(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5540155号
(P5540155)
(45)発行日　平成26年7月2日(2014.7.2)　(24)登録日　平成26年5月9日(2014.5.9)

(51)Int.Cl.　　F I
G06F 21/14 (2013.01)　　G06F 21/22 １１４Ａ
G06F 21/57 (2013.01)　　G06F 21/00 １５７Ｃ

請求項の数 21　(全 20 頁)

(21)出願番号　特願2013-520735 (P2013-520735)
(86)(22)出願日　平成23年7月12日 (2011.7.12)
(65)公表番号　特表2013-532864 (P2013-532864A)
(43)公表日　平成25年8月19日 (2013.8.19)
(86)国際出願番号　PCT/US2011/043638
(87)国際公開番号　WO2012/012218
(87)国際公開日　平成24年1月26日 (2012.1.26)
審査請求日　平成25年3月12日 (2013.3.12)
(31)優先権主張番号　12/841,811
(32)優先日　平成22年7月22日 (2010.7.22)
(33)優先権主張国　米国(US)

(73)特許権者　591003943
インテル・コーポレーション
アメリカ合衆国　９５０５４　カリフォルニア州・サンタクララ・ミッション　カレッジ　ブーレバード・２２００
(74)代理人　110000877
龍華国際特許業務法人
(72)発明者　シン、ビン　シー.
アメリカ合衆国　カリフォルニア州・サンタクララ・ミッション　カレッジ　ブーレバード・２２００　インテル・コーポレーション内

最終頁に続く

(54)【発明の名称】プラットフォーム独立メモリ論理の提供

(57)【特許請求の範囲】
【請求項１】
装置であって、
プロセッサ製造業者のプラットフォーム独立コードに対応する半導体統合コード（ＳＩＣ）を格納する不揮発性ストレージを備え、
前記ＳＩＣは、メモリコントローラの初期化によって、メモリを初期化するための埋め込みメモリ論理（ＥＭＬ）を含み、メモリ信号のマッピングは、前記ＥＭＬを介してアクセス可能なオンダイ終端データ構造（ＯＤＴデータ構造）を利用し、前記ＯＤＴデータ構造は、相手先ブランド製造業者（ＯＥＭ）が提供し、前記メモリのプラットフォーム依存メモリ構成のパラメータ化された規則セットに対応しており、
前記ＳＩＣは、プロセッサのリセットに呼応して、前記ＯＥＭの起動前コードに制御を提供する前に実行される、装置。
【請求項２】
前記ＳＩＣをロードする最初のプログラムローダ（ＩＰＬ）をさらに備え、
前記ＩＰＬは、前記プロセッサのマイクロコードから導入され、
前記ＩＰＬは前記ＳＩＣを認証してから前記ＳＩＣを導入する、請求項１に記載の装置。
【請求項３】
前記ＥＭＬは、前記ＯＤＴデータ構造を利用して前記メモリのＯＤＴ抵抗を計算する、請求項１または２に記載の装置。

【請求項４】
前記ＥＭＬは、前記ＯＤＴデータ構造を利用して前記メモリの起動マトリックスを計算する、請求項１から３のいずれか一項に記載の装置。
【請求項５】
前記ＥＭＬは、前記ＯＤＴデータ構造のポピュレーション情報に基づいて前記メモリのチャネルのポピュレーションパターンを計算する、請求項１から４のいずれか一項に記載の装置。
【請求項６】
前記ＥＭＬは、前記ポピュレーションパターンをＯＤＴ規則のリストと比較する、請求項５に記載の装置。
【請求項７】
前記ＥＭＬは、前記ＯＤＴ規則のリストの情報を利用して、公称終端抵抗と書き込み終端抵抗とを計算する、請求項６に記載の装置。
【請求項８】
プロセッサ製造業者が提供する埋め込みメモリコード（ＥＭＣ）を利用してシステムの第１の電力投入中に前記システムのプロセッサに連結されたメモリを初期化することで、前記メモリとメモリコントローラとを初期化して、前記システムの相手先ブランド製造業者（ＯＥＭ）が提供し、プラットフォーム依存情報のパラメータ化された規則セットに対応しており、前記ＥＭＣを介してアクセス可能なオンダイ終端データ構造（ＯＤＴデータ構造）を利用して前記メモリのメモリ構成を生成する段階と、
前記プロセッサの少なくとも１つの段階レジスタに前記メモリ構成のスナップショットを格納する段階と、
ＯＥＭ　ＢＩＯＳ（basic input/output system）に制御を渡す前に、前記システムの不揮発性ストレージに前記スナップショットをセーブする段階と
を備える方法。
【請求項９】
前記プロセッサの低電力モードから退出時に、前記不揮発性ストレージに格納されている前記スナップショットにアクセスする段階をさらに備える、請求項８に記載の方法。
【請求項１０】
前記スナップショットを利用して前記メモリコントローラの構成を復旧した後に、再開ベクトル識別子を利用して前記ＯＥＭ　ＢＩＯＳに制御を渡す段階をさらに備える、請求項８または９に記載の方法。
【請求項１１】
前記ＥＭＣを利用して、前記システムの第２の電力投入中に前記不揮発性ストレージに格納されている前記スナップショットにアクセスする段階をさらに備える、請求項８から１０のいずれか一項に記載の方法。
【請求項１２】
前記メモリの現在のメモリポピュレーションを検知して、格納されている前記スナップショットのメモリ構成と比較して、前記比較に呼応して増分的なトレーニングを施す段階をさらに備える、請求項８から１１のいずれか一項に記載の方法。
【請求項１３】
前記増分的なトレーニングでは、前記メモリのランクに有効なフラグが設定されているかを判断して、判断結果が肯定的である場合に、前記ランクについてのランクトレーニングをバイパスして、前記判断結果が否定的である場合に、前記ランクトレーニングを実行する、請求項１２に記載の方法。
【請求項１４】
システムであって、
複数のコアと不揮発性メモリとを有するプロセッサと、
相手先ブランド製造業者（ＯＥＭ）のファームウェアを格納するフラッシュメモリと、
前記プロセッサに連結されたシステムメモリと

