡは、またメタデータのＬＢＡ部分に対して選択的にチェックされる。例えば、コンフィギュレーションビットは、メタデータのＬＢＡ部分に対してＬＢＡをチェックするべきか否かを選択するために実現される。例えば、ＬＢＡは、コンフィギュレーションビットが設定されるとき、メタデータのＬＢＡ部分に対してチェックされる。一例では、下位３２ビットのみがシーディングのために用いられる。

【００８３】

フラッシュユニットナンバー（ＦＵＮ）フィールドは、コンテキストが適用されるフラッシュユニットを決定するために、識別子としてファームウェアによって用いられる。決定はハードウェアにトランスペアレントである。フラッシュユニットナンバーフィールドは、コンテキストがハードウェアによって消費された後、識別子がファームウェアに送信されれば、ファームウェアがコンテキストを一致させるように、専らファームウェアによって管理目的のために用いられる。

【００８４】

コンフィギュレーションビットフィールドは、一般に、コンテキストを構成し、ハードウェアフローの様々なポイントでコンテキスト（及びコンテキストが表わす転送）のディスポジションを決定するために用いられる全てのビットを含む。一例では、コンテキストのコンフィギュレーションビットフィールドで使用可能なコンテキストビット及びフィールドは、限定されるものではないが、以下のものを含む：消費コンテキストマネージャ割り込みイネーブル；フラッシュインタフェースで部分的なコマンドを用いて、ＤＭＡスキップマスクフィーチャの実行を有効／無効にする部分的なコマンドのイネーブルビット；システムバッファへのＤＭＡディスエーブル；ＦＬＣローカルバッファに又はそのバッファからの全てのデータ転送ディスエーブル；スクランブラ機能を有効／無効にするスクランブラ機能イネーブル；ＥＣＣエラー検出割り込みイネーブル；読み取りデータの復号化を有効／無効にするか、又は書き込みデータにパリティを追加するＥＣＣマクロバイパス；バッファＣＲＣイネーブル；システムデータバッファ、読み取り上のシステムバッファ又はコンテキストにメタデータを転送せず、書き込み上の全てのソースからのメタデータの挿入を防ぐ（例えば、メタデータフィールドブランクを残す）イグノアメタデータ信号（ビット）；ＤＭＡ（例えば、読み取り上のデータバッファにメタデータを転送し、書き込み上のデータバッファからメタデータを受け取る）のためにユーザデータを有するメタデータを保持するための信号（ビット）；メタデータの「起動ＬＢＡ」フィールドがコンテキストの始まるＬＢＡフィールドに対してチェックされるか否かを示す信号（ビット）；バッファにＥＣＣパリティフィールドを保持するか否か（例えば、読み取り上のシステムバッファへのＥＣＣパリティバイトの転送、及び書き込み上のシステムバッファからの

フラッシュへのＥＣＣの転送）を示す信号（ビット）；管理されたデータバッファ（例えば、どれだけのデータがクワンタムバーストのシステムバッファに転送されるかを判断するために用いられる）の代わりに平面バッファ（例えば、メモリの管理されていないエリアに／からデータを転送する）を用いるか否かを示す信号（ビット）；セクタデータ長（例えば、ビットがクリアな場合に、セクタデータ長は予約エリアセクタ長コンフィギュレーションから決定される）を決定するためにユーザデータセクタ長コンフィギュレーションを用いるか否かを示す信号（ビット）；フラッシュコンフィギュレーションデータを転送する（例えば、コンテキストにフラッシュデータを読み取り、又はコンテキストからのフラッシュデータを書き込む）か否かを示す信号（ビット）；システムバッファに又はそのシステムバッファから完全な生のフラッシュページを転送するか否かを示す信号（ビット）；オルタネートバッファ（例えば、セクタの転送を省略する代わりにオルタネートデータバッファポインタによってポイントされたバッファチャンクに／から転送されるマークセクタ（リージョン）に用いられるスキップマスクをもたらす）を有するスキップマスクを用いるか否かを示す信号（ビット）；スクランブラが有効になるときにリージョンのそれぞれのためのスクランブラシードを定義するフィールド。システムバッファに又はそのシステムバッファから完全な生のフラッシュページを転送するか否かを示す信号は、その性質によって大部分の他のオプションと相互排除である。システムバッファに又はそのシステムバッファから完全な生のフラッシュページを転送するか否かを示す信号は、一般に、他のビットセッティングを無視する。システムバッファに又はそのシステムバッファから完全な生のフラッシュページを転送する特徴は、一般に、ファームウェアがページの如何なるリージョンにも属さない未使用エリアを取得することを許可する。転送フラッシュコンフィギュレーションデータ信号は、ＲＥＡＤ　ＩＤ、ＧＥＴ　ＦＥＡＴＵＲＥＳ、及びＳＥＴ　ＦＥＡＴＵＲＥのような動作のようなコマンドのために用いられる。データのサイズは、コンフィギュレーションデータ長レジスタによって決定される。データは、メタデータによって通常占められるコンテキストの位置に現われる。

【００８５】

状態フィールドは、一般に、ファームウェアへの現状状態に用いるビットを備える。一例では、状態フィールドは、プログラム／消去動作のためにパス／フェイル状態を含む。別の一例では、状態フィールドは、読み取り動作のＥＣＣロジックによって検出されたエラー回数も含む。しかしながら、フィールドの他の用途は、特定の実施の形態のその設計基準を満たすために実現される。一例では、状態フィールドは、フラッシュページにおけるエラー回数及びコンテキストマネージャのための完了コード（例えば、クリーン、プログラム／消去エラー、訂正された読み取りエラー、訂正できない読み取りエラー、システムバッファからのプリフェッチのバッファＣＲＣエラー、動作タイムアウト、ＬＢＡ／メタデータミスマッチ、違法動作等）を示すように構成される。

【００８６】

先に述べたように、コンテキストは、コマンドの実行を決定するデータ構造である。コンテキストは、ファームウェアとハードウェアとの間の通信のユニットである。一般的なコンテキストフローは、図２及び３に関連して、以下の通り概説される。ファームウェアは、メモリポインタに加えてＦＭＣ１０２のプロセッサインタフェースロジック（ＰＩＬ）１５０のコンテキスト構造バッファ（ＣＣＢ）にコンテキストを書き込む。複数のコンテキストは、コンテキスト構造の内部のネクストポインタフィールドを用いることによってコンテキストリストのファームウェアによってリンクされる。ハードウェアは、コンテキストアップデートポート（ＣＵＰ）１６８によってメモリポインタに関連する位置にコンテキストのメモリ書き込みを実行する。ファームウェアは、例えば、フラッシュレーンコントローラ（ＦＬＣ）１５６ａ～１５６ｎのそれぞれのダイ管理モジュール（ＤＭＭ）２０６におけるダイ管理レジスタによる特定のダイの実行を有効にする。ダイ管理レジスタの別々のセットは、ダイのために存在する。ＤＭＭ２０６は、フラッシュレーンコントローラ（ＦＬＣ）１５６ａ～１５６ｎのそれぞれのコンテキストマネージャ（ＣＭ）２０４にレジスタの情報を通信する。その後、ＣＭ２０４は、コンテキスト訂正ポート（ＣＲ

Ｐ）１６６、及びフラッシュレーンバスのそれぞれでコマンドの実行のためのスケジュールを通じて処理するためにメモリから有効にされたコンテキストをフェッチする。ＣＭ２０４は、一般に、コマンドをスケジュールし、データ指向の動作のためにデータＤＭＡマネージャ（ＤＤＭ）ブロック１５２を指示する。ＤＤＭブロック１５２は、一般に、コンテキストからＣＲＰ１６６を通じてデータパラメータを抽出するためにメモリからコンテキストをフェッチする。ＤＤＭ１５２及びＣＭ２０４によるコンテキストの無事完了に際して、状態は消費コンテキストマネージャ（ＣＣＭ）１６４にアップデートされる。完了に際して、割り込みはハードウェアによって生成され、ファームウェアは与えられたコンテキストのために完了状態を読み取る。

