(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-234545
(P2004-234545A)
(43)公開日　平成16年8月19日(2004.8.19)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 12/16	G06F	12/16	320A	5B001
G06F 11/08	G06F	11/08	310A	5B018

審査請求 有　請求項の数 12　ＯＬ　（全 11 頁）

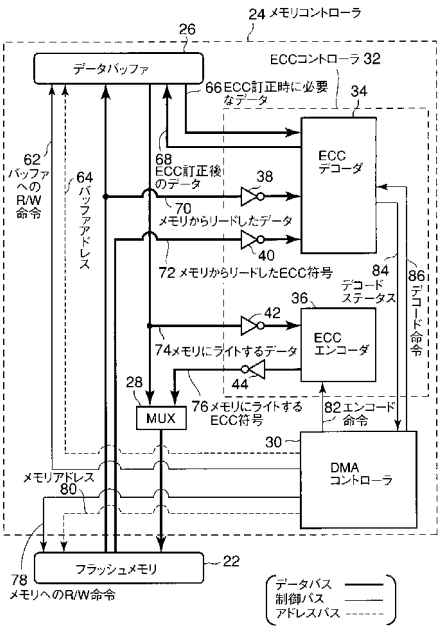
(21)出願番号　特願2003-24861(P2003-24861)
(22)出願日　平成15年1月31日(2003.1.31)

(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(74)代理人　100058479
弁理士　鈴江　武彦
(74)代理人　100091351
弁理士　河野　哲
(74)代理人　100088683
弁理士　中村　誠
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100084618
弁理士　村松　貞男
(74)代理人　100092196
弁理士　橋本　良郎

最終頁に続く

(54)【発明の名称】制御回路及びメモリコントローラ

(57)【要約】
【課題】消去後の初期値に対してＥＣＣエラーを誤検出することがなく、動作テストも容易な制御回路及びメモリコントローラを提供すること。
【解決手段】フラッシュメモリ（２２）にＥＣＣ符号として、データの反転値から計算されたリードソロモン符号の反転値が書き込まれる。データのビット列がオール“１”の場合は、オール“１”のＥＣＣ符号が書き込まれる。フラッシュメモリ（２２）から消去後の初期値（オール“１”）を読み出す場合、ＥＣＣデコーダ（３４）に入力されるデータのビット列はオール“０”、ＥＣＣ符号もオール“０”となるため、ＥＣＣエラーが検出されることがない。
【選択図】　図２



【特許請求の範囲】
【請求項１】
記憶装置から読み出したデータを全ビット反転する反転手段と、
前記反転手段の出力をエラー訂正復号処理する復号手段と、
を具備する制御回路。
【請求項２】
前記復号手段は記憶装置のデータ消去後の初期値の全ビット反転値に対してエラー無しを検出する請求項１に記載の制御回路。
【請求項３】
前記記憶装置は不揮発性半導体記憶装置を具備する請求項１に記載の制御回路。
【請求項４】
記憶装置に書き込まれるデータを全ビット反転する第１の反転手段と、
前記第１の反転手段の出力をエラー訂正符号化処理する符号化手段と、
前記符号化手段から出力されるデータを全ビット反転して記憶装置に書き込む第２の反転手段と、
記憶装置から読み出したデータを全ビット反転する第３の反転手段と、
前記第３の反転手段の出力をエラー訂正復号処理する復号手段と、
を具備する制御回路。
【請求項５】
前記符号化手段の符号化方式は、前記復号手段が記憶装置のデータ消去後の初期値の全ビット反転値に対してエラー無しを検出する方式である請求項４に記載の制御回路。
【請求項６】
前記復号手段は記憶装置のデータ消去後の初期値の全ビット反転値に対してエラー無しを検出する請求項４に記載の制御回路。
【請求項７】
前記記憶装置は不揮発性半導体記憶装置を具備する請求項４に記載の制御回路。
【請求項８】
記憶装置に接続されるメモリコントローラにおいて、
データを一時的に保持するバッファと、
前記バッファから記憶装置に書き込まれるデータを全ビット反転する第１の反転手段と、
前記第１の反転手段の出力をエラー訂正符号化処理する符号化手段と、
前記符号化手段から出力されるデータを全ビット反転して記憶装置に書き込む第２の反転手段と、
記憶装置から読み出したデータを全ビット反転する第３の反転手段と、
前記第３の反転手段の出力をエラー訂正復号処理する復号手段と、
を具備するメモリコントローラ。
【請求項９】
前記符号化手段の符号化方式は、前記復号手段が記憶装置のデータ消去後の初期値の全ビット反転値に対してエラー無しを検出する方式である請求項８に記載のメモリコントローラ。
【請求項１０】
前記復号手段は記憶装置のデータ消去後の初期値の全ビット反転値に対してエラー無しを検出する請求項８に記載のメモリコントローラ。
【請求項１１】
前記バッファの出力と、前記符号化手段の出力とを選択的に記憶装置に供給する手段をさらに具備する請求項８に記載のメモリコントローラ。
【請求項１２】
前記記憶装置は不揮発性半導体記憶装置を具備する請求項８に記載のメモリコントローラ。
【発明の詳細な説明】