を備え、
前記不揮発性メモリは、前記プロセッサの製造業者のプラットフォーム独立コードに対応する半導体統合コード（ＳＩＣ）を含み、
前記ＳＩＣは、前記プロセッサと、前記プロセッサを前記システムメモリに連結する少なくとも１つのリンクとを初期化する埋め込みプロセッサ論理を含み、
前記システムメモリを、埋め込みメモリ論理（ＥＭＬ）を介してアクセス可能なオンダイ終端データ構造（ＯＤＴデータ構造）を利用して初期化する前記ＥＭＬを含み、
前記ＯＤＴデータ構造は、相手先ブランド製造業者（ＯＥＭ）が提供し、前記システムメモリのプラットフォーム依存メモリ構成のパラメータ化された規則セットに対応しており、
前記ＳＩＣは、複数の第１レベルの段階に分割されて、各段階が少なくとも１つの第２レベルの段階を配下に有し、前記複数の第１レベルの段階それぞれ内では、固有のインデックスが各内部パラメータに割り当てられており、前記プロセッサはさらに、前記ＳＩＣの実行中に現在の第１レベルの段階を格納する少なくとも１つの段階レジスタを含む、システム。
【請求項１５】
前記プロセッサは、前記少なくとも１つの段階レジスタから前記現在の第１レベルの段階を読み出し、前記現在の第１レベルの段階を格納ユニットに格納する、請求項１４に記載のシステム。
【請求項１６】
前記プロセッサは、予め定められた第１レベルの段階が前記少なくとも１つの段階レジスタに書き込まれるときに、前記予め定められた第１レベルの段階での前記ＳＩＣの実行を停止する、請求項１４または１５に記載のシステム。
【請求項１７】
前記プロセッサは、前記予め定められた第１レベルの段階での前記ＳＩＣの実行の内部状態のスナップショットを格納する、請求項１６に記載のシステム。
【請求項１８】
前記ＯＥＭは、前記スナップショットを利用して前記システムをデバッグする、請求項１７に記載のシステム。
【請求項１９】
前記スナップショットは不揮発性ストレージに格納されており、低電力状態から再開するとき前記ＳＩＣによりアクセスされる、請求項１７または１８に記載のシステム。
【請求項２０】
前記ＳＩＣは、ファームウェアインタフェーステーブル（ＦＩＴ）にアクセスして、前記スナップショットの位置を判断する、請求項１７から１９のいずれか一項に記載のシステム。
【請求項２１】
前記ＳＩＣは、前記リンクをトレーニングすることなく、前記スナップショットを利用することで、前記プロセッサのメモリコントローラと前記システムメモリとを初期化する、請求項１７から２０のいずれか一項に記載のシステム。
【発明の詳細な説明】
【背景技術】
【０００１】
従来、コンピュータシステムの起動及び低レベル処理を行うために利用されてきた、低レベルソフトウェアであるＢＩＯＳ（basic input/output system）ソフトウェアは、コンピュータ製造業者である、いわゆるＯＥＭ（相手先ブランド製造業者：original equipment manufacturer）によって組み込まれてきた。プロセッサにＢＩＯＳをサポートさせるために、プロセッサの製造業者（いわゆるシリコン製造業者）は、詳細な情報をＯＥＭに与える。通常、情報はソースコードの形態で組み込まれてきた。しかしこのコード提供により、プログラム方法及び基礎となるハードウェアの詳細などの、プロセッサのハード

ウェアの知的財産（ＩＰ）面の要素が第三者に暴露されていた。さらに、ソースコードを提供すると、ＯＥＭはコードを修正して、非標準的な方法で統合して、非標準的なハードウェアに対応させるべく調整することで複雑性が増し、動作面でも懸念が生じ、シリコン製造業者にとってのサポート料が上がることにつながる。

【０００２】

例えば、通常システムの初期化のために提供される情報には、プロセッサ、インターコネクトリンク、メモリ部等の詳細に関する情報が含まれる。これら詳細には、レジスタの定義、このハードウェアへのインタフェース等が含まれうる。この情報は、ＯＥＭのみならず、独立系ＢＩＯＳ販売業者（ＩＢＶ）にも流通する。プロセッサその他の高度なシリコンコントローラは別の種類のシステム及びデバイスに組み込まれているので、ＩＢＶ及びＯＥＭのｘ８６コンピュータシステムの範囲は比較的小さいが、これから大きく拡張していくことが予想され、ＩＰ面での懸念及びサポート面での困難を生じる可能性がある。

