(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4652394号
(P4652394)
(45)発行日　平成23年3月16日(2011.3.16)　(24)登録日　平成22年12月24日(2010.12.24)

(51)Int.Cl.　　F I
G06F 13/28 (2006.01)　　G06F 13/28 310D
G06F 12/02 (2006.01)　　G06F 12/02 580J

請求項の数 5　(全 19 頁)

(21)出願番号　特願2007-501776(P2007-501776)
(86)(22)出願日　平成17年1月21日(2005.1.21)
(65)公表番号　特表2007-527071(P2007-527071A)
(43)公表日　平成19年9月20日(2007.9.20)
(86)国際出願番号　PCT/US2005/001776
(87)国際公開番号　WO2005/096161
(87)国際公開日　平成17年10月13日(2005.10.13)
審査請求日　平成20年1月21日(2008.1.21)
(31)優先権主張番号　10/792,591
(32)優先日　平成16年3月3日(2004.3.3)
(33)優先権主張国　米国(US)

(73)特許権者　504199127
フリースケール　セミコンダクター　インコーポレイテッド
アメリカ合衆国　７８７３５　テキサス州　オースティン　ウィリアム　キャノン　ドライブ　ウエスト　６５０１
(74)代理人　100116322
弁理士　桑垣　衛
(72)発明者　モイヤー、ウィリアム　シー.
アメリカ合衆国　７８６２０　テキサス州　ドリッピング　スプリングス　ピア　ブランチ　ロード　１００５

審査官　坂東　博司

最終頁に続く

(54)【発明の名称】マルチバーストプロトコルデバイスコントローラ

(57)【特許請求の範囲】
【請求項１】
方法であって、
バーストリクエストをマスターから受信すること、
リクエストされたバーストに対応するバースト特性を求めること、
複数のバーストエラープロトコルの一つを前記バースト特性に基づいて選択することであって、前記バースト特性を求めることは、前記リクエストされたバーストに対応するバーストタイプを決定することを含み、前記選択されたバーストエラープロトコルは、前記バーストタイプに基づいて選択される、複数のバーストエラープロトコルの一つを前記バースト特性に基づいて選択すること、
前記選択されたバーストエラープロトコルに従って前記バーストリクエストに応答することであって、
バースト転送を開始すること、
前記バースト転送の間にエラーを検出すること、
前記選択されたバーストエラープロトコルに基づき、前記バースト転送の間に検出されたエラーに応じて前記バースト転送を選択的に終了させること、
を含む、前記選択されたバーストエラープロトコルに従って前記バーストリクエストに応答すること、
を備える、方法。
【請求項２】

メモリコントローラであって、
リクエストされたバースト転送に対応する少なくとも一つのバースト特性を求め、前記少なくとも一つのバースト特性に基づいてバーストエラープロトコルを選択するバーストプロトコル選択ロジックであって、前記少なくとも一つのバースト特性は、前記リクエストされたバースト転送のバーストタイプを含む、バーストプロトコル選択ロジックと、
前記バーストプロトコル選択ロジックに接続され、前記選択されたバーストエラープロトコルに従って、リクエストされたバーストに応答して前記リクエストされたバースト転送を開始する制御ロジックであって、
前記リクエストされたバースト転送の間にエラーを検出するエラー検出ロジックと、
前記リクエストされたバースト転送を、前記バーストプロトコル選択ロジックから提供された前記選択されたバーストエラープロトコルに基づき、前記バースト転送の間に検出されたエラーに応じて選択的に終了させる終了制御ロジックと、を含む制御ロジックと、
前記バーストプロトコル選択ロジックに接続された制御レジスタであって、前記バーストプロトコル選択ロジックは、前記少なくとも一つのバースト特性及び前記制御レジスタに保存された情報に基づいて前記バーストエラープロトコルを選択し、前記制御レジスタに保存された情報は、複数のバーストエラープロトコルを示し、前記バーストプロトコル選択ロジックは、前記バーストエラープロトコルを前記複数のバーストエラープロトコルから選択し、前記複数のバーストエラープロトコルの各々は、あるバーストタイプに対応し、該バーストタイプは、リードバーストタイプ、ライトバーストタイプ、制限されたバーストタイプ、及び制限の無いバーストタイプのうちの少なくとも一つを含む、制御レジスタと、
を備える、メモリコントローラ。
【請求項３】
メモリコントローラであって、
バーストリクエストをマスターから受信する手段と、
リクエストされたバーストに対応するバースト特性を求める手段と、
複数のバーストエラープロトコルの一つを前記バースト特性に基づいて選択する手段と、
前記選択されたバーストエラープロトコルに従って前記バーストリクエストに応答する手段であって、
バースト転送を開始する手段と、
前記バースト転送の間にエラーを検出する手段と、
前記選択されたバーストエラープロトコルに基づき、前記バースト転送の間に検出されたエラーに応じて前記バースト転送を選択的に終了させる手段と、
を含む、前記選択されたバーストエラープロトコルに従って前記バーストリクエストに応答する手段と、
を備える、メモリコントローラ。
【請求項４】
複数のマスターと通信可能に構成されたメモリコントローラであって、
リクエストされたバースト転送に対応する前記複数のマスターの一つの識別情報を求め、前記複数のマスターの一つの識別情報に基づいてバーストエラープロトコルを選択するバーストプロトコル選択ロジックと、
前記バーストプロトコル選択ロジックに接続された制御レジスタであって、前記バーストプロトコル選択ロジックは、前記複数のマスターの一つの識別情報及び前記制御レジスタに保存された情報に基づいて、前記バーストエラープロトコルを選択する、制御レジスタと、
前記バーストプロトコル選択ロジックに接続され、前記選択されたバーストエラープロトコルに従って、リクエストされたバーストに応答する制御ロジックと、
を備え、前記制御ロジックが、

前記リクエストされたバースト転送を開始し、
前記バースト転送の間にエラーを検出し、
前記選択されたバーストエラープロトコルに基づき、前記バースト転送の間に検出されたエラーに応じて前記バースト転送を選択的に終了させる、
ことを特徴とするメモリコントローラ。

【請求項５】
メモリコントローラであって、
リクエストされたバースト転送に対応する少なくとも一つのバースト特性を求め、前記少なくとも一つのバースト特性に基づいて、複数のバーストエラープロトコルから一つのバーストエラープロトコルを選択するバーストプロトコル選択ロジックであって、当該メモリコントローラは、複数のマスターと通信可能に構成され、前記複数のバーストエラープロトコルの各々は、前記複数のマスターのうちの一つに対応する、バーストプロトコル選択ロジックと、
前記バーストプロトコル選択ロジックに接続され、前記選択されたバーストエラープロトコルに従って、リクエストされたバーストに応答する制御ロジックと、
を備え、前記制御ロジックが、
前記リクエストされたバースト転送を開始し、
前記バースト転送の間にエラーを検出し、
前記選択されたバーストエラープロトコルに基づき、前記バースト転送の間に検出されたエラーに応じて前記バースト転送を選択的に終了させる、
ことを特徴とするメモリコントローラ。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明はシステムに関し、特にシステム内のバス及びデバイスアクセスプロトコルに関する。

【背景技術】
【０００２】
マルチマスター（ｍｕｌｔｉ－ｍａｓｔｅｒ）システムオンチップ（ｓｙｓｔｅｍｓ　ｏｎ　ｃｈｉｐ：ＳｏＣ）のようなシステムでは、ＳＲＡＭまたはフラッシュメモリコントローラのようなスレーブデバイスは、各潜在マスターのインターフェースプロトコルに対する予測を正しく、かつ効率的にサポートする必要がある。従来のプラットフォームベース設計では、異なるマスターはエラーを含むバースト転送がどのように終了するかについて異なる予測を行い得る。例えば、ダイレクトメモリアクセス（ＤＭＡ）マスター及び所定の中央処理ユニット（ＣＰＵ）では、エラーが一旦バースト転送のいずれかのビートで通知されると、バースト転送の中断を予測する。また、他のＣＰＵ設計では特定ビートのエラー通知が行なわれるのにも拘らず、バーストが継続すると予測する。