【０００１】
【発明の属する技術分野】
本発明は半導体記憶装置等のエラーチェック訂正（Ｅｒｒｏｒ　Ｃｈｅｃｋｉｎｇ　ａｎｄ　Ｃｏｒｒｅｃｔｉｏｎ　ｏｒ　Ｅｒｒｏｒ　Ｃｏｒｒｅｃｔｉｏｎ　Ｃｏｄｅ）（以下、ＥＣＣと略称する）に関する制御回路及びメモリコントローラに関する。
【０００２】
【従来の技術】
不揮発性半導体記憶装置の一例としてフラッシュメモリがある。例えば、ＮＡＮＤ型のフラッシュメモリでは、５１２バイト単位のブロック毎に書込みが行われ、ブロックは不連続でも構わないが、書込みアドレスは増加しながら書き込まれるものもある。そのため、不連続な複数のブロックに一旦データを書き込んだ後、中間のブロック（アドレスが小さい）にデータを書込みたい場合は、一旦、他の領域へデータを移してから、アドレスが増加するブロックに順番に書き込む必要がある。この時、データが書き込まれていない領域からデータを読み出すことになるが、データが書き込まれていない領域は、データが消去されている消去領域である。不揮発性半導体記憶装置において、データの消去を行うと、その領域のデータはある初期値（例えば、すべてのビットが“１”）となる。
【０００３】
一方、不揮発性半導体記憶装置に書き込まれるデータはエラー訂正符号化処理され、本来の書込みデータに対してＥＣＣビット（１ビット、あるいは数ビット）が付加されている。不揮発性半導体記憶装置から読み出したデータはエラー訂正復号処理される。消去領域のデータは例えば、ビット列がオール“１”であるが、通常のＥＣＣ符号、例えば拡張ハミング符号では、ビット列がオール“１”、オール“０”のデータに対するＥＣＣビットはオール“１”、オール“０”にはならない。このため、消去領域から読み出したデータは誤まりは無い（正しい）にも係わらず、ＥＣＣエラー有りと誤判定されてしまう。
【０００４】
このようなＥＣＣエラー誤判定を防ぐために、ＥＣＣ制御回路内にエンコーダ用、デコーダ用の２つのインバータを内蔵し、ＥＣＣエンコーダの出力、ＥＣＣデコーダの出力の適宜ビットを反転することにより、ＥＣＣエラーが誤検出されないようにすることが考えられている（特許文献１参照）。
【０００５】
特許文献１に記載のＥＣＣ制御回路は、チェックビット生成回路、シンドロームデコーダ（ＥＣＣデコーダ）、チェックビット生成回路から生成されたチェックビットの少なくとも一部を反転する第１のビットインバータ、チェックビット格納用不揮発性メモリから読み出したチェックビットの少なくとも一部を反転する第２のビットインバータから構成されている。
【０００６】
第１のビットインバータは、データ領域の初期値に対して生成されるチェックビットが消去後の初期値と等しくなるようにその一部を反転する。第２のビットインバータは、チェックビット格納用不揮発性メモリから読み出したチェックビットについて第１のビットインバータと同じビットの反転を行う。
【０００７】
データ書込みの際は、データをデータ用不揮発性メモリへ書き込むとともにチェックビット生成回路へ供給する。チェックビット生成回路は、データに応じたチェックビットを生成する。第１のビットインバータは、このチェックビットを、ＥＣＣエラーが発生しないように適宜ビット反転させ、チェックビット格納用不揮発性メモリに書き込む。
【０００８】
データ読出しの際は、データをデータ用不揮発性メモリから読み出し、ＥＣＣ制御回路に供給し、チェックビットをチェックビット格納用不揮発性メモリから読出し、第２のビットインバータを介して、その一部が反転されてシンドロームデコーダへ供給される。シンドロームデコーダはエラーがあれば適宜修正し、修正後のデータをデータバスへ送出する