【図面の簡単な説明】

【０００３】

【図１】本発明の一実施形態における、プラットフォーム独立シリコン一体化コード（ＳＩＣ）のコンポーネントのブロック図である。

【図２】本発明の一実施形態におけるシステムのブロック図である。

【図３】本発明の一実施形態における埋め込みメモリ論理（ＥＭＬ）コードの実行を示すフロー図である。

【図４】本発明の一実施形態におけるＥＭＬのさまざまなコンポーネントのブロック図である。

【図５】本発明の一実施形態における終端抵抗を決定する方法のフロー図である。

【図６】本発明の一実施形態におけるプロセッサのブロック図である。

【図７】本発明の一実施形態における、スナップショットを生成して利用する方法のフロー図である。

【図８】本発明の一実施形態における、メモリ初期化に対する増分的なトレーニングを行う方法のフロー図である。

【発明を実施するための形態】

【０００４】

さまざまな実施形態で、最初の初期化ソフトウェアは、基盤となるプロセッサハードウェアに対して密接に取り付けることができるので、ＯＥＭ　ＢＩＯＳにより実装されるのではなく、シリコン製造業者が最初の初期化ソフトウェアを制御して提供することができる。

【０００５】

シリコン一体化コード（ＳＩＣ）は、コードがロードされ、出所を確かめて、ＳＩＣの実行後に、滞りなくＯＥＭ　ＢＩＯＳに制御を渡される環境で利用することができる。ＳＩＣは、アナログリンクの設定がなされるトレーニング、診断、及びテストフックの設計といったメモリ初期化の低レベル面の動作に利用することができる。本発明の範囲はこの点に限定はされないが、メモリコントローラ及びインターコネクトリンクの主要な初期化コード、及び、さまざまなプロセッサ及びシステムの特徴部のランタイムを潜在的に助ける部材を、ＳＩＣの形態とすることができる。一部の実施形態では、このコードに対する信頼ルートが、プロセッサのマイクロコードのフック経由で提供されて、ＳＩＣを認証してロードするためのＳＩＣロードを実装することができ、これは、シリコン製造業者のＢＩＯＳの形態であってよく、ＯＥＭ　ＢＩＯＳに制御を渡すことができる。ＯＥＭ　ＢＩＯＳへのアップグレードが難しいことを鑑みると、フラッシュメモリに組み込まれたファームウェア等のＢＩＯＳを格納するメモリが適合しない場合もあるので、ＳＩＣローダを導入するためのＳＩＣローダプロセッサパッケージ内のコアのマイクロコードに対応する信頼ルートが、ＳＩＣが組み込まれたＯＥＭ　ＢＩＯＳの補助ブロックが有効であることを検証するために利用される。

【０００６】

ＳＩＣの実行後に、制御は、ＵＥＦＩオープンソースコア（例えば２００６年2月２１日付けのＵＥＦＩ仕様バージョン２．０）に従って、または、オープンファームウェアＩＥＥＥ１２７５－１９９４、または、従来のＢＩＯＳまたは特許化されているＯＥＭブートコード（それぞれ１以上のオペーティングシステムを起動する）を介して実装することができるＯＥＭ　ＢＩＯＳに渡されてよい。実装例によっては、ＳＩＣは、例えばエラー解消のため、またはバイナリ変換を行うために、パッチを実行するために利用することができる。さらに、ＳＩＣは、バイナリ変換、エラー処理のためのメモリブロック、またはランタイム中に実行されるシリコン製造業者の他のコードを隠してＯＳから見えなくするために、一定のメモリブロックに対する保護を設定することができる。このようにすることで、プロセッサが、例えばマイクロコード読み出し専用メモリ（ｕＲＯＭ）オフロード、コプロセッサのサポート等の逆コードを利用する場合には、機械を第三者であるＯＥＭのコードに晒される前に、ＳＩＣがこのメモリを提供することで、メモリのサブセットをプロセッサに利用可能としておくことができるようになる。

【０００７】

さまざまな実施形態では、ＳＩＣは、フラッシュその他の不揮発性メモリ等であるＯＥＭ　ＢＩＯＳに組み込まれるバイナリコードとして提供されてよく、または、ＳＩＣをプロセッサパッケージの不揮発性ストレージに実装してもよい。従って実施形態では、プロセッサその他のシリコン製造業者が顧客に対して、ソースコードではなくて、バイナリモジュールを出荷することができるので、より高いＩＰ保護が可能となる。顧客数が増えると、シリコンとともにＢＩＯＳ構築ブロックを出荷することによって実現される技術サポート料の削減量が顕著となる。

【０００８】

図１は、ＳＩＣに組み込むことができる、本発明の一実施形態のプラットフォーム独立ＢＩＯＳのコンポーネントのブロック図である。プラットフォーム独立という用語は、機械の仕様に関わらず（例えばメモリの種類、リンクの速度、利用可能な電圧等）、どのプラットフォームでも、含まれているコンポーネントが実行可能であることを示す。後述するように、プラットフォーム依存であるコンポーネントは、プラットフォームの製造業者のデータファイルにより符号化することができる。一例としては、プラットフォーム販売業者は、システム基板等に関する情報等の一定の情報をパラメータ化することができる。基礎的な初期化を行う一定のコンポーネント（例えば信頼ルートを構築するリンク初期化）は、第三者のコードの制御を処理する前にＳＩＣによって行われてよい。このようにＳＩＣを一時的に隔離することで、電力をリセット中にもプロセッサがハードウェアを有して、制御を第三者のコードに渡すまではＳＩＣだけが実行可能であることを確認することができる。

【０００９】

図１からわかるように、ＳＩＣコンポーネントは、第三者のコードの前にＳＩＣの残りを発見してロードする役割を担う最初のプログラムローダ（ＩＰＬ）２０を含んでよい。このローダは、ＳＩＣのコンポーネントとは考えられない場合もある。ＳＩＣのコンポーネントが一時的であったり持続的であったりする。持続的な場合には、アクティブにされたり（プラットフォーム初期化（ＰＩ）モジュール）またはパッシブにされたり（新たな命令によってライブラリのように機能すること）してよい。