【発明の開示】
【発明が解決しようとする課題】
【０００３】
上記のような異なる予測はメモリコントローラにとって問題となる。従来の方法はＭＡＣ層のレベル（ハードウェアレベル）よりも高いレベルの複数のプロトコルを使用する。このような方法では、バス転送または複数のマスタープロトコルをサポートするという課題を解決することができない。従って、メモリコントローラの設計及び実装方法を改善することが望ましい。

【発明を実施するための最良の形態】
【０００４】
本発明は、添付の図を参照することにより一層深く理解することができ、この技術分野の当業者であれば、本発明の多くの目的、特徴、及び利点を理解し得る。異なる図における同じ参照記号の使用は、同様の、または同じ構成要素であることを意味する。

【０００５】
以下の説明は、本発明の少なくとも一つの例に関する詳細な説明により、この技術分野の当業者が本発明を理解し、容易に実施し得るようにするために為される。以下の説明は例示であるので、本発明自体を制限するものであると捉えられてはならない。そうではなく、いかなる変更も、特許請求の範囲に厳密に規定される本発明の技術範囲に含まれる。
【０００６】
マルチプロトコルはステートマシン設計にハードウェアレベルで取り入れることができ、適切なプロトコルはバーストタイプや、現在のバスマスターの識別情報などに基づいて選択することができることが分かっている。これにより、例えばスレーブはマルチマスターシステムオンチップ（ＳｏＣ）の複数のエラープロトコルをサポートし、かつバースト転送を特定の方法で、転送を開始する特定のマスターに基づいて選択的に終了する機能を備えることができる。プロトコルの選択は、一連の入力コンフィグレーションピンを使用するか、またはスレーブまたはスレーブインターフェースブロック（例えば、メモリコントローラ）に設けられたプログラマブルレジスタを使用して行われ、制御される。一実施形態では、マスターごとのバーストエラープロトコルをサポートするマルチマスターシステムオンチップ（ＳｏＣ）メモリコントローラについて説明する。マスターは、マスターが供給する、またはマスター外部のロジックが供給する種々の信号によって、スレーブに対して特定することができる。各潜在マスターは通常、制御フィールドをプログラマブルレジスタに有し、このフィールドは使用する適切なプロトコルを示す。一つの実施形態では、複数のプロトコルを個々にリードバースト及びライトバーストに関してプログラムすることができる。その理由は、ハンドシェークプロトコルが異なり得るからである。このようにして、バスプロトコルを現在のバスマスターに基づいてプログラム可能な制御レジスタ手段によって効率的に選択する機構を設けてバーストエラー終了のようなイベントを処理する。
【０００７】
図１は情報処理システム１０の一つの実施形態を示している。システム１０は接続マスター１２、接続マスター１４、Ｉ／Ｏ回路１６、周辺装置１８、及び他のスレーブ２０のような種々のシステム要素を含む。メモリコントローラ３６及びメモリ３５はシステム１０内部の別のスレーブ素子を表わす。メモリ３５は共有ストレージユニットとなる。その理由は、このメモリ３５は２以上の接続マスターによって共有されるからである。メモリコントローラ３６はバーストプロトコル選択ロジック５０、制御レジスタ３８、及び制御ロジック４０を含む。制御ロジック４０は終了制御ロジック４２及びエラー検出ロジック４４を含む。
【０００８】
システム相互接続２２はマスター１２、マスター１４、Ｉ／Ｏ回路１６、周辺装置１８、他のスレーブ２０、及びメモリコントローラ３６を相互接続する。マスター１２はシステム相互接続２２に導体４８を介して双方向接続され、マスター１４はシステム相互接続２２に導体５０を介して双方向接続され、Ｉ／Ｏ回路１６はシステム相互接続２２に導体５２を介して双方向接続され、周辺装置１８はシステム相互接続２２に導体５４を介して双方向接続され、他のスレーブ２０はシステム相互接続２２に導体５６を介して双方向接続され、メモリコントローラ３６はシステム相互接続２２に導体２４を介して双方向接続される。メモリ３５はメモリコントローラ３６に導体３３を介して双方向接続される。
【０００９】
導体２４はメモリ及び／又はバス接続情報をメモリコントローラ３６に送信する種々の導体を含む。図示の実施形態では、このような情報はマスター識別子２６、アドレス／データ２７、Ｒ／Ｗ信号２８、バーストタイプ信号３０、命令／データ信号３２、エラー信号３７、及び他の信号３４を含む。このような情報の幾つか、または全ては、マスター１２またはマスター１４、或いは他のデバイスによるメモリ３５へのアクセス（例えば、バーストアクセス）の各インスタンス（ｉｎｓｔａｎｃｅ）で、メモリコントローラ３６に対して送信することができる。特定の物理導体はメモリアクセス情報の各タイプに対して

設ける必要はない。このような情報はシリアルにまたは別の形式で送信することができる。

【００１０】

動作状態では、マスター１２及び１４はシステム相互接続２２へのアクセスをリクエストして、他のスレーブ２０へのアクセス、周辺装置１８へのアクセス、またはメモリコントローラ３６を介したメモリ３５へのアクセスを要求する。リクエスト側のマスターはアクセスリクエストをシステム相互接続２２を介してメモリコントローラ３６に供給することができる。アクセスリクエストは、例えばデータまたは命令のいずれかに関するリードリクエストまたはライトリクエストとすることができる。このアクセスはバーストアクセスとすることができ、バーストアクセスでは、マルチビート（ｍｕｌｔｉ－ｂｅａｔ）のリードまたはライトによって一連のデータブロックが転送される。別の構成として、ノンバーストタイプ（ｎｏｎ－ｂｕｒｓｔ　ｔｙｐｅ）のアクセスはシングルビート（ｓｉｎｇｌｅ　ｂｅａｔ）のリードまたはライトを含むことができる。メモリコントローラ３６はリードアクセスリクエストに応答して、リクエストされた情報（データまたは命令）をリクエスト側のマスターにシステム相互接続２２を介して返送する。

【００１１】

マスター識別子２６がメモリコントローラ３６に供給され、メモリコントローラ３６はどのマスターが現在のアクセスをリクエストしているかについて識別する。メモリコントローラ３６はどのマスターがバーストアクセスをリクエストしたかについて、マスター識別子２６を処理することにより判断する。例えば、マスター１２は識別子「０」を有し、そしてマスター１４は識別子「１」を有することができる。従って、各マスターには固有の識別子を割り当てることができる。別の実施形態では、幾つかのマスターが同じ識別子を共有することができる。従って、識別子によって、固有に識別することができるマスターとは異なるマスタークラスを識別することができる。また、別の実施形態では、アクセスをリクエストするマスターの識別情報は、マスター識別子２６のような信号を供給する方法以外の方法により求めることができる。

【００１２】

現在のアクセスリクエストがリードタイプのアクセスであるか、またはライトタイプのアクセスであるかの通知がメモリコントローラ３６に（及び、例えば以下に説明する制御ロジック４０に）Ｒ／Ｗ信号２８によって送信される。