。
【０００９】
このように、消去領域から読み出したデータからチェックビットを生成し、この一部のビットをエラーが発生しないように適宜反転させることにより、ＥＣＣエラーの誤検出を防止している。
【００１０】
【特許文献１】
特開２００１－９２７２３号公報（段落００１３～段落００１４、図３）
【００１１】
【発明が解決しようとする課題】
特許文献１記載では、ＥＣＣ符号の特性を考慮していないため、ビットインバータが反転するビットは固定ではないので、インバータが複雑になる。一般に、集積回路は回路記述をした後、シミュレーションテストが必要であるが、複雑なインバータはテスト項目数も多く、テストに時間がかかる欠点がある。
【００１２】
この発明の目的は、消去後の初期値に対してＥＣＣエラーを誤検出することがなく、動作テストも容易な制御回路及びメモリコントローラを提供することである。
【００１３】
【課題を解決するための手段】
上記した課題を解決し目的を達成するために、本発明は以下に示す手段を用いている。
【００１４】
制御回路は、
記憶装置から読み出したデータを全ビット反転する第１の反転手段と、
前記第１の反転手段の出力をエラー訂正復号処理する復号手段と、
を具備する。
【００１５】
記憶装置に接続されるメモリコントローラにおいて、
データを一時的に保持するバッファと、
前記バッファから記憶装置に書き込まれるデータを全ビット反転する第１の反転手段と、
前記第１の反転手段の出力をエラー訂正符号化処理する符号化手段と、
前記符号化手段から出力されるデータを全ビット反転して記憶装置に書き込む第２の反転手段と、
記憶装置から読み出したデータを全ビット反転する第３の反転手段と、
前記第３の反転手段の出力をエラー訂正復号処理する復号手段と、
を具備する。
【００１６】
この発明によれば、簡単な構成で、記憶装置の消去後の初期値に対してＥＣＣエラーを誤検出することがなく、動作テストも容易である。
【００１７】
【発明の実施の形態】
以下、図面を参照して本発明による制御回路及びメモリコントローラの実施の形態を説明する。ここでは、説明の便宜上、不揮発性半導体記憶装置はフラッシュメモリであるとし、消去後の初期値のビット列はオール“１”であるとし、誤り訂正符号をリードソロモン符号とする。図１はリードソロモン符号化回路への入力データとその出力符号との関係を表す。リードソロモン符号は入力データのＥＸＯＲの組み合わせで構成されているため、データのビットがオール“０”の場合は、リードソロモン符号もオール“０”になるが、データのビットがオール“１”の場合は、リードソロモン符号はほとんどの場合オール“１”にはならない。このため、消去後の初期値がオール“１”ということは、データビットと符号ビットがともにオール“１”ということであり、ＥＣＣエラー有りと判定されてしまう。