【００８７】

ブロック１７２ａ～１７２ｎ及び１７４ａ～１７４ｎを備えるデータＤＭＡインタフェースポート（ＤＤＩＰ）は、一般に、ＤＴＰ１５８ａ～１５８ｎとシステムバッファコントローラ、及び関連するコンテキストとの間でリアルタイムデータをルーティングする役割を担う。データ転送のそれぞれは、１つ以上の領域を備える。リージョンは、１つ以上のエリアを備える。実際のフォーマットは、関連するコンテキストと同様にＦＭＣ（例えば、レジスタを用いて）からのコンフィギュレーションセッティングによって決定される。セッティングは、転送ごと、リージョンごと、及びエリアごとに基づいて分類される。ルーティングは、追加コンフィギュレーションセッティングと同様にデータ転送フォーマットのエリアタイプに従って決定される。ルーティングは、一般に、機能性をパッドしストリップすることを含む。データ転送のそれぞれは、全てのコンテキストｄｗｏｒｄを含む。従って、ＤＤＭ１５２は、ＤＤＩＰによってコンテキストアクセスを開始し、ＤＤＩＰは実データ移動に先立ってそれ自身のコンテキストキャッシュエリアに全てのコンテキストｄｗｏｒｄを読み取る。関連するコンテキストが検索されれば、ＤＤＩＰは適切なコンフィギュレーションセッティングを有するその内部制御回路をプログラムし、ＤＴＰ１５８ａ～１５８ｎのそれぞれにいくらかのコンフィギュレーション情報を提供する。その後、ＤＤＩＰは、リージョンのそれぞれの内部のエリアのそれぞれを追跡し、適切なルーティングのためのロジックをセットアップする。実データ転送の間に、データがコンテキストにリダイレクトされる場合、ＤＤＩＰはライトモードにコンテキストアクセスを切り替える。転送が完了したとき、ＤＤＩＰは転送とコンテキストの両方のアクセスが終わったことをＤＤＭ１５２に通知する。ＤＤＩＰはエラー検査を実行する。例えば、ＤＤＩＰは、システムバッファＣＲＣチェッキング及び／又は受信されたＬＢＡがコンテキストからの期待するＬＢＡとマッチするかベリファイを実行するように構成される。システムバッファＥＣＣバイトエラー（許可されたとき）と同様にこれらのチェックの結果は、一般に、ＤＤＭブロック１５２に送信される。ＤＤＭブロック１５２は、次に選択されたＦＬＣ１５６ａ～１５６ｎに情報を送信する。ＤＤＩＰもＣＲＣを生成し挿入するように構成される。全ユーザデータはリージョンのそれぞれから全てのユーザデータバイトを集めることによって構成される。同様に、全メタデータは、リージョンのそれぞれからメタデータバイトを集めることによって構成される。一例では、定義済みの上限に到達するまで、メタデータは集められる。一例では、ＤＤＩＰは、ＤＤＩＰがシステムバッファに／からユーザデータエリアの使用された部分のみ、及びコンテキストに／からメタデータエリアの使用された部分のみをルートするデフォルトコンフィギュレーションを有する。ＥＣＣエリアはストリップ／パッドされる。デフォルトコンフィギュレーションは複数の変化を提供するために修正される。例えば、ＤＤＩＰは（１）未使用のバイトを含む全ユーザデータエリアをそのまま保持し、（２）システムバッファに／からユーザデータエリアを有するメタデータエリアを保持し、（３）メタデータエリアをストリップ／パッドし、（４）ＥＣＣをそのまま保持し、及び／又は（５）全領域をそのままに保持する（例えば、ストリップ／パッドがない）ように構成される。

【００８８】

コンテキストレイアウト構造３００（図１０に関して上に記述した）は、ターゲットの如何なるページをプログラムするために用いられる。トップモジュールの間でのプログラ

ムデータフローの一例は以下の通り概説される。ファームウェアは、プロセッサロジックインタフェース（ＰＩＬ）１５０の内部のＣＣＢの関連するメモリポインタを有するプログラムコマンドを有するコンテキストをプログラムする。ＣＣＢは、ＣＵＰ１６８を通じてメモリへの書き込み動作を実行する。ファームウェアは、コンテキストの実行を許可するためにＦＬＣ１５６ａ～１５６ｎのそれぞれの内部のＤＭＭ２０６を有効にする。ＦＬＣ１５６ａ～１５６ｎのそれぞれの内部のＣＭ２０４は、メモリからＣＲＰ１６６を通じてコンテキストをフェッチし、フラッシュバスコントローラ（ＦＢＣ）１５４ａ～１５４ｎのそれぞれのコマンドのスケジューリングのために用いられるコンテキストの部分を処理する。ＦＬＣはコンテキストについてＤＤＭ１５２と通信し、ＤＤＭ１５２はＣＲＰ１６６を通じてコンテキストをフェッチし、ＤＴＰブロック１５８ａ～１５８ｎのそれぞれにデータチャンクのポインタを提供する。ＤＴＰブロック１５８ａ～１５８ｎのそれぞれは、コンテキストから復号化されたポインタによってメモリからデータを要求する。その要求は、ブロック１７２ａ～１７２ｎ及び１７４ａ～１７４ｎによって構成されたデータＤＭＡインタフェースポート（ＤＤＩＰ）を介して行われる。データＤＭＡインタフェースポート（ＤＤＩＰ）は、クワンタムバーストのメモリからデータを読み取る。

【００８９】

メモリから読み取られたデータは、一般に、ブロック１７２ａ～１７２ｎ及び１７４ａ～１７４ｎによって構成されたＤＤＩＰによってＤＴＰに送信される。ＤＴＰでは、有効であれば、データはＣＲＣのためにチェックされ、有効であってＦＬＣ１５６ａ～１５６ｎのそれぞれに通過したのであれば、ＥＣＣはユーザデータに追加される。データの完全性が失敗する場合、状態はＣＣＭブロック１６４にアップデートされる。ＦＬＣ１５６ａ～１５６ｎのそれぞれのフラッシュ動作マネージャ（ＦＯＭ）２０８は、フラッシュバスコントローラ（ＦＢＣ）１５４ａ～１５４ｎのそれぞれのプログラムサイクルに関連するコマンド及びデータをスケジュールする。ＦＢＣ１５４ａ～１５４ｎのそれぞれは、フラッシュインタフェース仕様ごとにフラッシュバスインタフェースでプログラム動作を実行する。ＦＢＣ１５４ａ～１５４ｎのそれぞれは、動作の完了のためにフラッシュから状態を読み取る。ＦＢＣ１５４ａ～１５４ｎのそれぞれは、ＦＬＣ１５６ａ～１５６ｎのそれぞれと通信し、ＦＬＣ１５６ａ～１５６ｎのそれぞれは、動作の完了したときにＣＣＭ１６４と通信する。ＣＣＭ１６４は、ＣＣＭ１６４に関連するコンテキストレジスタに属するコンテキストが完了し、ファームウェアが検査のために準備ができていることをファームウェアに通知するためにファームウェアに割り込みをアサートする。その後、ファームウェアは、ＣＣＭ１６４からコンテキストを読み取る。ファームウェアがコンテキストを読み取り処理することを終了するとき、ファームウェアは別の消費されたコンテキストがロードされることを許可する信号（例えば、ビット）をアサートする。