【００１３】

バーストタイプはメモリコントローラ３６にバーストタイプ信号３０によって供給される。バーストタイプ信号は、現在のアクセスがバーストタイプのアクセスであるか、またはノンバーストタイプのアクセスであるかについて示す。アクセスがバーストリクエストである場合、バースト信号３０はバーストリクエストのタイプも示す。例えば、バーストタイプ信号３０はバーストアクセスが、バースト転送長さが制限されたアクセス（ｂｏｕｎｄｅｄ　ａｃｃｅｓｓ）であるか、またはバースト転送長さに制限の無いアクセス（ｕｎｂｏｕｎｄｅｄ　ａｃｃｅｓｓ）であるかについて示すことができる。バーストタイプ信号３０は更に、または別の構成として、バーストアクセスがインクリメントバーストタイプ（ｉｎｃｒｅｍｅｎｔｉｎｇ　ｂｕｒｓｔ　ｔｙｐｅ）であるか、またはラップバーストタイプ（ｗｒａｐｐｉｎｇ　ｂｕｒｓｔ　ｔｙｐｅ）であるかについて示すこともできる。データのバースト転送は制限することもできるし、所定の長さとすることもできるし、または制限せず、ビート数をバースト内で動的に決定することもできる。制限の無いバーストは未定義長バースト（ｕｎｄｅｆｉｎｅｄ－ｌｅｎｇｔｈ　ｂｕｒｓｔｓ）と呼ぶこともできる。制限されたバーストは固定長バースト（ｆｉｘｅｄ－ｌｅｎｇｔｈ　ｂｕｒｓｔｓ）と呼ぶこともできる。一つの実施形態では、一つのバースト転送内での４ビート，８ビート，または１６ビートの制限された転送は、情報処理システム１０によってサポートされる。他の実施形態では、他の所定長を有し、かつ制限されたバーストをサポートすることができる。

【００１４】

メモリコントローラ３６は、現在のアクセスリクエストに対応するアドレス情報も受信し、リクエストされた情報をアドレス／データ２７を介して供給する。命令／データ信号３２は、現在のアクセスリクエストが命令またはデータのいずれに関するものであるかを示すために、メモリコントローラ３６に供給される。メモリコントローラ３６に送信及びメモリコントローラ３６から受信するために必要な他のいずれの信号も、他の信号３４に含めて供給することができる。

【００１５】

メモリコントローラ３６の内部では、バーストプロトコル選択ロジック５０は、バースト制御情報をバースト制御情報ソースから受信するように接続される。図示の実施形態では、バースト制御情報は、制御レジスタ３８及びコンフィグレーションピン３９の少なくとも一つから供給される。バーストプロトコル選択ロジック５０は、制御ロジック４０に、バースト制御情報に応じたバーストプロトコル選択情報を供給するように接続される。制御ロジックは、メモリ３５へのバーストリクエストの処理を、バーストプロトコル選択情報を使用して行なう。例えば、図示の実施形態では、終了制御ロジック４２はメモリ３５へのバーストリクエストの終了を、エラー検出ロジック４４が検出するエラーに対応するバーストプロトコル選択情報に従って制御する。

【００１６】

異なるタイプのバーストプロトコルは異なるタイプのバースト特性によって起動することができる。バースト特性は、例えば、どのマスターがアクセスをリクエストしているか、アクセスリクエストがバーストアクセスの一部であるかどうか、バーストリクエストが制限されたバースト、制限の無いバースト、インクリメントバーストタイプ、及びラップバーストタイプのいずれであるか、を含む。従って、現在のアクセスリクエスト（すなわち、現在のバーストリクエスト）に対応するマスター識別子２６、Ｒ／Ｗ信号３８、及びバーストタイプ信号３０の値に基づいて、かつ制御レジスタ３８（及び／又はコンフィグレーションピン３９）に基づいて、バーストプロトコル選択ロジック５０は、現在のアクセスリクエストによって実行されるバースト動作、及びこの動作に関して生じるエラー状態を判断する。

【００１７】

図２は制御レジスタ３８の一つの実施形態を示し、この実施形態の制御レジスタ３８は、各マスターに対応するバーストリードエラー（ＢＲＥ）終了制御フィールド及びバーストライトエラー（ＢＷＥ）終了制御フィールドのような、バーストリクエスト制御情報を格納するフィールドを含む。図２に示すように、制御レジスタ３８はマスター１２のＢＲＥ終了フィールド６０、マスター１４のＢＲＥ終了フィールド６２、マスター１２のＢＷＥ終了フィールド６４、及びマスター１４のＢＷＥ終了フィールド６６を含む。従って、別の実施形態では、制御レジスタ３８は必要に応じて、上記よりも多い、または少ないフィールドを含んで所望のバースト制御情報を格納することができる。尚、制御レジスタ３８は、システム相互接続２２に接続されたマスター１２または１４のようなマスターからの命令を介してプログラムすることができる。値は、例えばユーザが供給することもできるし、または設計時にプログラムすることもできる。

【００１８】

図３は、図２の制御レジスタ３８のフィールド定義の一つの実施形態を示している。例えば、一つの実施形態では、フィールド６０，６２，６４，及び６６の各々は２ビットフィールドであり、この場合、各フィールドは４つの値（００，０１，１０，及び１１）を有することができる。図３に示すように、フィールド６０，６２，６４，及び６６は、バーストリードエラーまたはバーストライトエラーが生じるときにバーストリクエストの終了に使用するプロトコルに対するマスター固有の制御を行なうための情報を含む。

【００１９】

図３の例では、ＢＲＥフィールド６０及び６２の各々は２ビットフィールドであり、値００は、バーストリードエラーが生じるときにアクションを起こす必要がないことを示す。例えば、マスター１２のＢＲＥフィールド６０が００に設定される場合、マスター１２

からのバーストリードリクエストの処理はエラーの影響を受けずに継続され、したがって、リードバーストの残りのビートは完了することができる。同様に、マスター１４のＢＲＥフィールド６２が００に設定される場合、バースト終了のようなエラー処理は、マスター１４からのバーストリードに対応するバーストリードエラーによって開始されることはない。

【００２０】

ＢＲＥフィールド６０及び６２の値０１は、データ転送中にバーストリードエラーが生じたときにリードバーストを中断する必要があることを示す。例えば、マスター１２のＢＲＥフィールド６０が０１に設定される場合、マスター１２からのバーストリードリクエストの処理はエラーによって終了し、したがって、リードバーストの残りのビートは完了することができない。同様に、マスター１４のＢＲＥフィールド６２が０１に設定され、かつエラーが生じる場合、バースト終了エラー処理は、マスター１４からのバーストリードに対応するバーストリードエラーによって開始される。

【００２１】

ＢＲＥフィールド６０及び６２の値１０は、転送の第１ビート（クリティカルワード（ｃｒｉｔｉｃａｌ　ｗｏｒｄ））の間にバーストリードエラーが生じたときにリードバーストを中断する必要があることを示す。例えば、マスター１２のＢＲＥフィールド６０が１０に設定される場合、マスター１２からのバーストリードリクエストの処理は、エラーがマスター１２への転送の第１ビートの間に生じる場合に当該エラーによって終了し、したがって、リードバーストの残りのビートは完了することができない。第１ビートに続くバースト内の他のエラーによってリードバーストが終了することはない。マスター１２への転送の第１ビートの後に同じエラーが生じる場合、リードバーストの残りの全てのビートは完了することができる。同様に、例えば、マスター１４のＢＲＥフィールド６２が１０に設定され、かつ、マスター１４が関係するバーストリード転送の間において第１ビート（クリティカルワード）にエラーが生じる場合、バースト終了エラー処理が開始される。マスター１４のＢＲＥフィールド６２が１０に設定され、かつエラーがクリティカルワードの後に生じる場合、バースト終了エラー処理は開始されない。

【００２２】

ＢＲＥフィールド６０及び６２の値１１は、未定義長のリードバーストの間に生じるエラーによってリードバーストが終了するが、定義長のリードバーストの間に生じるエラーによってリードバーストが終了することがないことを示す。例えば、マスター１２のＢＲＥフィールド６０が１１に設定される場合、マスター１２からのバーストリードリクエストの処理は、バーストタイプ（例えば、バーストタイプ信号３０によって示される）が未定義長のバーストリクエストである場合にエラーによって終了する。このような場合においては、リードバーストの残りのビートは完了することができない。同様に、マスター１４のＢＲＥフィールド６２が１１に設定され、かつエラーが生じる場合、バースト終了エラー処理は、バーストタイプがマスター１４からの未定義長のリードバーストを示す場合に開始される。