【００１８】
第１の実施の形態
図２は本発明の第１の実施の形態に係る制御回路を含むシステムの全体構成を示す図である。本実施の形態は、消去後の初期値からＥＣＣエラーが検出されないようにデータバスやＥＣＣ符号バスにインバータを接続するものである。
【００１９】
フラッシュメモリ２２にデータバス、制御バス、アドレスバスを介してメモリコントローラ２４が接続される。メモリコントローラ２４はデータバッファ２６、マルチプレクサ２８、ＤＭＡコントローラ３０、ＥＣＣコントローラ３２からなる。データバッファ２６はフラッシュメモリ２２へ書き込むデータ、フラッシュメモリ２２から読み出すデータを一時的に保持する。ＥＣＣコントローラ３２はフラッシュメモリ２２へ書き込むデータにＥＣＣ符号化処理、フラッシュメモリ２２から読み出すデータにＥＣＣ復号処理を施す。マルチプレクサ２８はデータバッファ２６からのデータとＥＣＣコントローラ３２からのデータを選択的にフラッシュメモリ２２へ供給する。ＤＭＡコントローラ３０は、フラッシュメモリ２２、データバッファ２６にアドレスデータ、制御データを供給するとともに、ＥＣＣデコーダ３４、ＥＣＣエンコーダ３６に命令を供給する。
【００２０】
ＥＣＣコントローラ３２は、データバッファ２６から供給されるデータに基づいてＥＣＣ符号を作成するＥＣＣエンコーダ３６、メモリから読み出したデータとＥＣＣ符号とに基づいてデータのエラーがあるか否かを判定し、エラーが訂正できる場合は訂正するＥＣＣデコーダ３４、ＥＣＣデコーダ３４へ供給されるデータ（メモリから読み出したデータ）の全ビットを反転するインバータ３８、ＥＣＣデコーダ３４へ供給されるＥＣＣ符号（メモリから読み出したＥＣＣ符号）の全ビットを反転するインバータ４０、ＥＣＣエンコーダ３６へ供給されるデータ（フラッシュメモリ２２に書き込まれるデータ）の全ビットを反転するインバータ４２、ＥＣＣエンコーダ３６から出力されるＥＣＣ符号の全ビットを反転するインバータ４４からなる。
【００２１】
上記構成のシステムの動作を説明する。
【００２２】
データ書き込み
フラッシュメモリ２２に書き込まれるデータがデータバッファ２６に一旦保持される。ＤＭＡコントローラ３０がデータバッファ２６に対してデータバッファ用命令線６２にリード命令を、データバッファ用アドレス線６４に所望のアドレスを発行することにより、データバッファ２６からデータ（メモリにライトするデータ）をリードする。ＤＭＡコントローラ３０がフラッシュメモリ２２に対してフラッシュメモリ用命令線７８にライト命令を、フラッシュメモリ用アドレス線８０に所望のアドレスを発行することにより、マルチプレクサ２８を介してデータをフラッシュメモリ２２に書き込む。ＤＭＡコントローラ３０はＥＣＣエンコーダ３６に対してエンコード命令線８２にエンコード開始命令を送り、データバッファ２６－フラッシュメモリ２２間データバスからスヌープされたデータ７４をインバータ４２で反転し、反転データをＥＣＣエンコーダ３６へ送る。データバッファ２６からフラッシュメモリ２２へデータを転送している際は、マルチプレクサ２８はデータを通過させる処理を行う。マルチプレクサ２８の出力であるフラッシュメモリ入力データバスを通してデータを転送し終わった後、ＥＣＣエンコーダ３６はデータバッファ２６から読み出したデータの反転値に対してＥＣＣ符号を生成し、データ反転機能付きマルチプレクサ入力バス７６を通じてＥＣＣ符号の反転値を送信する。マルチプレクサ２８はＥＣＣ符号を通過させる処理を行い、このＥＣＣ符号の反転値がフラッシュメモリ２２へ書き込まれる。
【００２３】
上記手順に従いフラッシュメモリ２２へオール“１”のデータを書き込むと、ＥＣＣエンコーダ３６に入力されるデータは反転してオール“０”となるため、ＥＣＣエンコーダ３

６ではオール“０”のＥＣＣ符号が生成される。これを反転してフラッシュメモリ２２に書き込むため、データ、ＥＣＣ符号ともにオール“１”が書き込まれる。

【００２４】

データ読み出し

ＤＭＡコントローラ３０がフラッシュメモリ２２に対してデータバッファ用命令線７８にリード命令を、フラッシュメモリ用アドレス線８０に所望のアドレスを発行することにより、フラッシュメモリ２２からデータをリードする。ＤＭＡコントローラ３０がデータバッファ２６に対してデータバッファ用命令線６２にライト命令を、データバッファ用アドレス線６４に所望のアドレスを発行することにより、フラッシュメモリ２２から読み出したデータをフラッシュメモリ－データバッファ間データバスを通してデータバッファ２６に書き込む。ＤＭＡコントローラ３０はＥＣＣデコーダ３４に対してデコード命令線８６を通じてデコード開始命令を送り、インバータ３８の出力であるＥＣＣデコーダ入力用データ反転バスからリードデータの反転値を、インバータ４０の出力であるＥＣＣデータ入力用ＥＣＣ符号反転バスからＥＣＣ符号の反転値をＥＣＣデコーダ３４へ送る。