【００１０】

図１の実施形態では、ＳＩＣ１０が、埋め込み中央処理装置（ＣＰＵ）論理（ＥＣＬ）３０を含み、これは、オンボードのインターコネクト等のインタープロセッサ論理を初期化する。リンク及びメモリ初期化の両方のトレーニングには、リンクの両端の間の通信が含まれてよい。リンクの場合には、プロセッサは、別のプロセッサまたは出入力（ＩＯ）ハブに連結されてよく、データ処理可能な動作周波数及びリンク幅を決定することができる。ＥＣＬモジュールは通常は一時的である。一部の実施形態では、リンクの初期化は、低電力状態（例えばＡＣＰＩ（Advanced Configuration and Power Interface）低電力状態、その一例が２００９年6月１６日付けのＡＣＰＩ仕様バージョン４．０のＳ３である

）からの再開処理中に、メモリコンテンツにダメージを与えずにやり直すことができる。
【００１１】
ＳＩＣ１０はさらに、メモリの初期化に際して埋め込みメモリ論理（ＥＭＬ）４０を含んでよい。このコンポーネントは、ランタイム中に（例えば温度のスロットルまたはエラー処理のため）一部が一時的であり一部が持続的であってもよい。別のコンポーネントではメモリを別の視点から見るために、メモリ初期化には、アドレス指定の設定がさらに含まれていてもよい。例えば、ＣＰＵは、システムアドレスと称される物理的アドレス空間の視点を有することができる。システムメモリ（例えばダイナミックランダムアクセスメモリ（ＤＲＡＭ））に連結するメモリコントローラは、チャネル、ランク、及びランクを有するオフセットにより特定される特定のメモリセルにアドレスをマッピングする。メモリアクセスは全て、プロセッサノードにルーティングされて、復号され、チャネル番号、リンク番号、及びランクにおけるオフセットに変換される。従って、リンクアドレスの符号器のプログラミングの仕方によって、メモリは異なる見方で見ることができる。プロセッサについて隔離されたメモリ量を確保するために、アドレス復号器及びメモリ復号器のサブセットをプロセッサについてプログラミングすることができる。このようにして隔離されたメモリは、システム動作中にはプロセッサが独占して利用することができ、マイクロコードオフロード、バイナリ変換器、ライセンス情報、管理情報等を格納することができる。これに加えて、予備論理５０を利用して、プラットフォーム独立論理の他の特徴部をサポートすることができる。本発明の範囲は、図１の実施形態に示すこの点に限定はされない。
【００１２】
図２は、本発明の一実施形態におけるシステムのブロック図である。図２に示すシステム１００は、ハードウェアと、さまざまなソフトウェア層とを含んでいる。示されているハードウェア実装は、代表的な高位のレベルであり、システムによっては数多くの他のコンポーネントが存在しうる。さらに、ソフトウェアの各部分が様々な場所に分散された様々なコンポーネントとして示されているが、特に、様々な実施形態でＳＩＣの各部分が様々な場所に存在してよいことを理解されたい。
【００１３】
システムの基礎部分であるハードウェア１１０は、１以上のプロセッサ、メモリ、入出力デバイス等を含む様々なコンポーネントを含んでよい。しかし、図示の便宜上、図２の実施形態にはこれらのコンポーネントの一部のみを示している。特に、コアで実行するものとして、プロセッサコア１１５及び対応するマイクロコード１２０が示されている。様々な実施形態では、コア１１５は、それぞれが自身のマイクロコードストアを含んでよいマルチコアプロセッサの複数のコアを表すことができる。加えて、多くの実装例では、プロセッサパッケージがアンコア１１８（プロセッサコアの外に存在する様々な論理を含む）を含むことができる。様々な実施形態では、アンコア１１８は、コア１１５、統合メモリコントローラ（ＩＭＣ）１２５、及び入出力コントローラハブ（ＩＯＨ）１２８に、Ｉｎｔｅｌ（登録商標）Ｑｕｉｃｋ　Ｐａｔｈ　Ｉｎｔｅｒｃｏｎｎｅｃｔ（ＱＰＩ）リンクを介して連結される。これを受けてＩＭＣ１２５は、システムメモリ１３０に連結されてよく、これは一実施形態では、ダブルデータレート（ＤＤＲ）－３ＤＩＭＭ等の複数のデュアル・インライン・メモリモジュール（ＤＩＭＭ）から構成されてよい。さらに図示されているように、ＩＯＨ１２８は、周辺コントローラハブ（ＰＣＨ）１３５に直接媒体インタフェース（ＤＭＩ）インターコネクトにより連結されてよく、これが、トラステッド・プラットフォームモジュール（ＴＰＭ）１４０及び周辺モジュール１４５等の他のデバイスに連結されてよい。本発明の範囲はこれに限定されないが、様々なデバイスのＰＣＨ１３５へのインターコネクトは、低ピンカウント（ＬＰＣ）バスによって行われてよい。これら限定されたコンポーネントのみが示されているが、多くの他のコンポーネントも別の実装例では存在する可能性がある。
【００１４】
プロセッサハードウェアにはＳＩＣモジュール１５０が密接に連結されている。様々な