【００２３】

従って、マスター１２または１４のいずれかからのバーストアクセスリクエストによって開始された、メモリ３５に対するバースト動作をＢＲＥフィールド６０及び６２に基づいて決定及び制御することができる。フィールド６０は、マスター１４が必要とする、または要求するものとは異なるバースト処理プロトコルをマスター１２が必要とする、または要求する場合に、フィールド６２とは異なる値を保持することができる。

【００２４】

ＢＷＥフィールド６４及び６６の各々は２ビットフィールドであり、値００は、バーストライトエラーが生じるときにアクションを起こす必要がないことを示す。例えば、マスター１２のＢＷＥフィールド６４が００に設定される場合、マスター１２によるライトバーストリクエストの処理は、エラーの影響を受けずに継続され、したがって、ライトバーストの残りのビートは完了することができる。同様に、マスター１４のＢＷＥフィールド

６６が００に設定される場合、バースト終了のようなエラー処理は、マスター１４からのバーストライトに対応するバーストライトエラーによって開始されることはない。

【００２５】

ＢＷＥフィールド６４及び６６の値０１は、データ転送中にバーストライトエラーが生じたときにライトバーストを中断する必要があることを示す。例えば、マスター１２のＢＷＥフィールド６４が０１に設定される場合、マスター１２によるライトバーストリクエストの処理はエラーによって終了し、したがって、ライトバーストの残りのビートは完了することができない。同様に、マスター１４のＢＷＥフィールド６６が０１に設定され、かつエラーが生じる場合、バースト終了エラー処理は、マスター１４からのバーストライトに対応するバーストライトエラーによって開始される。

【００２６】

ＢＷＥフィールド６４及び６６の値１０は、転送の第１ビート（クリティカルワード）の間にバーストライトエラーが生じたときにライトバーストを中断する必要があることを示す。例えば、マスター１２のＢＷＥフィールド６４が１０に設定される場合、マスター１２によるライトバーストクエストの処理は、エラーがマスター１２への転送の第１ビートの間に生じる場合に当該エラーによって終了し、したがって、ライトバーストの残りのビートは完了することができない。他のエラーによってライトバーストが終了することはない。マスター１２への転送の第１ビートの後に同じエラーが生じる場合、ライトバーストの残りの全てのビートは完了することができる。同様に、例えば、マスター１４のＢＷＥフィールド６６が１０に設定され、かつ、マスター１４が関係するクリティカルワード転送の間にエラーが生じる場合、バースト終了エラー処理が開始される。マスター１４のＢＷＥフィールド６６が１０に設定され、かつエラーが第１ビート（クリティカルワード）の後に生じる場合、バースト終了エラー処理は開始されない。

【００２７】

ＢＷＥフィールド６４及び６６の値１１は、未定義長のライトバーストの間に生じるエラーによってライトバーストが終了するが、定義長のライトバーストの間に生じるエラーによってライトバーストが終了することがないことを示す。例えば、マスター１２のＢＷＥフィールド６４が１１に設定される場合、マスター１２からのバーストライトリクエストの処理は、バーストタイプ（例えば、バーストタイプ信号３０によって示される）が未定義長のバーストリクエストである場合にエラーによって終了する。このような場合においては、ライトバーストの残りのビートは完了することができない。同様に、マスター１４のＢＷＥフィールド６６が１１に設定され、かつエラーが生じる場合、バースト終了エラー処理は、バーストタイプがマスター１４からの未定義長のライトバーストを示す場合に開始される。

【００２８】

従って、マスター１２または１４のいずれかによって開始された、メモリ３５に対するライトデータバースト動作をＢＷＥフィールド６４及び６６、バーストタイプ信号３０、及びマスター識別信号２６に基づいて決定及び制御することができる。フィールド６４は、マスター１４が必要とする、または要求するものとは異なるバースト処理プロトコルをマスター１２が必要とする、または要求する場合に、フィールド６６とは異なる値を保持することができる。フィールド６０は、マスター１２が、リードバーストに関してライトバーストに関するものとは異なるバースト処理プロトコルを必要とする、または要求する場合にフィールド６４とは異なる値を保持することができる。フィールド６２は、マスター１４が、リードバーストに関してライトバーストに関するものとは異なるバースト処理プロトコルを必要とする、または要求する場合にフィールド６６とは異なる値を保持することができる。

【００２９】

図４は、本発明の一つの実施形態によるシステム１０の動作をフロー図として示している。フローは、バーストリクエストを受信する動作７２から始まり、この動作では、マスターからのバーストリクエストを受信する。このバーストリクエストは、リードリクエス

ト、ライトリクエスト、データリクエスト、命令リクエストなどのような多くの異なるタイプのリクエストとすることができる。

【００３０】

バーストリクエストを受信した後、受信バーストリクエストのバースト特性を、バースト特性を求める動作７４の間に求める。例えば、リクエスト側のマスターの識別情報を求める。この動作は、受信バーストリクエストに関連するマスター識別子２６を処理することにより行なわれる。更に別の例では、バーストリクエストのタイプを、マスター識別情報を求める動作の他に、または代わりに求める。この動作は、受信バーストリクエストに関連するバーストタイプ信号３０を処理することにより行なわれる。

【００３１】

バースト特性を求めた後、種々のバーストプロトコルの一つを、バーストプロトコルを選択する動作７８の間に選択する。例示としてのバーストプロトコルは図３に示される。これらのプロトコルは、決定されたバースト特性によって変わる。図３の場合、バースト特性はリクエスト側のマスターの識別情報に対応する。マスターが異なることによって制御レジスタ３８の設定が異なり、そしてこのような設定の結果さえも異なり得る。バーストプロトコルによる最終的な効果を決定する他の要素として、バーストの性質（例えば、バーストが制限されているかどうか）が含まれる。異なる効果（この場合は終了プロセス）を図３に示すように、異なるタイプのバーストアクセスに関して設定することができる。バスプロトコル情報の選択に関する設定は制御レジスタ３８、コンフィグレーションピン３９、または他の或る手段によって行なわれる。尚、別の実施形態では、複数のマスターは共通セットのプロトコル制御指定子を共有することができる。

【００３２】

バーストプロトコルをプロトコルを選択する動作７８の間に選択した後、メモリコントローラ３６は、選択されたバーストプロトコルに従って、バーストリクエストに応答する。このような応答の例は、図３を参照して上述したとおりである。

【００３３】

一つの実施形態では、バーストプロトコルの選択は回路設計の間に行われる。例えば、図５を参照すると、コントローラを表わすハードウェア記述言語（ｈａｒｄｗａｒｅ ｄｅｓｃｒｉｐｔｉｏｎ ｌａｎｇｕａｇｅ：ＨＤＬ）を処理して種々のバーストプロトコルの一つ以上を選択及び実行することができる。バーストをパラメータ化することができるＨＤＬを導入する動作８２では、異なるバーストプロトコルによってプログラムすることができる、コントローラを表わすＨＤＬコードが導入される。一旦、バーストをパラメータ化することができるＨＤＬが導入されると、バーストパラメータを導入する動作８４の間に、バーストプロトコルがＨＤＬに入力される。そして、バーストがパラメータ化されたＨＤＬがバーストプロトコルパラメータを評価する動作８６で評価される。

【００３４】

評価する動作８６の結果に応じて、第１の値が得られ、第１ＨＤＬを生成する動作８８に制御が移行するようになるか、または第２の値が得られ、第２ＨＤＬを生成する動作９０に制御が移行するようになる。第１ＨＤＬを生成する動作８８の間に、第１バーストプロトコルに従って動作するように構成されたコントローラに対応する第１ＨＤＬ記述が生成される。第２ＨＤＬを生成する動作９０の間に、第２バーストプロトコルに従って動作するように構成されたコントローラに対応する第２ＨＤＬ記述が生成される。他の結果によっては、他のタイプのバーストプロトコルのＨＤＬ生成が実行される。最終のバーストプロトコルが最終のＨＤＬを実行するために選択されるまで、繰り返しプロセスを使用して種々のバーストプロトコルが（ＨＤＬで）評価及び実行される。