【００２５】

ＥＣＣデコーダ３４のデコードの結果、データにエラーがあり修正が必要であった場合は、データ訂正動作が開始する。ＥＣＣデコーダ３４が訂正箇所を特定・訂正する場合は、デコードステータス線８４をビジー状態とし、訂正箇所特定・訂正中であることをＤＭＡコントローラ３０に通知し、データバッファ２６へのリード・ライト動作を禁止させる。次に、ＥＣＣデコーダ３４はデータバッファ２６から訂正箇所の情報をＥＣＣ訂正用リードデータバス６６を通してリードする。ＥＣＣデコーダ３４は訂正すべきビット位置のエラーデータをビット反転により訂正し、訂正後のデータをＥＣＣ訂正用ライトデータバス６８を通してデータバッファ２６にライトする。

【００２６】

上記手順に従いフラッシュメモリ２２からオール“１”のデータを読み出すと、ＥＣＣデコーダ３４に入力されるデータは反転してオール“０”となり、ＥＣＣ符号も反転してオール“０”（上記書き込み手順を参照）であるため、図１に示すように、ＥＣＣエラーが検出されることがない。

【００２７】

以上説明したように、本実施の形態によれば、フラッシュメモリ２２へ書き込む場合、フラッシュメモリ２２にはＥＣＣ符号として「データの反転値から計算されたリードソロモン符号の反転値」が書き込まれる。つまり、データ領域のビットがオール“１”の場合は、オール“１”のＥＣＣ符号が書き込まれる。フラッシュメモリ２２から消去後の初期値（オール“１”）を読み出す場合、ＥＣＣデコーダ３４に入力されるデータはオール“０”、ＥＣＣ符号もオール“０”となるため、ＥＣＣエラーが検出されることがない。

【００２８】

メモリ消去後の状態に対して、通常通りＥＣＣ判定を行うとエラーが誤検出されてしまう。特許文献１ではメモリ消去後のデータからＥＣＣ符号を生成し、この符号データをエラーが発生しないように適宜反転させて対応しているのに対し、本実施の形態ではメモリ消去後のデータの全ビットを反転させた後、ＥＣＣ制御回路に入力し、ＥＣＣ符号を生成している。特許文献１では、ＥＣＣエンコーダ・デコーダ内にインバータが実装されていることになるため、ＥＣＣエンコーダ・デコーダの単体テストが難しいが、本実施の形態では既存のＥＣＣエンコーダ・デコーダの外部にインバータを接続しているため、ＥＣＣエンコーダ・デコーダの単体テストが容易である。

【００２９】

以下、本発明による制御回路の他の実施の形態を説明する。他の実施の形態の説明において第１の実施の形態と同一部分は同一参照数字を付してその詳細な説明は省略する。

【００３０】

第２の実施の形態

図２に示す第１の実施の形態では、ＥＣＣ制御はＤＭＡコントローラ３０が司っているが

、ＤＭＡコントローラ３０の代わりにＣＰＵを用いて制御を行うこともできる。図３はこれを実現する第２の実施の形態のブロック図である。すなわち、ＤＭＡコントローラ３０の代わりにＣＰＵ５０が設けられる。ＣＰＵ５０を動作させるためのプログラムを格納するＲＯＭモジュール５２、ＣＰＵ５０の作業領域となるＲＡＭモジュール５４が必要となる。ＤＭＡコントローラがＥＣＣ制御を行う場合と比較して異なる制御を行うのは以下の２点である。

【００３１】

（１）ＥＣＣ制御を行う前に、ＣＰＵ５０はプログラムリード用データ線９２を通じてＲＯＭモジュール５２からプログラムをロードする。

【００３２】

（２）ＣＰＵ５０はＥＣＣ制御を行う際に、ＲＡＭアクセス用データ線９４を通じて作業領域としてＲＡＭモジュール５４を使用する。

【００３３】

その他の制御は、第１の実施の形態と同様である。

【００３４】

以上説明したように、第２の実施の形態によっても、第１の実施の形態と同じ効果が得られる。

【００３５】

次に、上述した構成の第１、第２の実施の形態に適用されるＬＳＩのテストを説明する。

【００３６】

図４は一般的なＬＳＩ設計フローを示す。ステップＳ１０で、チップやパッケージ、コスト、周波数、回路規模見積もり等のＬＳＩ外部仕様を決定する。ステップＳ１２で、外部仕様に従って、ＬＳＩ内部の各モジュールの担当機能（内部仕様）を決定する。ステップＳ１４で、内部仕様に従い、ハードウェア記述言語を用いて回路を記述する。ステップＳ１６で、記述された回路の機能が正常に動作しているかどうかを検証する（機能シミュレーション）。ステップＳ１８で、シミュレーションが成功（回路機能が正常に動作している）か否かを判定する。