実装例では、ＳＩＣモジュールは、フラッシュメモリ等の不揮発性ストレージに実装されており、例えばプロセッサパッケージ内に置かれてよい。この代わりに、ＳＩＣモジュールは、プロセッサの製造業者からＯＥＭにバイナリで提供されてよく、ＯＥＭは自身のＢＩＯＳにこれを組み込む。従って、ＳＩＣの実際の位置は、実装が変わると変わる場合がある。一実施形態では、ＳＩＣは、ＯＥＭ　ＢＩＯＳの一部である（例えばマザーボードに取り付けられたフラッシュデバイスに統合されている）。しかし、図示を簡単にすべく、ＳＩＣモジュール１５０が別個のコードブロックとして示されている。特に、示されているモジュールは、複数のＳＩＣコードの改定152_1－152_nを含んでいる。つまり、後で詳述するように、ＳＩＣはフィールドで更新されてよい。ＳＩＣモジュールには、コードに加えて、様々なデータブロック154_1から154_nが存在していてよい。一実施形態では、シリコン製造業者のポリシーが、このＳＩＣデータブロックに含まれていてよく、これがＳＫＵ（stock keeping unit）に基づいていてよい。この情報は、部分のグレーディング、サポートされた特徴、及び／または、シリコンの熱物理特性を示すような電圧及び温度のレーティングを含んでよい。これら様々なコード及びデータブロックにアクセスするために、ファームウェアインタフェーステーブル（ＦＩＴ）１５５を提供することができる。符号及びデータブロックはキーとなる対の構造により保護されていてよい。様々な実施形態では、ＳＩＣ１５０がハードウェアに暗号的に結合されており、コアマイクロコード１２０を媒体として起動がなされる。

【００１５】

さらなるシステムソフトウェアが、ＵＥＦＩ規格に準拠した起動前コードを含むＯＥＭ　ＢＩＯＳの形態であってよい。これらのコードは、様々なファームウェア部分で格納されてよく、セキュリティ初期化コード（ＳＥＣ）、前ＥＦＩ（ＰＥＩ）コード、及び、ドライバ実行環境（ＤＸＥ）及びブートデバイス選択（ＢＤＳ）コードを含んでよい。論理上の例としては、これらのコードは、第１のコードモジュール１６０と第２のコードモジュール１７０とに分けられてよい。ファームウェア部分は、ＢＩＯＳその他の低レベルソフトウェアを含んでよい不揮発性メモリの論理上の分割部分である。一般的には、モジュール１６０及び１７０がそれぞれ、第三者の拡張可能コードを有するファームウェア部分であってよい。第１のコードモジュール１６０は、ＯＥＭプラットフォームの初期化を行い、ＳＩＣコードからメモリアパーチャを取得するために利用することができる（ブロック１６２）。制御は次にコードブロック１６４に渡されて、物理的実態についての分析を実行して、セキュアなハッシュ演算を行うことができ、これにより、主要なファームウェア部１６８の測定を行う前に、セキュアなファームウェア更新メカニズム１６６を実行することができるようになる。これらの認証チェック全てにパスすると、制御は第２のコードモジュール１７０に渡され、これらがＤＸＥコアを含んでよく、ＵＥＦＩセキュアな起動モジュール１７４を実行してよい（例えばＡＣＰＩメモリに格納されている測定ログ１７６を利用して）。次に、ＯＳ１８０をロードするＯＳローダ及びドライバ１８５が実行されてよく、ＯＳ１８０はＵＥＦＩに準拠したＯＳであってよい。他の実装例では、レガシーのＯＳ１９０が、１以上の任意で設けられるＲＯＭ１９５を介して実装されてよい。

【００１６】

上述したように、本発明の一実施形態のＳＩＣは、埋め込まれたメモリ論理を含んでよい。この論理は通常、メモリ参照コード（ＭＲＣ）のバイナリ形態をカプセル化したものであってよく、メモリ参照コード（ＭＲＣ）は、プロセッサの製造業者がＯＥＭにメモリ等の初期化方法を知らせる目的で、ソースコードの形態でＯＥＭに提供してよいＢＩＯＳモジュールのことである。しかし上の説明によれば、基礎となるハードウェアＩＰをＯＥＭに見えない形で提供するために、埋め込みメモリ論理はバイナリ形式で提供されてよい。

【００１７】

ＥＭＬはプラットフォーム独立なために、プラットフォームに依存した問題を特定して提供する様々な情報が、ＥＭＬによりアクセス可能となる。一例としてはシステム基板のばらつきが挙げられる。例えば、プラットフォームごとに配線が異なるので、シリアル存

在検知（ＳＰＤ）／システム管理バス（ＳＭＢＵＳ）マッピング、信号マッピング、タイミング、オンダイ終端（ＯＤＴ）抵抗等が異なってくる。またさらに、さまざまに異なるＯＥＭは、カスタマイズ／デバッグに対して異なる要件を有する場合があり、特に、ＯＥＭが対応するソースコードを有さないことから、エラーの記録／報告が問題となる。後述するように、一部の実施形態では、ハードウェアを提供してデバッグ処理を助けている。プラットフォーム間でのばらつきの他の例には、販売業者が非標準的なＤＩＭＭまたは逸脱した部分を選択しているために、これらの限定を理解するために周辺作業が発生するような場合も含まれる。実施形態では、ＥＭＬをクリアなインタフェースでコンポーネントの階層に組み込むことで、ＯＥＭにプラットフォームをパラメータ化させ、ＥＭＬのバイナリ提供を可能としてよい。