【００９０】

基本的なフラッシュ読み取りデータフローは以下の通り説明される。コンテキスト構造は、ターゲットのあらゆるページを読み取るために用いられる。ファームウェアは、プロセッサロジックインタフェース（ＰＩＬ）１５０の内部のＣＣＢの関連するメモリポインタによって読み取りコマンド有するコンテキストをプログラムする。ＣＣＢは、ＣＵＰ１６８を通じてメモリに書き込みを実行する。ファームウェアは、コンテキストの実行を許可するために、ＦＬＣ１５６ａ～１５６ｎのそれぞれの内部のＤＭＭ２０６を有効にする。ＦＬＣ１５６ａ～１５６ｎのそれぞれの内部のＣＭ２０４は、メモリからＣＲＰ１６６を通じてコンテキストをフェッチし、ＦＢＣ１５４ａ～１５４ｎのそれぞれのコマンドのスケジューリングのために必要なコンテキストの部分を処理する。ＦＬＣ１５６ａ～１５６ｎのそれぞれは、ターゲットへの読み取りコマンドの実行のためにＦＢＣ１５４ａ～１５４ｎのそれぞれと通信する。ＦＬＣ１５６ａ～１５６ｎのそれぞれは、コンテキストについてＤＤＭ１５２と通信し、ＤＤＭ１５２は、ＣＲＰ１６６を通じてコンテキストをフェッチし、ＤＴＰブロック１５８ａ～１５８ｎのそれぞれにデータチャンクのポインタを提供する。

【００９１】

フラッシュからＦＢＣ１５４ａ～１５４ｎのそれぞれを介してＤＴＰ１５８ａ～１５８ｎのそれぞれにデータを受信するに際して、ＤＴＰ１５８ａ～１５８ｎのそれぞれは、コンテキストから復号化されたポインタによってメモリに書き込まれるデータを要求する。ＤＴＰ１５８ａ～１５８ｎのそれぞれは、ブロック１７２ａ～１７２ｎ及び１７４ａ～１７４ｎによって構成されたデータＤＭＡインタフェースポート（ＤＤＩＰ）を介して行う。ＤＤＩＰは、クワンタムバーストのメモリにデータを書き込む。ＤＴＰでは、有効であれば、データはＥＣＣのためにチェックされ、有効であれば、ＣＲＣはユーザデータに追加され、データはＤＤＩＰに送信される。データの完全性が失敗する場合、状態はＣＣＭブロック１６４にアップデートされる。読み取り動作の状態は、動作の完了したときにＣＣＭブロック１６４に提供される。ＣＣＭブロック１６４は、ＣＣＭブロック１６４のコンテキストレジスタに属するコンテキストが完了し、ファームウェアが検査のために準備ができていることをファームウェアに通知するためにファームウェアに割り込みをアサートする。ファームウェアは、ＣＣＭブロック１６４からコンテキストを読み取る。ファームウェアがコンテキストを読み取り処理することを終了するとき、ファームウェアは別の消費されたコンテキストがロードされることを許可する信号（例えば、ビット）をアサートする。

【００９２】

図１１を参照すると、本発明の実施の形態によるフラッシュページのリージョンパーティションの一例を図示する図が示される。ＮＡＮＤ型フラッシュメモリーは、一般に、ユーザデータを含むページに編成される。一般に、大部分の共通ページサイズは８ＫＢである。ユーザデータの８ＫＢに加えて、ＮＡＮＤ型フラッシュメモリーは、複数の目的のために用いられる付加的なバイトを含む予備のエリアを有する。最初に、予備のエリアは、一般に、ページメタデータと指称されるフラッシュ管理情報（例えば、ＬＢＡ、順序番号付、不良ブロックインジケータ等）を含む。次に、予備のエリアは、誤り訂正コード（ＥＣＣ）パリティ及び／又はエンドツーエンドデータの完全性チェック（ＥＤＣ）情報を含む。ＥＣＣパリティは、ユーザデータとメタデータのデータ保護（例えば、エラーの検出及び訂正）のために用いられる。一例では、８ＫＢユーザデータとは別に、メタデータ、ＥＣＣ、及びＥＤＣとの間で共有される５１２バイトの付加的な予備のエリアがある。

【００９３】

物理的ページは、複数のリージョンを備える。本発明の実施の形態によるフラッシュメディアコントローラは、従来のページリージョンを用いるが、リージョンが設定可能であることを許可する（例えば、レジスタ等を介して）。一例では、リージョンは、リージョンレイアウト４００を用いて配列される。別の一例では、リージョンは、リージョンレイアウト４１０を用いて配列される。

【００９４】

リージョンレイアウト４００が用いられるとき、リージョンのそれぞれは、ＥＣＣ４０６のリージョンに続くメタデータ４０４のエリアに続くユーザデータフィールド４０２を含む。ＥＣＣ４０６は、一般に、ユーザデータ４０２及びメタデータ４０４の両方を保護する。リージョン４００のサイズは、一般に、ユーザデータ４０２及びメタデータ４０４のサイズとして設定可能である。その後、ＥＣＣエリア４０６のサイズは、ユーザデータ４０２及びメタデータ４０４のサイズから続く。本発明の実施の形態によって実現されたフラッシュメディアコントローラは、一般に、従来のページフォーマット構造におけるような隣接したページリージョンをサポートする。隣接したページリージョンは、複数のリージョンの隣接を表わす。しかしながら、リージョンは２のべき乗のサイズに拘束されない；リージョンはあらゆるサイズである。リージョンがあらゆるサイズであることを許可することによって、本発明の実施の形態によって実現されたフラッシュメディアコントローラは、より大きいセクタ、ＤＩＦ、ＣＲＣ、及び／又は付加的なメタデータをサポートする。一般に、サイズは全てプログラマブルである。

【００９５】

リージョンレイアウト４１０が用いられるとき、リージョンのそれぞれは、第１のフィ

ールド４１２及び第２のフィールド４１４を含む。第１のフィールド４１２は、一般に、生、スクランブル（可能であれば）、暗号化（可能であれば）されたホストセクタデータを含む。一例では、ユーザデータのサイズは、５１２／１Ｋ／２Ｋ／４Ｋバイトプラスｎｄｗｏｒｄである、ここで、ｎは、例えば、０～３２まで変動する。一例では、現在のＬＢＡは、スクランブラと暗号化のエンジンのためにシードするために用いられる。第２のフィールド４１４は、一般に、ページメタデータと指称されるフラッシュ管理情報（例えば、ＬＢＡ、順序番号付、不良ブロックインジケータ等）を含む。リージョンレイアウト４１０を用いて、メタデータが物理的ページに配置される２つの方法がある、（１）メタデータはリージョンにわたって分散される、（２）全メタデータは最後のリージョンにのみ配置される。メタデータがリージョンにわたって分散されるとき、メタデータは、一般に、リージョンの一部として扱われる。リージョンのサイズは、一般に、ユーザデータとメタデータのサイズとして設定可能である。一例では、ユーザデータとメタデータのサイズはプログラマブルである（例えば、レジスタを介して）。一般に、ファームウェアは、プログラムされたサイズが物理的ページサイズの範囲内であることを保証するための役割を担う。

【００９６】

図１２Ａ及び１２Ｂを参照すると、フラッシュページ構造の一例を図示する図が示される。本発明の実施の形態によって実現されたフラッシュメディアコントローラは、一般に、従来のページリージョンを用いるが、ページリージョンが設定可能であることを許可する（例えば、レジスタを介して）。一例では、フラッシュページ構造５００は、リージョンレイアウト４００を用いて実現される。別の一例では、複数のフラッシュページ構造５１０、５１５、５２０、５２５、５３０、５３５は、リージョンレイアウト４１０を用いて実現される。