【００３５】

一旦、最終のＨＤＬ記述が生成されると（例えばＨＤＬを生成する動作８８または９０のいずれかの間に）、コントローラは、動作９２の間に、最終的に生成されたＨＤＬ記述を使用して最終的に設計または製作される。

【００３６】

従って、エラー処理のようなバースト動作をどのようにすれば種々のタイプのバースト特性に基づき開始して、システムを更にフレキシブルに構成することができ、性能を向上させることができ、延いてはシステム設計を更に効率的に行なうことができるかを理解することができる。本明細書で説明した実施形態を使用してバーストリクエストを種々の異なる方法及び形態で制御することができる。

【００３７】

例えば、一つの実施形態では、一つの方法は、バーストリクエストをマスターから受信するステップと、リクエストされたバーストに対応するバースト特性を求めるステップと、複数のバーストエラープロトコルの一つを前記バースト特性に基づいて選択するステップと、選択されたバーストエラープロトコルに従って前記バーストリクエストに応答するステップと、を含む。

【００３８】

別の実施形態では、前記バースト特性を求めるステップは、前記リクエストされたバーストに対応するバーストタイプを求めることを含む。この場合、前記選択されたバーストエラープロトコルは、前記求められたバーストタイプに基づいて選択される。前記バーストタイプはリードバーストタイプまたはライトバーストタイプとすることができる。前記バーストタイプは、制限されたバーストタイプ、または制限の無いバーストタイプを含むことができる。前記バーストタイプは、インクリメントバーストタイプまたはラップバーストタイプを含むことができる。

【００３９】

更に別の実施形態では、前記バースト特性を求めるステップは、前記リクエストされたバーストに対応するマスターの識別情報を求めることを含む。この場合、前記選択されたバーストエラープロトコルは、前記マスターの識別情報に基づいて選択される。

【００４０】

更に別の実施形態では、前記選択されたバーストエラープロトコルに従って前記バーストリクエストに応答するステップは、バースト転送を開始すること、前記バースト転送の間にエラーを検出すること、及び、前記選択されたバーストエラープロトコルに基づいて前記バースト転送を選択的に終了させること、を含む。

【００４１】

別の実施形態では、メモリコントローラはバーストプロトコル選択ロジック及び制御ロジックを含む。バーストプロトコル選択ロジックは、リクエストされたバースト転送に対応する少なくとも一つのバースト特性を求め、バーストエラープロトコルを前記少なくとも一つのバースト特性に基づいて選択する。制御ロジックは前記バーストプロトコル選択ロジックに接続される。制御ロジックは、前記選択されたバーストエラープロトコルに従って、リクエストされたバーストに応答する。

【００４２】

更に別の実施形態では、前記制御ロジックは、前記リクエストされたバーストに応答して、前記リクエストされたバースト転送を開始する。この実施形態では、メモリコントローラは更に、エラー検出ロジック及び終了制御ロジックを含む。エラー検出ロジックは、前記リクエストされたバースト転送の間にエラーを検出する。終了制御ロジックは、前記リクエストされたバースト転送を、前記バーストプロトコル選択ロジックから提供されたバーストエラープロトコル（選択されたバーストエラープロトコル）に基づいて選択的に終了させる。

【００４３】

更に別の実施形態では、前記メモリコントローラはまた、バーストプロトコル選択ロジックに接続された制御レジスタを含む。前記バーストプロトコル選択ロジックは、前記バーストエラープロトコルを、少なくとも一つのバースト特性及び前記制御レジスタに保存された情報に基づいて選択する。更に別の実施形態では、前記制御レジスタに保存された情報は、複数のバーストエラープロトコルを示し、前記バーストプロトコル選択ロジックは前記バーストエラープロトコルを前記複数のバーストエラープロトコルから選択する。

更に別の実施形態では、前記複数のバーストエラープロトコルの各々は一つのバーストタイプに対応する。前記バーストタイプは、リードバーストタイプ、ライトバーストタイプ、制限されたバーストタイプ、及び制限の無いバーストタイプの少なくとも一つを含む。更に別の実施形態では、前記少なくとも一つのバースト特性は、前記リクエストされたバースト転送のバーストタイプを含む。

【００４４】

更に別の実施形態では、前記メモリコントローラは複数のマスターと通信するように構成することができる。前記複数のバーストエラープロトコルの各々は前記複数のマスターの一つに対応する。更に別の実施形態では、前記少なくとも一つのバースト特性は、前記リクエストされたバースト転送をリクエストするマスターの識別子を含む。

【００４５】

更に別の実施形態では、前記バーストプロトコル選択ロジックは、前記バーストエラープロトコルを、前記少なくとも一つのバースト特性、及びコンフィグレーション入力を介して提供される情報に基づいて選択する。更に別の実施形態では、前記コンフィグレーション入力を介して提供される情報は、バーストエラープロトコルを前記バーストプロトコル選択ロジックに対して示す。

【００４６】

更に別の実施形態では、前記メモリコントローラはハードウェアとして用いられる。更に別の実施形態では、前記メモリコントローラは、コンピュータ読み取り可能な媒体上で符号化を処理するソフトウェアとして用いられる。

【００４７】

別の実施形態では、前記メモリコントローラは、バーストリクエストをマスターから受信する手段と、前記リクエストされたバーストに対応するバースト特性を求める手段と、複数のバーストエラープロトコルの一つを前記バースト特性に基づいて選択する手段と、前記選択されたバーストエラープロトコルに従って、前記バーストリクエストに応答する手段とを含む。

【００４８】

更に別の実施形態では、前記バースト特性を求める手段は、前記リクエストされたバーストに対応するバーストタイプを求める手段を含む。この場合、前記選択されたバーストエラープロトコルは前記バーストタイプに基づいて選択される。更に別の実施形態では、前記バーストタイプは、リードバーストタイプ、ライトバーストタイプ、制限されたバーストタイプ、制限の無いバーストタイプ、インクリメントバーストタイプ、及びラップバーストタイプの少なくとも一つを含む。

【００４９】

更に別の実施形態では、前記バースト特性を求める手段は、前記リクエストされたバーストに対応するマスターの識別情報を求める手段を含む。この場合、前記選択されたバーストエラープロトコルは、前記マスターの識別情報に基づいて選択される。

【００５０】

更に別の実施形態では、前記選択されたバーストエラープロトコルに従って前記バーストリクエストに応答する手段は、バースト転送を開始する手段と、前記バースト転送の間にエラーを検出する手段と、前記選択されたバーストエラープロトコルに基づいて前記バースト転送を選択的に終了させる手段と、を含む。

【００５１】

別の実施形態では、バーストリクエストが（例えば、マスターを含む情報処理システムの）マスターから受信される。リクエストされたバーストのバーストタイプが求められる。バーストタイプは、リードバーストタイプ及びライトバーストタイプのうちの一つを含む。複数のバーストプロトコルの一つがバーストタイプに基づいて選択される。バーストリクエストに対する応答は、前記選択されたバーストプロトコルに従って行なわれる。更に別の実施形態では、前記リクエストされたバーストに対応するマスターの識別情報が求められる。この場合、前記選択されたバーストプロトコルは、バーストタイプ、及び前記

リクエストされたバーストに対応するマスターの識別情報に基づいて選択される。

【００５２】

別の実施形態では、メモリコントローラはバーストプロトコル選択ロジック、及びバーストプロトコル選択ロジックに接続された制御ロジックを含む。バーストプロトコル選択ロジックは、リクエストされたバースト転送に対応するバーストタイプを求め、バーストプロトコルを前記バーストタイプに基づいて選択する。バーストタイプはリードバーストタイプ及びライトバーストタイプの一つを含む。制御ロジックは前記選択されたバーストエラープロトコルに従って、前記リクエストされたバーストに応答する。前記メモリコントローラはハードウェアまたはソフトウェアとして用いることができる。