【００１８】

概して、ＥＭＬは、ＯＥＭが供給するプラットフォーム構成データをパースすることでメモリを初期化してよい。以下の説明はＤＤＲ３タイプのメモリで実装されたシステムメモリに関するものであるが、本発明の範囲はこの点に限定はされない。

【００１９】

図３は、本発明の一実施形態におけるＥＭＬコードの実行を示すフロー図である。様々な実施形態では、システムをリセットすると、または、低電力状態からの再開時に、ＥＭＬに入ることができる（例えばＳ３状態）。図３からわかるように、方法２００は、様々なレジスタを初期化することで開始されてよい（ブロック２１０）。様々な実施形態では、これらレジスタ及びバスは、ＰＣＩ　Ｅｘｐｒｅｓｓ（登録商標）ベースのアドレスレジスタ、システム管理バス、汎用入出力（ＩＯ）レジスタ、及び、電力管理（ＰＭ）レジスタを含んでよい。一実施形態では、Ｉ／Ｏテーブル（例えばＩＯ／メモリマッピングされたＩＯ（ＭＭＩＯ）ポート及びプログラムへの値のリスト）を利用して、このステップを実装することができ、この場合には、ＩＯテーブルが、プラットフォーム製造業者により供給されＥＭＬにより実行されてよい。一定のデバイスがコードにより利用され、ＥＭＬの実行前の初期化が必要となることから、様々な初期化が生じうる。次に、処理がシステムへの電力投入または低電力状態（例えばＳ３）からの再開によって生じているかを決定してよい（ダイアモンド２１５）。最初の電力投入時には、制御はブロック２２０に渡されて、ここで取り付けられているシステムメモリが検知されてよい。一実施形態では、検知が、ＳＭバスを介したＳＰＤの読み取りによって生じてよい。ＤＩＭＭを見つけることのできるアドレスは、プラットフォームに依存している。例えば、数多くのＤＩＭＭを有するより大きなプラットフォームは、ＳＭバスに、様々なセグメント間を切り替えるためのマルチプレクサを実装してよい。

【００２０】

図３では、次に、メモリクロック比を決定してよい（ブロック２４０）。プラットフォームが特定のクロックレートをサポートしていなくてもよいために、この処理はプラット依存している可能性がある。つまり、クロック比は、メモリコントローラ／プラットフォームの設計のポピュレーション及びユーザの設定の少なくとも１つによって上限が設けられる場合がある。クロック比を設定した後に、制御はブロック２５０に進み、ここではメモリコントローラのグローバル初期化が行われてよい。このグローバル初期化によって、ＩＭＣがデフォルト状態へとリセットされる（例えば、ＩＭＣ側でのＯＤＴ提供の設定、及び、全ての信号について許可された最大ジッタ（参照電圧とサンプルされた電圧との間の差の最大値等））。様々な実施形態では、メモリコントローラは、図２を参照して上述した統合メモリコントローラであってよい。次に制御がブロック２６０に渡され、メモリ信号をメモリコントローラ信号にマッピングしてよい。ここでもまた、プラットフォーム依存信号ルーティングを行ってよい。例えばメモリコントローラは、最大８チップ選択（ＣＳ）信号を有してよく、ＤＩＭＭは、最大４つの信号（４個のランクのＤＩＭＭ）まで利用することができる。従って、メモリコントローラは、コマンドをあるランクに送る際に、チップ設定ピンのいずれをプルダウンするかを決定するよう構成されていてよい。例えば、ＤＩＭＭ１のＣＳ０は、プラットフォームの配線に応じてＩＭＣのＣＳ２またはＣ