【００９７】

一例では、メタデータは、ページ構造５００のリージョンのメタデータエリアにおいて左から右に送信される。例えば、８Ｋバイトのページサイズ及び５１２バイトの予備のエリアを有するフラッシュデバイスでは、５１２バイトのホストセクタサイズに最大５４４バイトを有するリージョンのそれぞれによって、１６のリージョンが格納される。一例では、リージョンのそれぞれは、ユーザデータの５１２バイト、メタデータの４バイト、及びＥＣＣのための２８バイトまでを備える。もし合計のメタデータカウントが２８バイトならば、メタデータの４バイトは第１の７つのリージョンのそれぞれのメタデータエリアに入り、９つの存続するメタデータエリア（例えば、リージョン７～１５に）に０が詰められるか、又は他の利用ために使用可能にされる。データＤＭＡインタフェースポートがページを処理しているとき、データＤＭＡインタフェースポートが、メタデータの挿入及び収集をどこで開始するべきか認識しており、リージョン境界でもあるコードワード境界がどこにあるか認識しているように、データＤＭＡインタフェースポートは、ページの処理におけるプログレスのトラックを維持する。リージョンサイズがプログラマブルな一方で、リージョンサイズはＥＣＣコーデックが働くことができる最大のコードワードより大きくなるべきでない。例えば、ＢＣＨ　ＥＣＣは、１Ｋ＋８０情報ワードのための４８ビットの訂正までサポートする。訂正ビットのそれぞれのために、１６ビットのパリティが追加される。

【００９８】

リージョンレイアウト４１０を用いて、様々なフラッシュページ構造５１０、５１５、５２０、５２５、５３０、５３５が実現される。一例では、フラッシュページは８つのリージョンに分割され、リージョンのそれぞれはリージョンレイアウト４１０を用いて構成される。フラッシュページの未使用の予備のエリアは、２つの方法で利用される。第１の例では、メタデータは、リージョン（構造５１０、５２０、５３０によって図示された）の一部としてリージョンにわたって分散される。第２の例では、全メタデータは、ページ（構造５１５、５２５、５３５によって図示された）の最後のリージョンにのみ配置される。第１の例（構造５１０及び５１５によって図示された）では、ＬＤＰＣの情報ワード

はホストセクタサイズに等しい。この一例では、リージョンは全て別々のＥＤＣを有し、ＥＣＣパリティ領域が付随される。ＥＤＣにリージョンの内部のホストセクタのＢＡがシードされる。第２の例（構造５２０及び５２５によって図示された）では、ＬＤＰＣの情報ワードはホストセクタサイズより大きい。この一例では、２つのリージョンごとに別々のＥＤＣを有し、ＥＣＣパリティが付随される。ＥＤＣにリージョンの内部の最後のホストセクタのＬＢＡがシードされる。第３の例（構造５３０及び５３５によって図示された）では、ＬＤＰＣの情報ワードはホストセクタサイズ以下である。この一例では、１つのリージョンの内部では、２つのＥＣＣパリティが付随され、１つのＥＤＣが付随される。ＥＤＣにリージョンの内部のホストセクタのＬＢＡがシードされる。ページリージョンが設定可能であるので、他のページ構造は特定の実施の形態のその設計基準を満たすために実現される。

【００９９】

図１３を参照すると、本発明の実施の形態によるプロセス６００を図示する流れ図が示される。処理（又は方法）６００は、一般に、本発明の実施の形態によるメタデータをハンドリングするためのステップを提供する。一例では、処理６００は、ステップ（又は状態）６０２、ステップ（又は状態）６０４、ステップ（又は状態）６０６、ステップ（又は状態）６０８、及びステップ（又は状態）６１０を備える。ステップ６０２では、処理６００は、コンテキストに格納される定義済みのメタデータサイズに対応する閾値を定義する。ステップ６０４では、処理６００は、要求がフラッシュデバイスからメタデータを読み取るか、又はフラッシュデバイスにメタデータを書き込むための要求のために待機する。読み取り又は書き込み動作が実行されるとき、処理６００は、ステップ６０６に移行する。ステップ６０６では、処理６００は、メタデータサイズが定義済みの閾値より大きいか否かを判断する。メタデータサイズが定義済みの閾値より大きいとき、処理６００は、ステップ６０８に移行する。メタデータサイズが定義済みの閾値以下であるとき、処理６００は、ステップ６１０に移行する。

【０１００】

ステップ６０８では、処理６００は、読み取りトランザクションに関連するコンテキスト構造のフラッシュデバイスから読み取られたメタデータを格納するか（例えば、コンテキストアップデートポート（ＣＵＰ）１６８を介して）、又は書き込みトランザクションに関連するコンテキスト構造からフラッシュデバイスにプログラムされるためのメタデータを検索する（例えば、コンテキスト訂正ポート（ＣＲＰ）１６６及びＢＣ＿ＲＤ＿Ｉ／Ｆを介して）。ステップ６１０では、処理６００は、読み取りトランザクションに関連するコンテキスト構造において定義されたメタデータポインタによって識別された位置（例えば、外部システムバッファの）のフラッシュデバイスから読み取られたメタデータを格納するか、又は書き込みトランザクションに関連するコンテキスト構造において定義されたメタデータポインタによって識別された位置（例えば、外部システムバッファの）からフラッシュデバイスにプログラムされるためのメタデータを検索する。例えば、プログラムコマンドの間に、メタデータはデータＤＭＡ読み取りインタフェースポート（ＤＤＲＩＰ）１７２ａ～１７２ｎのそれぞれ及びＢＣ＿ＲＤ＿Ｉ／Ｆを介して検索され、読み取りコマンドの間に、メタデータはデータＤＭＡ書き込みインタフェースポート（ＤＤＷＩＰ）１７４ａ～１７４ｎのそれぞれ及びＢＣ＿ＷＲ＿Ｉ／Ｆを介してアップデートされる。ステップ６０８又はステップ６１０のいずれかでは、トランザクションを完了するに際して、処理６００は、ステップ６０４にリターンする。

【０１０１】

本発明の実施の形態によるフラッシュメディアコントローラ（ＦＭＣ）は、一般に、システムバッファ及び／又はフラッシュメディアコントローラでメタデータストレージをハンドリングする様々な方法を提供する。本発明の実施の形態によるフラッシュメディアコントローラは、メタデータハンドリングのために以下の特徴を含む。メタデータ情報は、ページベースごとにコンテキストが定義されるコンテキストベースごとに定義される。メタデータサイズが定義済みの閾値以下であるとき、完了したメタデータはコンテキスト構

造の内部に格納される。フラッシュプログラムサイクルの間に、コンテキストからのメタデータ（例えば、ＣＲＰ１６６及びＢＣ＿ＲＤ＿Ｉ／Ｆを介して）は、フラッシュターゲットに格納され、読み取りサイクルの間に、フラッシュターゲットから読み取られたメタデータは、コンテキスト構造の後に格納される（例えば、ＣＵＰ１６８を介して）。メタデータサイズが定義済みの閾値より大きいとき、コンテキスト構造の内部のメタデータを格納する代わりに、メタデータポインタはコンテキストの内部に定義される。メタデータポインタは、フラッシュプログラムサイクルの間に外部システムメモリからメタデータを検索し（例えば、ＤＤＲＩＰのそれぞれ及びＢＣ＿ＲＤ＿Ｉ／Ｆを介して）、コンテキストによって指定されたメタデータポインタによって示された外部システムメモリの位置へのフラッシュ読み取りコマンドから読み取られたデータを格納するためにアドレスポインタを提供する（例えば、ＤＤＷＩＰのそれぞれ及びＢＣ＿ＷＲ＿Ｉ／Ｆを介して）。別の一例では、ＦＭＣは、メタデータの別の部分が外部システムメモリによってハンドリングされる間に、メタデータのある部分がコンテキストによってハンドリングされるように構成される。