【００３７】

失敗の場合は、回路記述（ステップＳ１４）に戻る。成功した場合は、ステップＳ２０で、ハードウェア記述言語で記述された回路をＡＮＤゲートやＯＲゲート等の論理ゲートに変換する（論理合成）。論理ゲートに変換された回路を以下「ネット」と称する。

【００３８】

ステップＳ２２で、ネットがハードウェア記述言語で記述された回路と同等の機能を持っているかどうかを検証する（論理シミュレーション）。検証内容は機能シミュレーションの検証テストと似通っている事が多い。ステップＳ２４で、ネットが希望動作周波数で正常に動作するかどうかを検証する（タイミング解析）。なお、ステップ２２と２４は並列で作業ができるため、どちらを先に実行しても良い。ステップＳ２６で、論理シミュレーションとタイミング解析が成功（ネットが正常に動作している）か否かを判定する。

【００３９】

機能シミュレーションにおいて正常な動作をしなかった場合、あるいはタイミング解析が成功しなかった場合には、回路記述を変更し、機能シミュレーションをやり直す（ステップＳ１４に戻る）。また、論理シミュレーションやタイミング解析でネットに異常が発見された場合は、論理合成をやり直したり（ステップＳ２０に戻る）、回路記述までさかのぼってやり直す（ステップＳ１４に戻る）。成功した場合は、ステップＳ２８で、実際にＬＳＩ基板上に論理ゲートを配置、配線する（配置配線）。

【００４０】

図４の機能シミュレーション（ステップＳ１６、Ｓ１８）の詳細を図５に示す。図中の単体機能とはＥＣＣデコーダ３４、ＥＣＣエンコーダ３６、ＤＭＡコントローラ３０等の単体のモジュールの機能を指す。これに対して、複合機能とはＥＣＣデコーダ３４とインバータ３８、４０、ＥＣＣエンコーダ３６とインバータ４２、４４、ＥＣＣデコーダ３４と

ＥＣＣエンコーダ３６とインバータ３８、４０、４２、４４等のように複数のモジュールが組み合わさった機能を指す。ほとんどの場合、単体機能に比べて複合機能は機能が複雑になるため、その機能を検証するテストは多岐に渡り、テストパターンの作成や検証方法が難しくなることが多い。このためハードウェア言語で記述された回路を検証するためには、図５のように先ず単体機能テストを行って、単体の機能が正しいことが検証されてから、複合機能テストを行うのが一般的である。

【００４１】

例としてＥＣＣデコーダ３４を挙げると、ＥＣＣデコーダ３４の機能を検証するためには、ステップＳ５２で、単体機能のためのテストパターンを作成する。ステップＳ５４で、単体機能のシミュレーションを行う。ステップＳ５６で、シミュレーションが成功か失敗か、失敗の場合、その原因がテストパターンにあるのか、回路記述にあるのかを判定し、原因に該当するステップに戻り、やり直す。

【００４２】

ステップＳ５８で、複合機能（例えば、ＥＣＣデコーダとインバータを組み合わせた機能）テストのためのテストパターンを作成する。ステップＳ６０で、複合機能のシミュレーションを行う。ステップＳ６２で、シミュレーションが成功か失敗か、失敗の場合、その原因がテストパターンにあるのか、回路記述にあるのかを判定し、原因に該当するステップに戻り、やり直す。複合機能テストでは単体機能テストと異なり、それぞれのモジュールの機能の組み合わせの数（複合機能数）だけテストパターンが存在する。このテストパターンの数が多いと、それだけテスト作成に時間がかかるとともに、そのテストのシミュレーション時間が増大するため、ＬＳＩ開発が現実的には困難となっている。

【００４３】

上述した本実施の形態のデータ、ＥＣＣ符号を反転させるインバータは、リードソロモン符号の特性を利用して一意に全ビットを反転させているため、機能シミュレーションのためのテストパターンは１パターンである。しかし、特許文献１のインバータは、データ用フラッシュメモリやチェックビット用フラッシュメモリの特性に合わせて適宜チェックビットを反転させる必要があり、そのテストパターンは２のチェックビット乗必要となる。このため、複合機能テストパターンの作成や複合機能シミュレーションを２のチェックビット乗回数だけ実施する必要があり、全テストパターンだけ機能シミュレーションを実施することは非常に困難である。論理シミュレーションもこのテストパターンを使用するので、論理シミュレーションの実施も困難である。