Ｓ４に接続されてよい。この情報は、プラットフォーム製造業者により供給される。
【００２１】
次に制御がブロック２７０に渡されて、スタティックな構成を実行することができる。スタティックな構成とは、ＳＰＤから導出可能な全ての設定のことを示していてよい（例えば、トレーニングを用いて決定される「ダイナミック設定」に対する２ＤＤＲコマンドの間で必要となる遅延）。プラットフォーム依存の構成の一例としては、ブロック２７０で決定されるＯＤＴ抵抗がプラットフォーム依存型であってよい。他のスタティック構成処理には、ＯＤＴ起動及び抵抗選択に加えて、内部の論理対物理ランクマッピングが含まれてよい。これら選択と起動の詳細について説明する。ブロック２８０で、ここでもプラットフォーム依存であってよいトレーニングを実施して、一部のプラットフォームでは、トレーニングされていない一部の遅延に、デフォルト値とは異なる値を割り当ててよい。トレーニングを実行した後で、ブロック２９０でテストを実行してよい。そしてメモリマップが完了してよい（ブロック２９５）。メモリマップは、「チャネル／ランク／オフセット」のインターリーブへの「システムアドレス」への参照である点に留意されたい。つまり、ＩＭＣは、複数の「インターリーブ」モードをサポートすることができる。例えば、性能を上げるために、２つの連続した「システムアドレス」を２つの異なるチャネルにマッピングすることができ、同じチャネルを有する２つの連続したアドレスが、さらに２つの異なるランクにインターリーブされて、熱を分散することができる。
【００２２】
最初の電力投入ではなくて低電力状態からの起動によってＥＭＬに入る場合には、制御はダイアモンド２１５からブロック２３０に渡されて、これはＳ３の再開経路であり、前に不揮発性ストレージに格納されていた構成情報を取得することができるが、これに関しては後で詳述する。つまりこの経路においては、情報は不揮発性メモリからテストを行わずに取得される。というのもテストを行うとメモリの内容が損なわれるからである。次に制御はブロック２９５で直接前に生成されたメモリマップに渡される。図３の実施形態にはこの特定の実装例を示したが、本発明の範囲がこの点に限定されないことは理解されたい。
【００２３】
図４は、本発明の一実施形態におけるＥＭＬのさまざまなコンポーネントのブロック図である。図４に示すように、ＥＭＬ３００は、汎用メモリ論理３１０、プロセッサ論理３２５、及びプラットフォーム論理３３０を含む様々な論理を含みうる。概して、メモリ論理３１０は、メモリ自身を初期化してイネーブルするための様々なコード（例えば、ランク論理３１２、ＤＩＭＭ論理３１４、及びチャネル論理３１５）を含むことができる。これら論理は、ランクのための信号及びＭＲＳ論理、ＤＩＭＭ及びＤＩＭＭ／ランクポピュレーションのためのＳＰＤ及び信号論理、ＤＤＲ３タイミング、ＯＤＴ構成、及びチャネルのためのトレーニング及びテストシーケンスを含んでよい。またさらに、コードは、メモリコントローラの初期化、ＤＩＭＭ検知及びグローバルタイミングを実行するためのメモリ制御論理３２０を含んでよい。プロセッサ論理３２５は、ドライバとして実装されて、コマンド及びステータスのレジスタ操作、内部タイミング及びコマンド／トレーニングを実行してよい。プラットフォーム論理３３０は、トポロジー／配線、タイミング／遅延、ＯＤＴ規則及び様々なデバイスを介してプラットフォームと相互作用してよい。別の実装例では、図４に示すコードが異なる量で、プロセッサ製造業者が提供するバイナリで実装されてもよい点に留意されたい。例えば汎用メモリ論理３１０及びプロセッサドライバ３２５がバイナリで実装されてよい。また別の実施形態では、プロセッサ論理３２５がバイナリとして実装されてよく、残りのコードがＯＥＭ　ＢＩＯＳで実装されてよい。
【００２４】
上述したようにＥＭＬはプラットフォーム独立型である。このコードをバイナリ形式で配信するために、プラットフォーム固有のプラットフォームの様々な局面（例えばプラットフォーム依存したもの）を、ＥＭＬにアクセス可能な構成テーブルのデータとして表してよい。図３を参照して上述したように、メモリ初期化の一部は、抵抗と起動とを含むＯ

ＤＴ構成である。抵抗とは、ＤＩＭＭが、信号を終端するために利用する抵抗のことであり、起動は、読み書きのコマンドをメモリチャネルのいずれかのランクに送信するときにメモリコントローラ（ＭＣ）が制御するＯＤＴ信号のことである。ＯＤＴ構成は、ＤＩＭＭポピュレーションとＤＩＭＭスロット書き込み両方に依存している。従来のＢＩＯＳでは、これらの構成を、それぞれ異なるプラットフォームについて書き換えられるプラットフォーム依存関数で計算していた。これに対して、様々な実施形態では、初期化処理に関する全てのプラットフォームのカスタム化を、ＯＥＭコードでではなく、データで実装することができる。このようにしてプロセッサ製造業者（及びＥＭＬプロバイダ）は、ＳＩＣのセキュリティを保証することができる。

【００２５】

様々な実施形態では、ＯＤＴ構成は、例外を規定する機能を有し、一般的且つコンパクトな形態でパラメータ化することができる。特にここで「ＯＤＴ規則」と称されるデータ構造が、プラットフォーム用のＯＤＴ構成を生成するために利用されてよい。このデータ構造あるいはＯＤＴ規則は、プラットフォームの製造業者によって、例えばＰＭＤの一部として供給されてよい。別の実装例では、このＯＤＴ規則が一般的であっても固有であってもよい。ここで、あるプラットフォームが一般的及び固有の規則を両方とも有してもよい点にも留意されたい。ＥＭＬは、現れた順序で整合する規則を探すので、ほとんどの場合には固有の規則が一般的な規則の前に来る。一般的なＯＤＴ規則は、複数のポピュレーションパターンに整合していてよく、ＯＤＴ抵抗及び起動マトリックスを一般的なアルゴリズムによって計算することができる。固有のＯＤＴ規則は１つのポピュレーションパターンに整合している。固有のＯＤＴ規則では、ＯＤＴ抵抗がこの規則で所与となっており、起動マトリックスは一般的なアルゴリズムによって生成することができる。いずれの場合であっても、起動マトリックスは、ＯＤＴ規則に追加されたフラグにより無効にされてよい。

【００２６】

一実施形態では、様々な情報が１つのＯＤＴ規則内に含まれていてよい。一般的な規則及び固有の規則両方について、ビットマスクとして実装可能な、規則を適用するメモリチャネルセット、ランクの公称抵抗値Ｒｔｔ＿Ｎｏｍ、及び、ランクの書き込み抵抗値Ｒｔｔ＿Ｗｒ、及び、起動マトリックスを無効にするフラグが含まれてよい。ＪＥＤＥＣ規格では、そのＯＤＴピンがメモリコントローラによってアサートされると、ＤＲＡＭデバイスがＯＤＴを起動する。現在のコマンドが書き込みコマンドではない限り、Ｒｔｔ＿Ｎｏｍが利用され、現在のコマンドが書き込みコマンドである場合には、Ｒｔｔ＿Ｗｒが代わりに利用される。一実施形態では、これらのフラグは自身の読み出しのオン、ピアの読み出しのオン、自身の書き込みのオン、および、ピアの書き込みのオンを含んでよい。