【００５３】

更に別の実施形態では、前記メモリコントローラはまた、バーストプロトコル選択ロジックに接続された制御レジスタを含む。バーストプロトコル選択ロジックはバーストプロトコルを、バーストタイプ及び制御レジスタに保存された情報に基づいて選択する。更に別の実施形態では、前記制御レジスタに保存された情報は複数のバーストプロトコルを示す。前記バーストプロトコル選択ロジックは前記バーストプロトコルを前記複数のバーストプロトコルから選択する。

【００５４】

別の実施形態では、一つの方法は、バーストリクエストをマスターから受信するステップと、リクエストされたバーストに対応するマスターの識別情報を求めるステップと、複数のバーストエラープロトコルの一つを前記リクエストされたバーストに対応するマスターの識別情報に基づいて選択するステップと、前記選択されたバーストエラープロトコルに従って前記バーストリクエストに応答するステップと、を含む。

【００５５】

別の実施形態では、メモリコントローラは複数のマスターと通信するように構成することができる。前記メモリコントローラはバーストプロトコル選択ロジック、及びバーストプロトコル選択ロジックに接続された制御ロジックを含む。バーストプロトコル選択ロジックは、リクエストされたバースト転送に対応する複数のマスターのうちの一つの識別情報を求め、前記複数のマスターのうちの一つの識別情報に基づいてバーストエラープロトコルを選択する。制御ロジックは、前記選択されたバーストエラープロトコルに従って、前記リクエストされたバーストに応答する。メモリコントローラはハードウェアまたはソフトウェアとして用いることができる。

【００５６】

更に別の実施形態では、前記メモリコントローラはまた、バーストプロトコル選択ロジックに接続された制御レジスタを含む。バーストプロトコル選択ロジックはバーストプロトコルを、複数のマスターのうちの一つの識別情報、及び制御レジスタに保存された情報に基づいて選択する。更に別の実施形態では、前記制御レジスタに保存された情報は複数のバーストプロトコルを示す。各プロトコルは複数のマスターの一つに対応する。バーストプロトコル選択ロジックは、前記複数のバーストプロトコルから前記バーストプロトコルを選択する。

【００５７】

別の実施形態では、一つの方法は、バーストをパラメータ化することができるハードウェア記述言語（ＨＤＬ）記述を導入するステップと、前記バーストをパラメータ化することができるＨＤＬ記述への入力として前記バーストプロトコルパラメータを供給するステップと、前記バーストプロトコルパラメータが第１の値を有する場合に、第１バーストプロトコルに従って動作するように構成されるメモリコントローラに対応する第１ＨＤＬ記述を生成するステップと、前記バーストプロトコルパラメータが第２の値を有する場合に、第２バーストプロトコルに従って動作するように構成されるメモリコントローラに対応する第２ＨＤＬ記述を生成するステップと、を含む。

【００５８】

更に別の実施形態では、前記第１バーストプロトコル及び前記第２バーストプロトコル

は異なるバーストプロトコルである。更に別の実施形態では、前記第１バーストプロトコル及び前記第２バーストプロトコルの各々は、バースト転送をそのバースト転送中のエラー検出に応答して終了させるかどうかを示す。

【００５９】

更に別の実施形態では、当該方法は、第１及び第２ＨＤＬ記述のうちの生成されたＨＤＬ記述を使用してメモリコントローラを設計するステップを含む。更に別の実施形態では、当該方法は、第１及び第２ＨＤＬ記述のうちの生成されたＨＤＬ記述を使用してメモリコントローラを製作するステップを含む。

【００６０】

別の実施形態では、コンピュータ読み取り可能な媒体上で符号化され、バーストをパラメータ化することができるＨＤＬ記述は、バーストプロトコルパラメータを受信する第１セットの命令と、前記バーストプロトコルパラメータが第１の値を有する場合に、第１メモリコントローラモデルを生成する第２セットの命令と、前記バーストプロトコルパラメータが第２の値を有する場合に、第２メモリコントローラモデルを生成する第３セットの命令と、を含む。前記第１メモリコントローラモデルは第１バーストプロトコルに従って動作するメモリコントローラに対応し、前記第２メモリコントローラモデルは第２バーストプロトコルに従って動作するメモリコントローラに対応する。

【００６１】

更に別の実施形態では、前記第１バーストプロトコル及び前記第２バーストプロトコルは異なるバーストプロトコルである。更に別の実施形態では、前記第１バーストプロトコル及び前記第２バーストプロトコルの各々は、バースト転送をそのバースト転送中のエラー検出に応答して終了させるかどうかを示す。

【００６２】

上述の実施形態のうちの幾つかの実施形態を使用する場合には、種々の異なる情報処理システムを使用して実現することができる。例えば、図１及び図１に関する説明では、例示としての情報処理アーキテクチャについて記載したが、この例示としてのアーキテクチャは単に、本発明の種々の態様を説明するための有用な参照事例となるために提示される。勿論、アーキテクチャに関する記述は説明を簡単にするために簡素化されており、かつ当該アークテクチャは、本発明に従って使用することができる多くの異なるタイプの適切なアーキテクチャの一つに過ぎない。この技術分野の当業者であれば、複数の論理ブロックの間の境界は単なる例示であり、かつ別の実施形態では複数の論理ブロックまたは複数の回路素子を統合する、または機能の別の形で分解して複数の部分機能とし、これらの部分機能を種々の論理ブロックまたは回路素子に割り当てることができることを理解し得る。

【００６３】

従って、ここに示すアーキテクチャは単なる例示であり、かつ実際に、同じ機能を実現する多くの他のアーキテクチャアを用いることができることを理解し得る。要約すると、限定的な意味ではあるが、同じ機能を実現するいかなる構成の要素群も、所望の機能を実現するように効果的に「関連付けられる」。従って、特定の機能を実現するために本明細書において組み合わせられる２つの要素は、アーキテクチャまたは中間要素がどのようなものであるかに拘らず、所望の機能を実現するように互いに「関連付けられる」ものとして理解し得る。同様に、このように関連付けられる２つの要素は必ず、互いに「動作可能に接続されて」または「動作可能に結合させて」所望の機能を実現するものとして見なし得る。

【００６４】

また例えば、一つの実施形態では、システム１０の図示の要素群は、単一の集積回路に配置される、または同じデバイスの内部に配置される回路である。別の構成として、システム１０は、相互接続されるどのような数の個別集積回路または個別デバイスも含むことができる。例えば、メモリ３５はマスター１２及び１４と同じ集積回路に、または別の集積回路に配置する、またはシステム１０の他の要素から離れて個別化された別の周辺装置

またはスレーブの内部に配置することができる。周辺装置１８及びＩ／Ｏ回路１６は、別の集積回路群またはデバイス群に配置することもできる。また例えば、システム１０または当該システムの各部分は、物理回路の、または物理回路に変換することができる論理表現のソフト表現またはコード表現とすることができる。従って、システム１０はいずれかの適切なタイプのハードウェア記述言語として具体化することができる。