【０１０２】

メタデータのサイズは、一般に、フラッシュトランザクションレイヤ（ＦＴＬ）によって定義される。一例では、ＦＴＬはおよそ１２バイトのメタデータを用いる。図１０に示されたコンテキスト３００の最大の使用可能な予備のエリアは、２８バイトである。ＦＴＬメタデータサイズパラメータ及びコンテキスト３００の予備のエリアパラメータの両方を考慮に入れて、２８バイトの定義済みの閾値が選択される。例えば、メタデータサイズが２８バイト以下であるとき、完了したメタデータは、フラッシュメディアコントローラのフラッシュレーンコントローラの内部のコンテキスト構造の内部に格納される。フラッシュプログラムサイクルの間に、コンテキストからのメタデータはフラッシュターゲットに格納され、読み取りサイクルの間に、フラッシュターゲットから読み取られたメタデータは、コンテキスト構造の後に格納される。メタデータサイズが２８バイトを超えているとき、コンテキスト構造の内部にメタデータを格納する代わりに、フラッシュレーンコントローラは、メタデータが格納される外部メモリの位置を指すコンテキストの内部のメタデータポインタを定義する。ページサイズがホストサイズの倍数である場合に（例えば、図１２に示されるように）、メタデータは、一般に、ホストユーザデータのそれぞれに分散される。例えば、１Ｋバイトのページサイズ及び１０２４バイトのホストサイズのために、フラッシュプログラムコマンドの間に、コンテキストからの２８バイトは、２つのホストユーザデータの間に分散される（例えば、ユーザデータのそれぞれにメタデータの１４バイト）：

ホストユーザデータ１（５１２バイト）＋メタデータ１（１４バイト）＋ＥＣＣ＋ホストユーザデータ２（５１２バイト）＋メタデータ２（１４バイト）＋ＥＣＣ。

【０１０３】

メタデータは、一般に、ＥＣＣの利用によって保護される。ＥＣＣの利用は、一般に、メタデータ部分を通じて、完全性チェック及び訂正を提供する。

【０１０４】

本発明の様々な信号は、一般に、「オン」（例えば、デジタルＨＩＧＨ、又は１）又は「オフ」（例えば、デジタルＬＯＷ、又は０）がある。しかしながら、信号のオン（例えば、アサートされた）及びオフ（例えば、デアサートされた）の状態の特定の極性は、特定の実施の形態のその設計基準を満たすために調節される（例えば、逆にされる）。

【０１０５】

図１～９、及び１３によって実行される機能は、従来のメインプロセッサ、デジタルコンピュータ、マイクロプロセッサ、マイクロコントローラ、ＲＩＳＣ（縮小命令型コンピュータ）プロセッサ、ＣＩＳＣ（複雑命令セットコンピュータ）プロセッサ、ＳＩＭＤ（単一命令多重データ）プロセッサ、信号プロセッサ、中央演算処理装置（ＣＰＵ）、算術論理演算装置（ＡＬＵ）、ビデオデジタル信号プロセッサ（ＶＤＳＰ）、及び／又は関連する技術分野における当業者に明らかであろうように本明細書の教示に従ってプログラム

された同様の計算機の１つ以上を用いて実現される。適切なソフトウェア、ファームウェア、符号化、ルーチン、命令、オペコード、マイクロコード、及び／又はプログラムモジュールもまた関連する技術分野における当業者に明らかであろうように本明細書の教示に基づいて熟練したプログラマによって容易に準備される。ソフトウェアは、一般に、機械実装のプロセッサの１つ以上によって１つの媒体又は複数の媒体から実行される。

【０１０６】

本発明は、更にＡＳＩＣ（特定用途向け集積回路）、プラットホームＡＳＩＣ、ＦＰＧＡ（フィールドプログラマブルゲートアレイ）、ＰＬＤ（プログラマブルロジックデバイス）、ＣＰＬＤ（コンプレックスプログラマブルロジックデバイス）、シーゲート、ＲＦＩＣ（高周波集積回路）、ＡＳＳＰ（特定用途専用標準品）、１つ以上のモノリシック集積回路、１つ以上のチップ又はダイに配置されたフリップチップモジュール及び／又はマルチチップモジュールによって、又はここで説明されるように従来の構成回路の適切なネットワークを相互に接続することによって実現され、それらの修正は、技術分野における当業者に容易に明らかである。

【０１０７】

本発明は、また本発明による１つ以上の処理又は方法を行うように機械をプログラムするために用いられる命令を含むコンピュータ製品である記憶媒体又は媒体及び／又は送信媒体又は媒体を含む。コンピュータ製品に包含される命令の機械による実行は、回路素子を取り巻く動作に加えて、入力データを記憶媒体で１つ以上のファイル、及び／又はオーディオ及び／又はビジュアルな描写のようなフィジカルオブジェクト又は実体の典型である１つ以上の出力信号に変換する。記憶媒体は、限定されるものではないが、フロッピー（登録商標）ディスク、ハードドライブ、磁気ディスク、光ディスク、ＣＤ－ＲＯＭ、ＤＶＤ、及び光磁気ディスクを含む如何なるタイプのディスク、及びＲＯＭ（リードオンリメモリ）、ＲＡＭ（ランダムアクセスメモリ）、ＥＰＲＯＭ（電気的プログラム可能リードオンリメモリ）、ＥＥＰＲＯＭ（電気的消去可能リードオンリメモリ）、ＵＶＰＲＯＭ（紫外線消去可能リードオンリメモリ）、フラッシュメモリ、磁気カード、光カードのような回路、及び／又は電子命令の格納のために適切な如何なるタイプの媒体を含む。

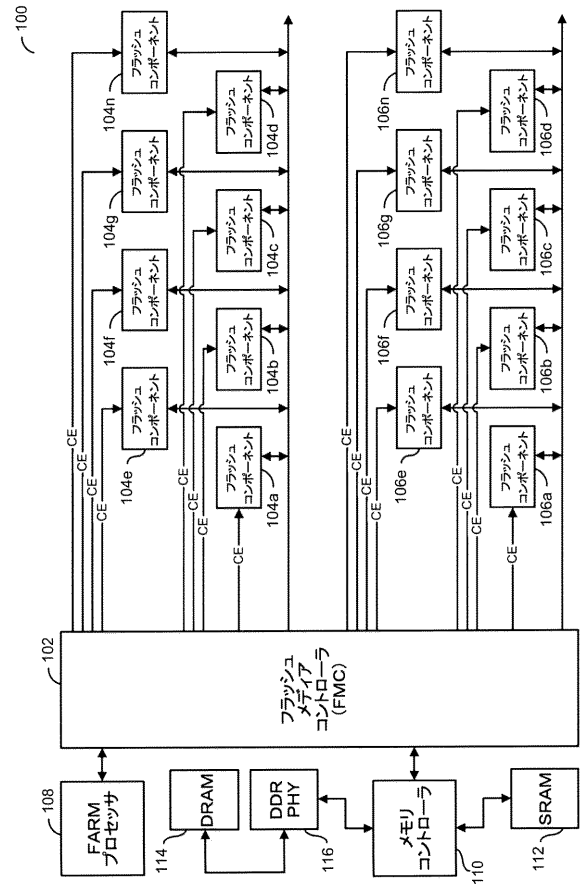
【０１０８】

本発明のエレメントは、１つ以上のデバイス、ユニット、コンポーネント、システム、機械及び／又は装置の一部又は全てを構成する。デバイスは、限定されるものではないが、サーバ、ワークステーション、記憶アレイコントローラ、記憶システム、パーソナルコンピュータ、ラップトップコンピュータ、ノートブックコンピュータ、パームコンピュータ、携帯情報端末、携帯電子デバイス、バッテリ駆動デバイス、セットトップボックス、エンコーダ、デコーダ、トランスコーダ、コンプレッサ、デコンプレッサ、プリプロセッサ、ポストプロセッサ、トランスミッタ、レシーバ、トランシーバ、サイファ回路、携帯電話、デジタルカメラ、ポジショニング及び／又はナビゲーションシステム、医療機器、ヘッドアップ表示装置、ワイヤレスデバイス、オーディオ録音、記憶及び／又は再生装置、ビデオ録画、記憶及び／又は再生装置、ゲームプラットホーム、周辺装置及び／又はマルチチップモジュールを含む。関連する技術分野における当業者は、特定用途の基準を満たす他のタイプのデバイスで、本発明のエレメントが実現されることを理解するであろう。