【００４４】

本発明は上述した実施の形態に限定されず、種々変形して実施可能である。例えば、上述の説明では、ＥＣＣ符号としてはリードソロモン符号を説明したが、他のＥＣＣ符号を用いてもよい。例えば、オール“１”の時にＥＣＣエラーが必ずオール“０”のときにＥＣＣエラーになるＥＣＣ符号化方式と消去後オール“０”になるメモリでも実装可能である。また、メモリの消去後の初期値はオール“１”としたが、これに限られない。

【００４５】

なお、本願発明は上記実施の形態に限定されるものではなく、実施段階ではその趣旨を逸脱しない範囲で種々に変形することが可能である。また、各実施の形態は可能な限り適宜組み合わせて実施してもよく、その場合組合わせた効果が得られる。さらに、上記実施の形態には種々の段階の発明が含まれており、開示される複数の構成要件における適宜な組合わせにより種々の発明が抽出され得る。例えば、実施の形態に示される全構成要件から幾つかの構成要件が削除されても、発明が解決しようとする課題の欄で述べた課題が解決でき、発明の効果の欄で述べられている効果が得られる場合には、この構成要件が削除された構成が発明として抽出され得る。

【００４６】

【発明の効果】

以上説明したように本発明によれば、記憶装置の消去後の初期値に対してＥＣＣエラーを誤検出することがなく、動作テストも容易な制御回路及びメモリコントローラを提供する

ことができる。

【図面の簡単な説明】

【図１】リードソロモン符号方式の概要を示す図。

【図２】本発明による制御回路の第１の実施の形態の構成を示すブロック図。

【図３】本発明による制御回路の第２の実施の形態の構成を示すブロック図。

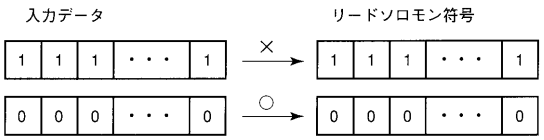
【図４】本発明による制御回路の第１、第２の実施の形態に適用される動作テストを示すフローチャート。

【図５】図４に示すフローチャートの機能シミュレーションの詳細を示すフローチャート。

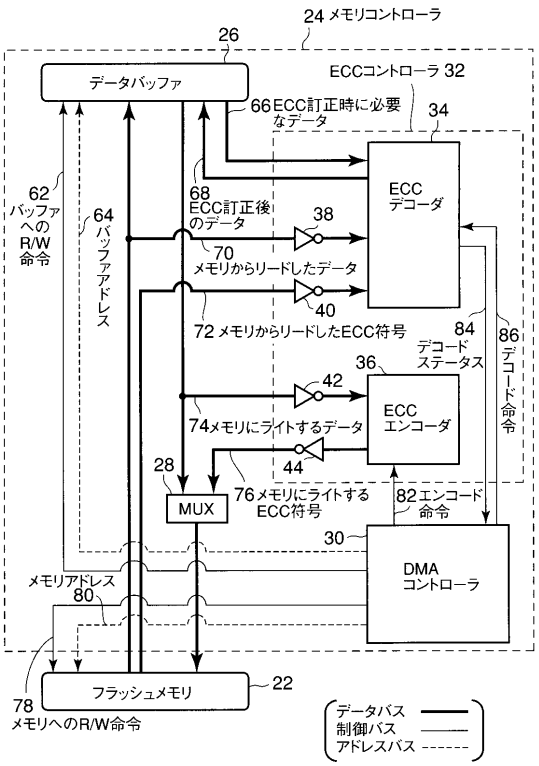
【符号の説明】

２２…フラッシュメモリ、２４…メモリコントローラ、２６…データバッファ、２８…マルチプレクサ、３０…ＤＭＡコントローラ、３２…ＥＣＣコントローラ、３４…ＥＣＣデコーダ、３６…ＥＣＣエンコーダ、３８、４０、４２、４４…インバータ