【００６５】

更に、システム１０の種々の構成要素はこれらの要素の更に一般的なクラスを表わす。例えば、バスマスター１２及びバスマスター１４は、命令を実行する機能を備える、マイクロプロセッサ、プロセッサコア、デジタル信号プロセッサのようなプロセッサとすることができる、またはダイレクトメモリアクセス（ＤＭＡ）回路またはデバッグ回路のような他のいずれかのタイプの接続マスターとすることができる。周辺装置１８は、ユニバーサル非同期送受信機（ｕｎｉｖｅｒｓａｌ　ａｓｙｎｃｈｒｏｎｏｕｓ　ｒｅｃｅｉｖｅｒ　ｔｒａｎｓｍｉｔｔｅｒ：ＵＡＲＴ）、リアルタイムクロック（ＲＴＣ）、キーボードコントローラのようないずれかのタイプの周辺装置とすることができる。尚、他のスレーブ２０は、例えばマスター１２及び１４がアクセスすることができるメモリのようないずれかのタイプの接続スレーブだけでなく、周辺装置１８と同じタイプの周辺装置を含み、システムバスに位置するいずれかのタイプの周辺装置を含むことができる。Ｉ／Ｏ回路１６は、情報をシステム１０の外部から受信する、または外部に供給するいずれかのタイプのＩ／Ｏ回路を含むことができる。メモリ３５は、例えばリードオンリメモリ（ＲＯＭ）、ランダムアクセスメモリ（ＲＡＭ）、不揮発性メモリ（例えば、Ｆｌａｓｈ）、磁気ＲＡＭ（ＭＲＡＭ）などのようないずれかのタイプのコンピュータ読み取り可能な媒体とすることができる。本明細書において特定の事例を使用する場合には、特に断らない限り、特定の事例が当該事例のクラスを表わすようにしているので、特定のデバイスが本明細書において列挙される事例に含まれていない事実を、制限が望ましいことを示唆しているものと捉えてはならない。

【００６６】

システム相互接続２２はシステムバスプロトコルに従って動作するシステムバスとして用いることができる、または例えば、情報を種々のデバイスの間でルーティングするスイッチング回路のような相互接続回路を使用して用いることができる。本明細書において使用するように、「バス」という用語は複数の信号または導体を指すために用いられ、これらの信号または導体を使用して、データ、アドレス、制御、またはステータスのような一つ以上の種々のタイプの情報が伝送される。本明細書において説明する導体は、単一導体、複数の導体、単方向性導体、または双方向性導体であるものとして例示または記載される。しかしながら、異なる実施形態では、導体の実施形態を変えることができる。例えば、双方向性導体ではなく個別の単方向性導体を使用することができる、または逆に、個別の単方向導体ではなく双方向導体を使用することができる。また、複数の導体の代わりに、複数の信号をシリアルに、または時間多重方式で伝送する単一の導体を使用することができる。同様に、複数の信号を伝送する単一の導体は、これらの信号の一部分を伝送する種々の異なる導体に分離することができる。従って、信号を伝送するために多数の選択肢が存在する。

【００６７】

別の実施形態では、特定の構成要素の複数のインスタンスを組み合わせることができる。例えば、上述の実施形態では、単一の周辺装置１８を図１に示している。別の実施形態では、システム１０はシステム相互接続２２に接続されるどのような数の周辺装置も含むことができる。同様に、どのような数のマスター及びスレーブもシステム相互接続２２に接続することができ、かつ図１に示すマスター及びスレーブに制限されない。メモリ３５はどのような数の種々のタイプのメモリデバイスも含むことができる。

【００６８】

別の実施形態では、制御レジスタ３８は必要に応じて、例示の数よりも多い、または少ないビットを有する各マスターに関して例示の数よりも多い、または少ないフィールドを

含むことができる。また、制御レジスタ３８のフィールドに関して図３において説明した設定は一例として提示される。別の実施形態では、エラー応答を図３に示す属性とは異なる属性、図３の属性よりも多い属性、または図３における属性の一部分に基づいて決定することができる。尚、別の実施形態では、現在のアクセスリクエストの特定の属性を制御回路４０に対して、図１に示す信号２６，２８，及び３０によってではなく、種々の方法によって通知することができる。

【００６９】

更に、この技術分野の当業者であれば、上述の動作の機能の境界は例示に過ぎないことを理解し得る。複数の動作の機能を組み合わせて単一の動作とすることができる、そして／または単一の動作の機能を更に別の複数の動作に分割することができる。更に、別の実施形態は、特定の動作の複数のインスタンスを含むことができ、これらの動作の順番は他の種々の実施形態において変えることができる。

【００７０】

本明細書に記載するソフトウェアの全て、または幾つかは、例えばメモリ３５のようなコンピュータ読み取り可能な媒体、または他のコンピュータシステムで利用する他の媒体から受信するシステム１０の要素とすることができる。このようなコンピュータ読み取り可能な媒体は、システム１０のような情報処理システムに、半永久的に、取り外し可能に、または離れた位置から接続することができる。コンピュータ読み取り可能な媒体は、例えば本発明を制限するものではなく、どのような数の媒体も含むことができる。２～３例を挙げると、媒体は、ディスク及びテープ記録媒体を含む磁気記録媒体、コンパクトディスク媒体（例えば、ＣＤ－ＲＯＭ，ＣＤ－Ｒなど）やデジタルビデオディスク記録媒体のような光記録媒体、ＦｌａｓｈメモリやＥＥＰＲＯＭやＥＰＲＯＭやＲＯＭのような半導体メモリユニットを含む不揮発性メモリ記憶媒体、強磁性デジタルメモリ、ＭＲＡＭ、レジスタやバッファやキャッシュやメインメモリやＲＡＭなどを含む揮発性記憶媒体、及びコンピュータネットワークや、ポイントツーポイント通信用設備や、搬送波伝送媒体を含むデータ伝送媒体を含む。

【００７１】

一つの実施形態では、システム１０は、パーソナルコンピュータシステムのようなコンピュータシステムである。他の実施形態は異なるタイプのコンピュータシステムを含むことができる。コンピュータシステムは情報処理システムであり、このシステムは、個々の演算能力を一以上のユーザに提供するように構成することができる。コンピュータシステムは、これらには制限されないが、メインフレーム、ミニコンピュータ、サーバ、ワークステーション、パーソナルコンピュータ、ノートパッド、携帯情報端末、電子ゲーム、自動車用システム及び他のエンベデッドシステム、携帯電話機及び他の種々の無線機器を含む多数の形態に含まれる。通常のコンピュータシステムは、少なくとも一つの処理ユニット、関連メモリ、及び多数の入力／出力（Ｉ／Ｏ）デバイスを含む。

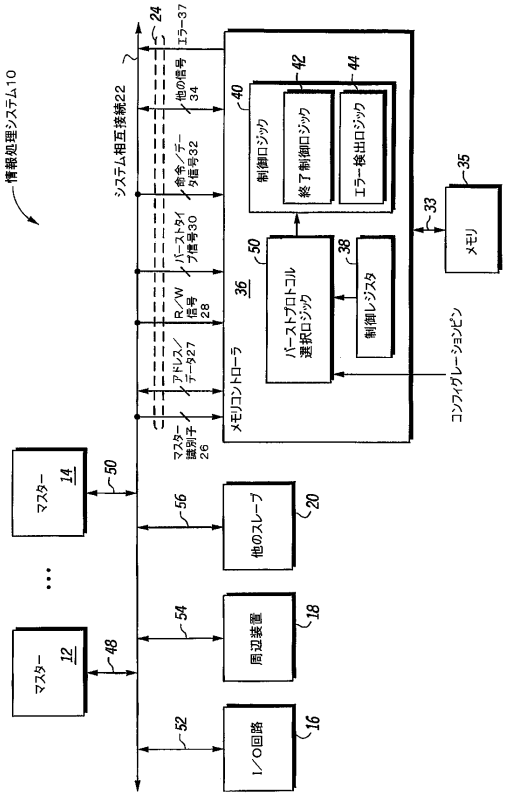
【００７２】

コンピュータシステムは情報をプログラムに従って処理し、結果としての出力情報をＩ／Ｏデバイスを介して生成する。プログラムは、特定のアプリケーションプログラム及び／又はオペレーティングシステムのような命令リストである。コンピュータプログラムは通常、コンピュータ読み取り可能な記録媒体に内部的に格納されるか、またはコンピュータシステムにコンピュータ読み取り可能な伝送媒体を介して送信される。コンピュータプロセスは通常、実行（動作）プログラムまたはプログラムの一部分、現在のプログラム値及び状態情報、及びオペレーティングシステムがプロセスの実行を管理するために使用するリソースを含む。親プロセス（ｐａｒｅｎｔ　ｐｒｏｃｅｓｓ）は他の子プロセス（ｃｈｉｌｄ　ｐｒｏｃｅｓｓ）を生み出して、親プロセスの機能全体を実行し易くすることができる。親プロセスは特殊な形で子プロセスを生み出して親プロセスの機能全体の一部分を実行することができるので、子プロセス（及び孫プロセス（ｇｒａｎｄｃｈｉｌｄ　ｐｒｏｃｅｓｓ）など）が実行する機能は、親プロセスが実行するものとして記述することができる場合がある。