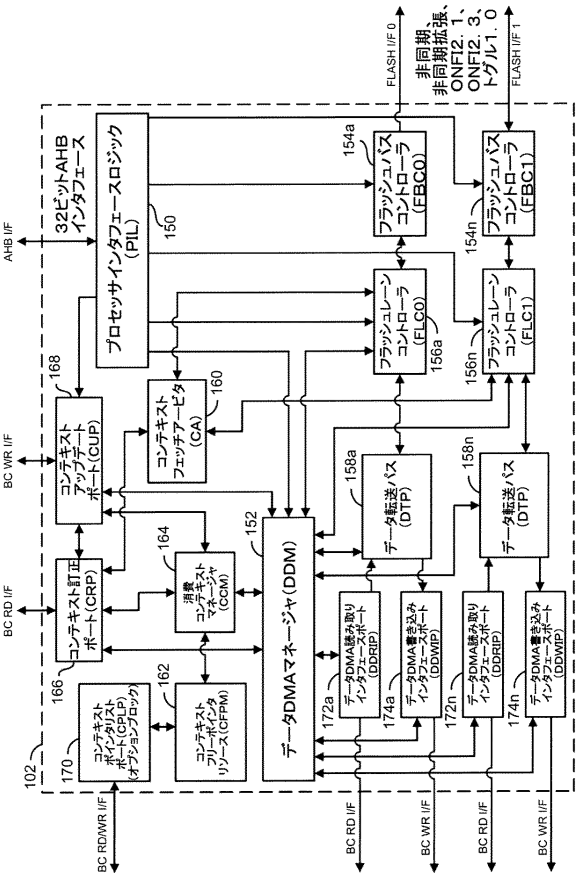
【０１０９】

本発明は、特にその好適な実施の形態に関して表わされ説明されたが、本発明の範囲から逸脱することなく、形式と細部の様々な変更なし得るであろうことが当業者によって理解されるであろう。

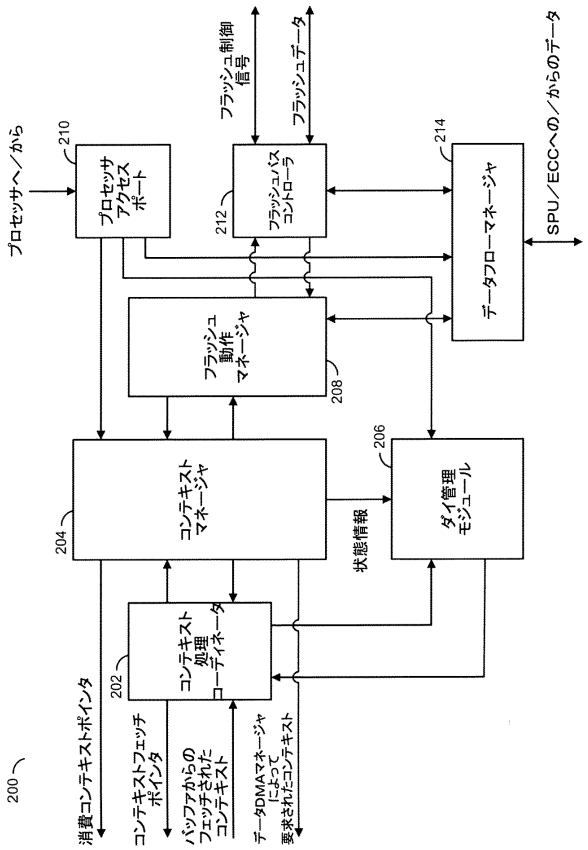
【図１】


100
フラッシュコンポーネント
CE
フラッシュメディアコントローラ(FMC)
102
FARMプロセッサ
108
DRAM
114
DDR PHY
116
メモリコントローラ
110
SRAM
112

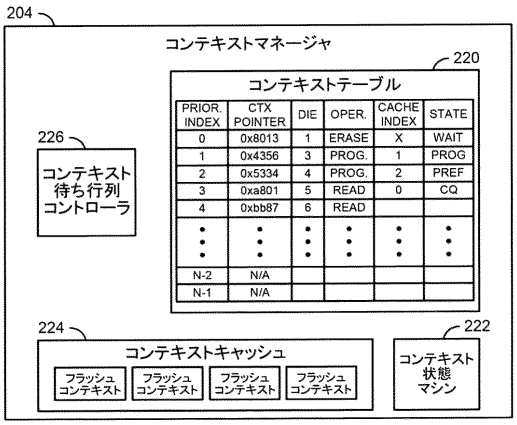
【図２】


FLASH I/F 0
FLASH I/F 1
非同期、非同期拡張、ONFI2.1、ONFI2.3、トグル1.0
AHB I/F
32ビットAHBインタフェース(PIL)
150
プロセッサインタフェースロジック
フラッシュバスコントローラ(FBC0)
154a
フラッシュバスコントローラ(FBC1)
154n
フラッシュレーンコントローラ(FLC0)
156a
フラッシュレーンコントローラ(FLC1)
156n
BC WR I/F
コンテキストアップデートポート(CUP)
168
コンテキストフェッチアービタ(CA)
160
データ転送バス(DTP)
158a
データ転送バス(DTP)
158n
BC RD I/F
コンテキスト訂正ポート(CRP)
消費コンテキストマネージャ(CCM)
164
データDMAマネージャ(DDM)
152
データDMA読み取りインタフェースポート(DDRIP)
172a
データDMA書き込みインタフェースポート(DDWIP)
174a
データDMA読み取りインタフェースポート(DDRIP)
172n
データDMA書き込みインタフェースポート(DDWIP)
174n
コンテキストポインタリスト(GPLP)(オプショナルブロック)
166
コンテキストフリーポインタリソース(CFPM)
162
102
170
BC RD/WR I/F
BC RD I/F
BC WR I/F
BC RD I/F
BC WR I/F

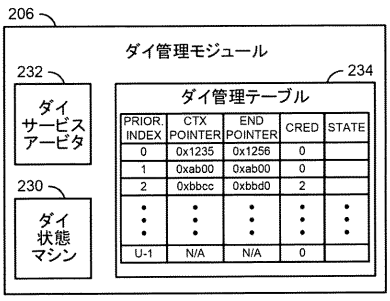
【図３】


プロセッサへ／から
210
プロセッサアクセスポート
フラッシュ制御信号
フラッシュデータ
212
フラッシュバスコントローラ
214
データフローマネージャ
SPU／ECCへの／からのデータ
フラッシュ動作マネージャ
208
コンテキストマネージャ
204
ダイ管理モジュール
206
状態情報
コンテキスト処理コーディネータ
202
200
消費コンテキストポインタ
コンテキストフェッチポインタ
バッファからのフェッチされたコンテキスト
データDMAマネージャによって要求されたコンテキスト

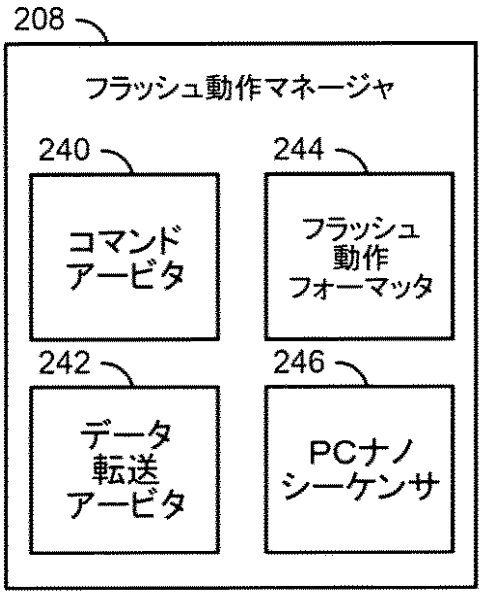
【図４】


204
コンテキストマネージャ
220
コンテキストテーブル
PRIOR. INDEX
CTX POINTER
DIE
OPER.
CACHE INDEX
STATE
0 0x8013 1 ERASE X WAIT
1 0x4356 3 PROG. 1 PROG
2 0x5334 4 PROG. 2 PREF
3 0xa801 5 READ 0 CQ
4 0xbb87 6 READ
N-2 N/A
N-1 N/A
226
コンテキスト待ち行列コントローラ
224
コンテキストキャッシュ
フラッシュコンテキスト
フラッシュコンテキスト
フラッシュコンテキスト
フラッシュコンテキスト
222
コンテキスト状態マシン