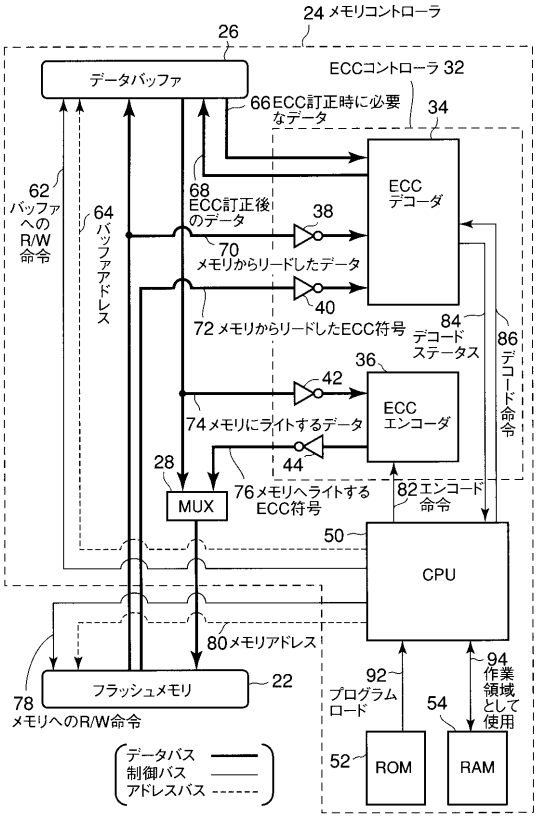
【図１】



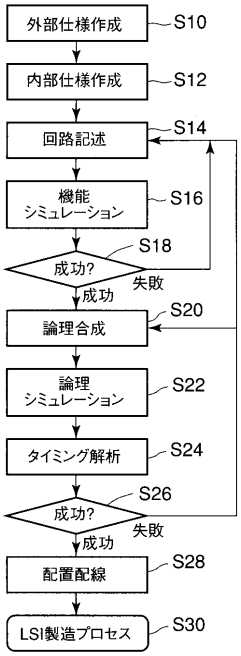
【図２】



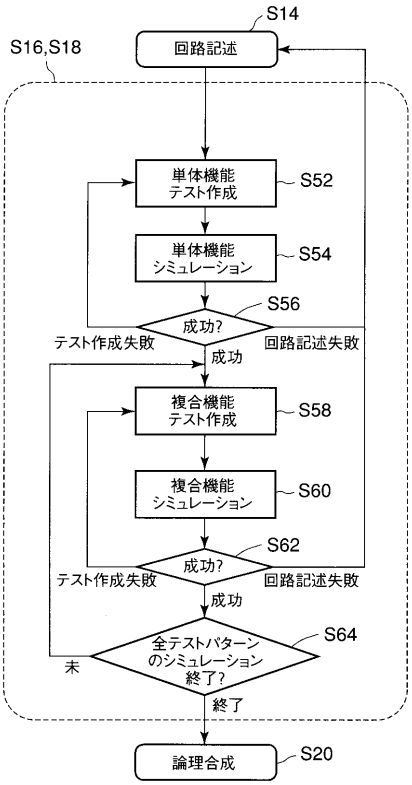
【図３】


24 メモリコントローラ
26
データバッファ
ECCコントローラ 32
66 ECC訂正時に必要なデータ
34
ECC デコーダ
62 バッファへのR/W命令
64 バッファアドレス
68 ECC訂正後のデータ
38
70 メモリからリードしたデータ
40
72 メモリからリードしたECC符号
84 デコードステータス
86 デコード命令
36
42
ECC エンコーダ
74 メモリにライトするデータ
44
28
MUX
76 メモリへライトするECC符号
82 エンコード命令
50
CPU
80 メモリアドレス
78 メモリへのR/W命令
フラッシュメモリ
22
92 プログラムロード
54
94 作業領域として使用
52
ROM
RAM
データバス
制御バス
アドレスバス

【図４】


外部仕様作成
S10
内部仕様作成
S12
回路記述
S14
機能シミュレーション
S16
成功?
S18
失敗
成功
論理合成
S20
論理シミュレーション
S22
タイミング解析
S24
成功?
S26
失敗
成功
配置配線
S28
LSI製造プロセス
S30

【図５】


S14
S16,S18
回路記述
単体機能テスト作成
S52
単体機能シミュレーション
S54
成功?
S56
テスト作成失敗
回路記述失敗
成功
複合機能テスト作成
S58
複合機能シミュレーション
S60
成功?
S62
テスト作成失敗
回路記述失敗
成功
全テストパターンのシミュレーション終了?
S64
未
終了
論理合成
S20

フロントページの続き

(72)発明者　鷹居　頼治
東京都青梅市末広町２丁目９番地　株式会社東芝青梅事業所内

Ｆターム(参考) 5B001 AA03 AD03
5B018 GA01 GA02 HA11 NA06