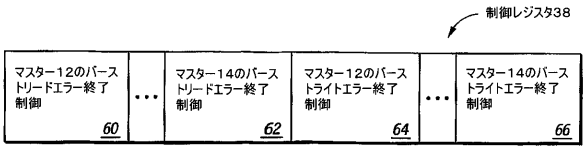
【００７３】
上述の詳細な説明は例示であるので、「一つの実施形態」について記載する場合、当該実施形態は例示としての実施形態である。従って、このような関係における「一つの」という単語は、一つの実施形態のみが記載される特徴を有することができることを意味するために使用されるのではない。そうではなく、他の多くの実施形態が、例示としての「一つの実施形態」に関して記載される特徴を有することができ、かつ有することができる場合が多い。従って、上に使用されるように、本発明が一つの実施形態に関連して記載される場合、当該一つの実施形態は本発明の多数の可能な実施形態のうちの一つである。
【００７４】
詳細な説明における「一つの実施形態」という用語の使用に関する上記注釈にも拘らず、この技術分野の当業者は、以下に示す請求要素の特定の「数」が以下の請求項において或る意図として設定される場合、このような意図は請求項において明示的に引用されることになり、このような引用が無い状態では、このような制限は無い、または必要ではないことを理解し得る。例えば、請求項において、請求要素が「一つの」特徴を有するものとして記載される場合、当該要素が記載の特徴の一つ、かつ一つのみに制限されることを意図している。更に、請求要素が請求項において、「或る」特徴を含む、または備えるものとして記載される場合、当該要素が記載の特徴の一つ、かつ一つのみに制限されることを意図しているのではない。そうではなく、例えば「或る」特徴を含む請求項は、説明対象の特徴の一つ以上を含む装置または方法として解釈される。すなわち、説明対象の装置または方法は或る特徴を含むので、請求項は、装置または方法が別のこのような同様の特徴を含むかどうかには関係のない装置または方法として解釈される。或る請求項の或る特徴に関する非制限的な前置冠詞としての「或る」という単語のこのような用法は、反対の特異な、または先例としての判例を見付け出すことができるとしても、出願人によって本明細書において過去の多数の法廷が採用した解釈と同じであるとして採用される。同様に、請求要素が請求項において、前述した特徴（例えば、「前記」特徴）を含む、または備えるものとして記載される場合、当該要素は、記載される特徴の一つ、かつ一つのみに、限定冠詞を付随的に使用することによって単純に制限される、というものではないものとする。
【００７５】
更に、請求項における「少なくとも一つの」及び「一つ以上の」のような前置語句の使用は、同じ請求項が、「一つ以上の」及び「少なくとも一つの」という前置語句、及び「或る」のような非限定冠詞を含む場合でも、非限定冠詞「或る」を用いて別の請求要素を導入することが、このような請求要素を含む全ての特定の請求項を、このような要素を一つのみ含む発明に限定することになるものとして捉えられるべきではない。同じことが、限定冠詞を使用する場合にも当てはまる。
【００７６】
本発明の特定の実施形態について示し、そして説明してきたが、この技術分野の当業者は、本明細書における示唆に基づいて、種々の変形物、代替構成要素、及び均等物を、本明細書において請求する発明から逸脱しない範囲において使用することができることを理解し得る。従って、添付の請求項は、これらの請求項が示す技術範囲に、全てのこのような変更、変形などを、本発明の真の技術思想及び技術範囲のものであるとして含む。更に、本発明は添付の請求項によってのみ規定されることを理解し得る。上記説明は本発明の実施形態を全て網羅したリストを提示するものではない。特に断らない限り、非制限的、包括的な用語、または同様の用語が各例に関して同時に表現されるかどうかに関係なく、本明細書において提示される各例は非制限的な、または包括的な例である。幾つかの例示としての実施形態及びこれらの実施形態に関する例示としての変形例を概括しようと試みてきたが、他の実施形態及び／又は変形例は以下の請求項に規定される本発明の技術範囲に含まれる。
【図面の簡単な説明】
【００７７】

【図１】本発明の一つの実施形態による情報処理システムを示すブロック図。
【図２】本発明の一つの実施形態による、図１のシステムの制御レジスタを示すブロック図。
【図３】本発明の一つの実施形態による、図２の制御レジスタのフィールド記述を示すテーブル図。
【図４】本発明の一つの実施形態による、バーストリクエストをマルチバーストプロトコルで処理する方法を示すフロー図。
【図５】本発明の一つの実施形態による、ハードウェア記述言語をマルチバーストプロトコルで処理する方法を示すフロー図。

【図１】



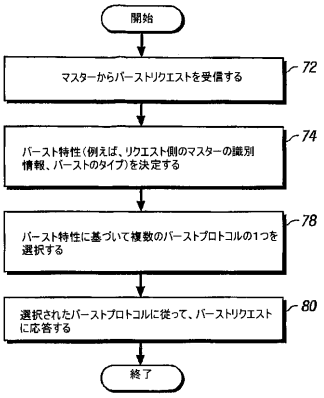
【図２】



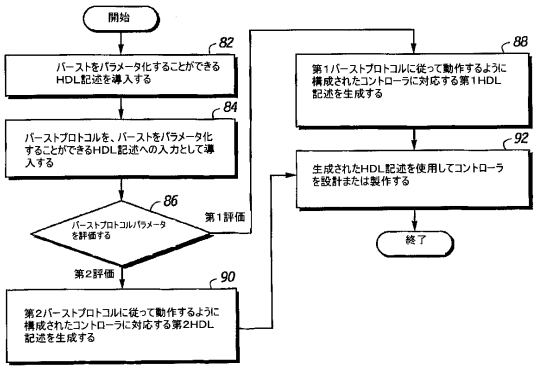
【図３】

名前	記述	設定
バーストリードエラー終了制御フィールド60及び62	マスターXのエラー終了制御－これらのビットを使用してバーストアクセスリードエラー時に使用されるプロトコルを制御する	00 － リードバースト転送中に生じるエラーはリードバーストの残りのビートに影響しない 01 － リードバースト転送中に生じるエラーによってリードバーストが中断する 10 － 第1ビート(「クリティカルビート」)の間に生じるエラーによってリードバーストが終了する;他のエラーによってリードバーストが終了することはない 11 － 未定義長のリードバースト中に生じるエラーによってリードバーストが終了する;他のリードバーストタイプに関するエラーによってリードバーストが終了することはない
バーストライトエラー終了制御フィールド64及び66	マスターXのエラー終了制御－これらのビットを使用してバーストアクセスライトエラー時に使用されるプロトコルを制御する	00 － ライトバースト転送中に生じるエラーはライトバーストの残りのビートに影響しない 01 － ライトバースト転送中に生じるエラーによってライトバーストが中断する 10 － 第1ビート(「クリティカルビート」)の間に生じるエラーによってライトバーストが終了する;他のエラーによってライトバーストが終了することはない 11 － 未定義長のライトバースト中に生じるエラーによってライトバーストが終了する;他のライトバーストタイプに関するエラーによってライトバーストが終了することはない

【図４】



【図５】



フロントページの続き

(56)参考文献 特開平１１－０８５６７３（ＪＰ，Ａ）
特開平０８－０４４６６５（ＪＰ，Ａ）
特開平０５－２０４８４５（ＪＰ，Ａ）
特表２００４－５３１８３０（ＪＰ，Ａ）
国際公開第２００３／００１３８８（WO，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 12/02
G06F 13/28