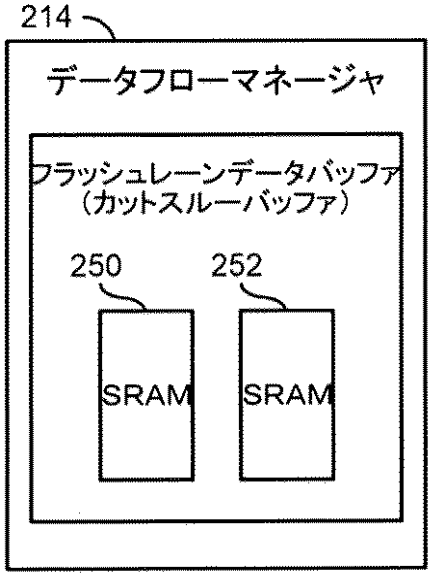
【図５】


206
ダイ管理モジュール
232
ダイサービスアービタ
230
ダイ状態マシン
234
ダイ管理テーブル
PRIOR. INDEX
CTX POINTER
END POINTER
CRED
STATE
0 0x1235 0x1256 0
1 0xab00 0xab00 0
2 0xbbcc 0xbbd0 2
U-1 N/A N/A 0

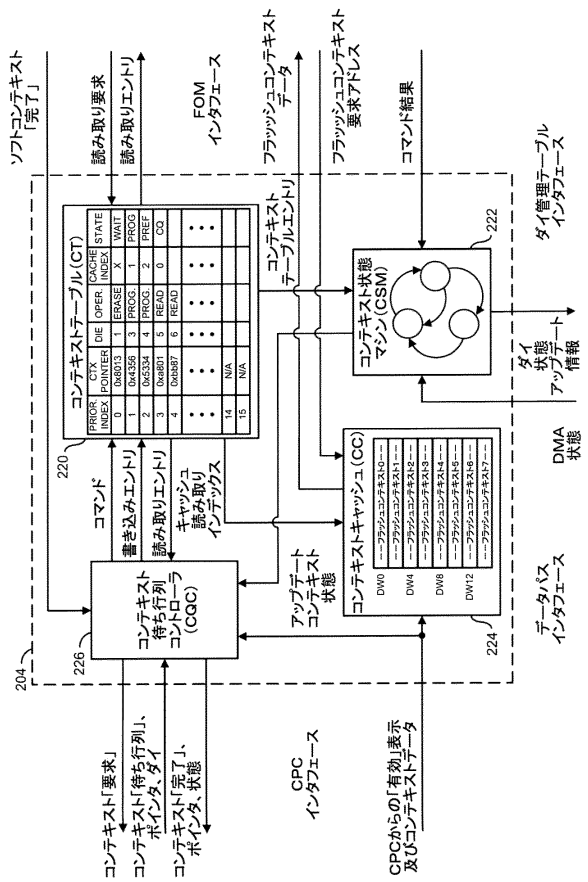
【図６】


208
フラッシュ動作マネージャ
240
コマンド
アービタ
244
フラッシュ
動作
フォーマッタ
242
データ
転送
アービタ
246
PCナノ
シーケンサ

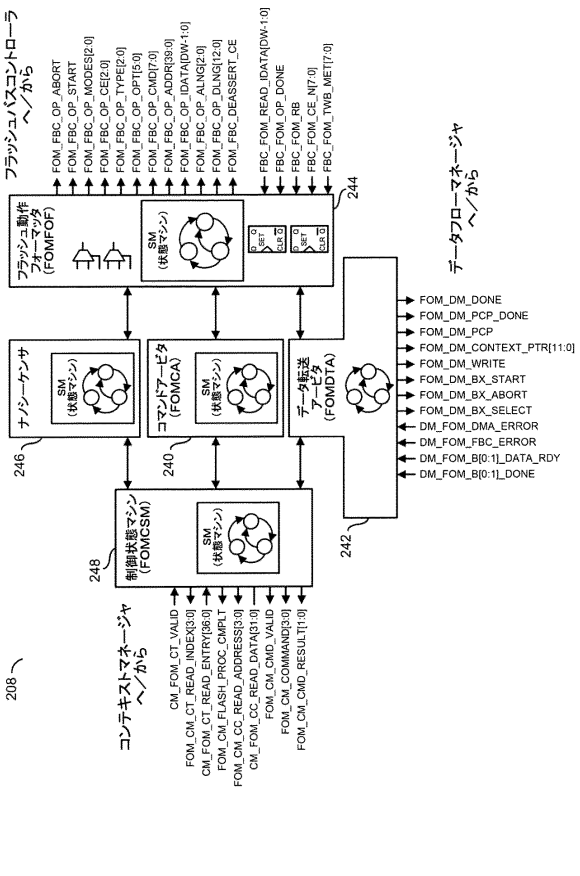
【図７】


214
データフローマネージャ
フラッシュレーンデータバッファ
（カットスルーバッファ）
250
252
SRAM
SRAM

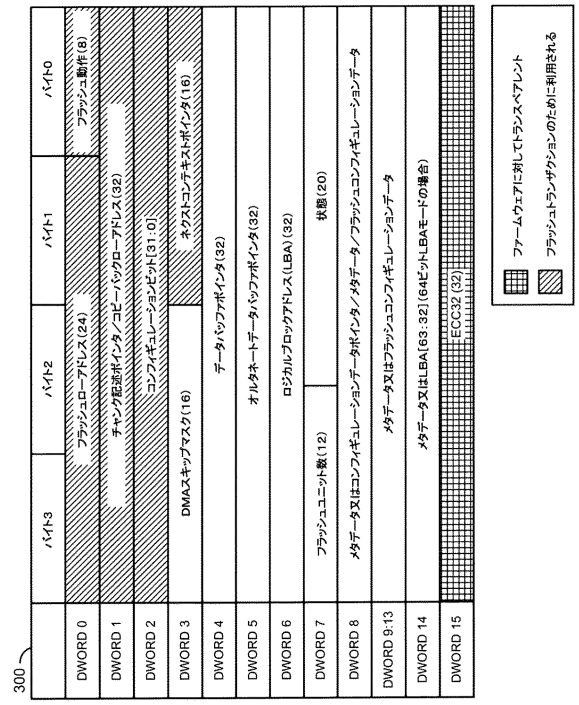
【図８】


ソフトコンテキスト「完了」
読み取り要求
読み取りエントリ
FOM インタフェース
フラッシュコンテキストデータ
フラッシュコンテキスト要求アドレス
コマンド結果
コンテキストテーブル(CT)
コンテキストテーブルエントリ
コンテキスト状態マシン(CSM)
ダイ管理テーブルインタフェース
ダイ状態アップデート情報
コマンド
書き込みエントリ
読み取りエントリ
キャッシュ読み取りインデックス
コンテキストキャッシュ(CC)
アップデートコンテキスト状態
コンテキスト待ち行列コントローラ(CQC)
データバスインタフェース
DMA状態
204
220
222
224
226
コンテキスト「要求」
コンテキスト「待ち行列」、ポインタ、ダイ
コンテキスト「完了」、ポインタ、状態
CPC インタフェース
CPCからの「有効」表示及びコンテキストデータ

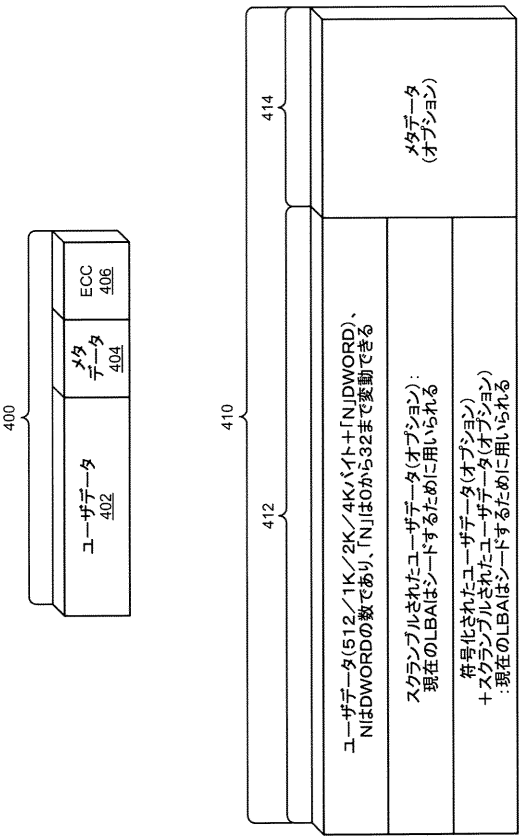
【図９】


フラッシュバスコントローラへ／から
データフローマネージャへ／から
フラッシュ動作フォーマッタ(FOMFOF)
ナノシーケンサ
コマンドアービタ(FOMCA)
データ転送アービタ(FOMDTA)
制御状態マシン(FOMCSM)
SM(状態マシン)
コンテキストマネージャへ／から
208
240
242
244
246
248

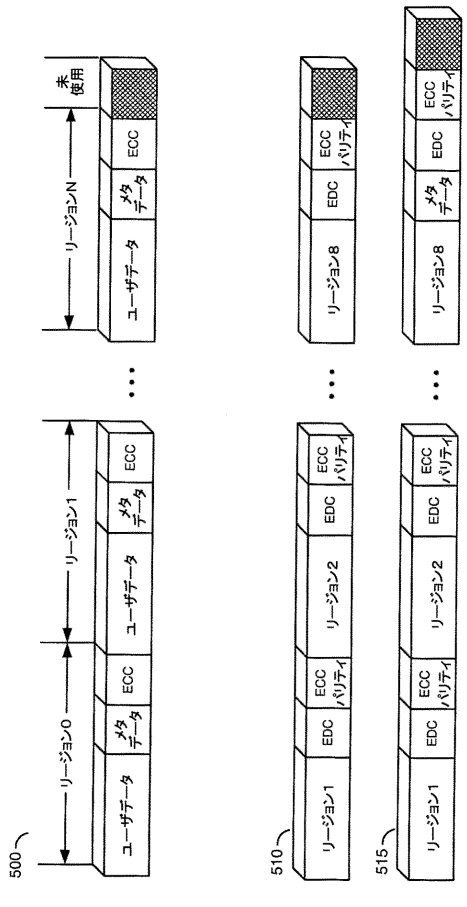
【図１０】



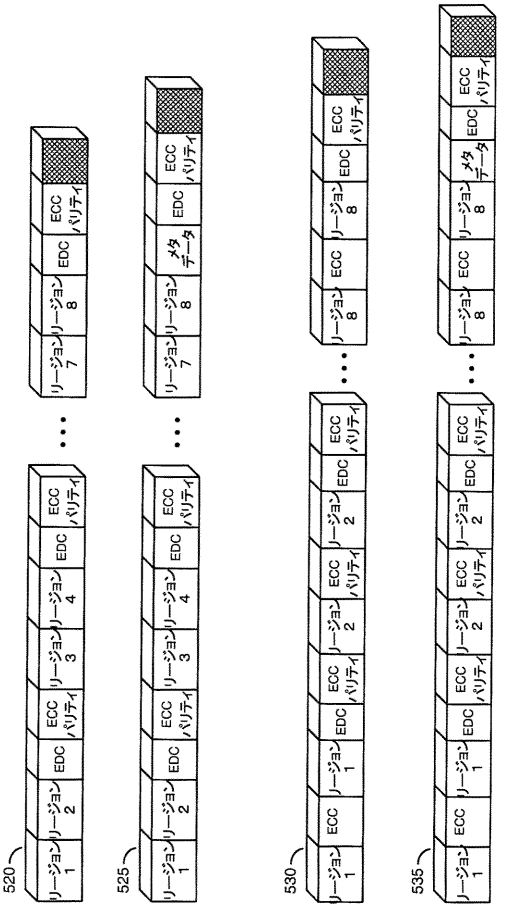
【図１１】



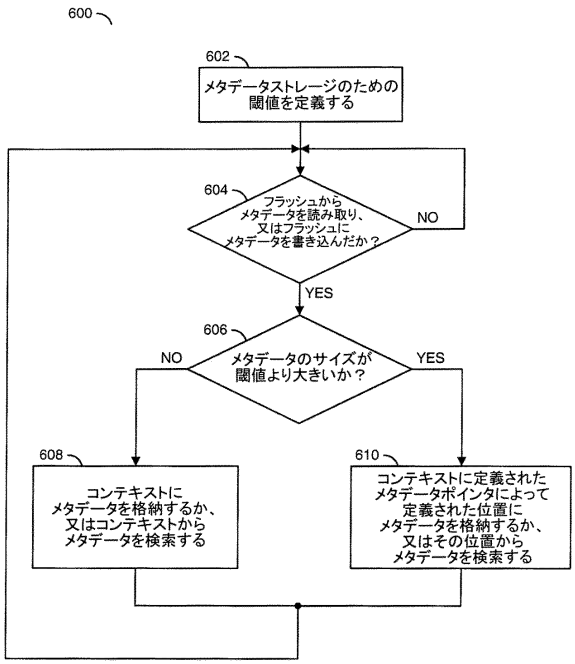
【図１２Ａ】



【図１２Ｂ】



【図１３】


600
602
メタデータストレージのための
閾値を定義する
604
フラッシュから
メタデータを読み取り、
又はフラッシュに
メタデータを書き込んだか？
NO
YES
606
メタデータのサイズが
閾値より大きいか？
NO
YES
608
コンテキストに
メタデータを格納するか、
又はコンテキストから
メタデータを検索する
610
コンテキストに定義された
メタデータポインタによって
定義された位置に
メタデータを格納するか、
又はその位置から
メタデータを検索する

フロントページの続き

(72)発明者　ヴィネイ・アショック・ソマナシュ
インド　４１１０２７　プルネ・マハラシトラ　アウンド・アンエクス　ピンプル・サウダガール　ドゥワルカディーシ・レジデンシー　フラット・ナンバー　ジェイ－７
(72)発明者　マイケル・エス・ヒッケン
アメリカ合衆国　ミネソタ州　ロチェスター　オルムステド　ベイリー・サミット・ドライブ・サウスウェスト　２１１９
(72)発明者　パメラ・エス・ヘンプステッド
アメリカ合衆国　ミネソタ州　オロノコ　オルムステド　ホワイト・バーチ・コート　７７４
(72)発明者　ティモシー・ダブリュー・スワトシュ
アメリカ合衆国　ミネソタ州　ロチェスター　オルムステド　ノース・オークス・コート・ノースイースト　６４０６
(72)発明者　ジャクソン・エル・エリス
アメリカ合衆国　コロラド州　フォート・コリンズ　ポール・ライン・レイン　２２１８
(72)発明者　マーティン・エス・デル
アメリカ合衆国　ペンシルヴァニア州　ベツレヘム　ハービー・ロード　４９６０

審査官　後藤　彰

(56)参考文献　国際公開第２０１０／０３８７３６（WO，Ａ１）
特表２０１２－５０３２３４（ＪＰ，Ａ）
特表２００９－５１２０２２（ＪＰ，Ａ）
国際公開第２０１０／０８４７５４（WO，Ａ１）
米国特許出願公開第２０１０／００２３６８２（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００
Ｇ０６Ｆ　１２／０２