【００２７】

一般的な規則では、ＯＤＴ規則はさらに、規則を適用するＤＩＭＭセット（例えばＤＩＭＭのビットマスクまたは最大インデックス）をチャネル内に含み、ＤＩＭＭの公称抵抗値及び書き込み抵抗値Ｒｔｔ＿Ｎｏｍ及びＲｔｔ＿Ｗｒを含んでよい。この抵抗は、同じＤＩＭＭ内でランクごとに分割されてもよい点に留意されたい。

【００２８】

固有の規則については、ＯＤＴ規則が、特定のポピュレーションに整合させるためのポピュレーションパターン、ランクについてのＲｔｔ＿Ｎｏｍ及びＲｔｔ＿Ｗｒに対する抵抗をさらに含んでよい。Ｒｔｔ＿Ｗｒ及びＲｔｔ＿Ｎｏｍはランクごとに特定することができる。

【００２９】

ＯＤＴ規則を整合させるために、メモリの各チャネルについてポピュレーションパターンを計算してよい。一般的には、Ｎビットを利用して、論理ランクをＮ個までサポートするチャネルを記述してよい。あるチャネルに対するポピュレーションパターンの計算は、一実施形態では以下のように行うことができる。

【００３０】

マスク＝０

【数２】

$$\forall d\varepsilon\{DIMM\}: mask \leftarrow mask|(1<<(DIMM[d].\#\_of\_ranks\_DIMM[d].rank\_rank\_offset-1))$$

【００３１】

従って、この計算によって、ｄランクのチャネルのポピュレーションパターンを一意に特定するマスクが提供される。

【００３２】

ポピュレーション及び様々な１つのランク（ＳＲ）についての符号、デュアルランク（ＤＲ）及び４個のランク（ＱＲ）メモリ構成の例は以下のようになる。 SR - 00000001b

DR/DR - 00100010b

DR/QR - 00101000b

DR/DR/DR/DR -10101010b

【００３３】

計算されたパターンは、ＯＤＴ規則のリストとの比較に利用することができる。これは、最初の規則のみが有効なケースである、複数の規則に整合させるポピュレーションについて可能である点に留意されたい。一般的にはＯＥＭでは、データ構造内の一般的な規則の前に固有の規則を配置することで、あるパターンが先ずはその固有の規則に整合するようにしている。

【００３４】

一実施形態では、起動マトリックスが以下の規則に従って生成されてよい。

【数１】

1. ACT[R]←)
2. $\forall r\varepsilon\{Rank\}r \neq VDD: ACT[R] \leftarrow ACT[R]|(1<<Rank[r].ODT)$
3. ACT[R]
4. ACT[W] | (1<<Rank[R].ODT)

ＡＣＴ[Ｒ]は、ＶＤＤに接続されていない場合の読み出しコマンドのＯＤＴマスクであり、ＡＣＴ［Ｗ］は、書き込みコマンドのＯＤＴマスクであり、これ以外の場合には書き込みマスクは読み出しマスクと同じである。従って２つのマトリックスが存在する可能性があり、このうち１つが読み出しコマンドのものであり他方が書き込みコマンドのものである。マトリックスは論理ランク番号によりインデックスされるアレイである。マトリックスの全てのエントリは、ＩＭＣがあるチャネルについて有しているＯＤＴピン数と同じ数のビットのビットマスクである。ＩＭＣは、読み出しコマンドの場合には、ＡＣＴ[Ｒ][ｔａｒｇｅｔｅｄ＿ｒａｎｋ＿＃]に従ってＯＤＴ制御ピンをアサートして、書き込みコマンドの場合には、ＡＣＴ［Ｗ］[ｔａｒｇｅｔｅｄ＿ｒａｎｋ＿＃]をアサートする。ほとんどの場合、ACT[W][rank_#] = ACT[R][rank_#] | (1 << rank_#)である。従ってアルゴリズムは、ＯＤＴ規則により無効化されない場合に、デフォルトの振る舞いとしてこれを記述する。ＶＤＤは例外であり、この場合にはＩＭＣが対象とされているランクのＯＤＴ制御ピンを制御しない。計算後であってＭＣ構成に適用する前に、整合しているＯＤＴ規則に追加されたフラグに基づいて起動マトリックスを改定することができる。公称終端抵抗及び書き込み終端抵抗は、以下の仮定に基づいていてよい。つまり、普通の処理についてのＤＩＭＭの全終端抵抗は、そのＤＩＭＭにあるランク数に関わらず同じである。また、１つのチャネル内の全てのＤＩＭＭについて同じ抵抗を利用することができる。

【００３５】

ほとんどの場合、これら仮定が成り立っている。成り立っていない場合には、固有の規則を利用してデフォルトの規則を無効にすることができる。一般的な規則については、Ｒ

ｔｔ＿ＮｏｍをＤＩＭＭレベルで特定する。つまり、１を超える数のランクが存在している場合には、Ｒｔｔ＿Ｎｏｍを、ＭＣに接続されているＯＤＴ制御信号を有する全てのランク間で分割してよい。ＤＤＲ３仕様では、ＤＩＭＭ１つについて最大で２つのＯＤＴ信号しかないので、以下の３つの場合が想定される。１つ目の場合は、ＤＩＭＭにＯＤＴ制御信号が接続されていない場合、Ｒｔｔ＿Ｎｏｍをディセーブルする。もしＯＤＴが１つだけ接続されている場合には、Ｒｔｔ＿Ｎｏｍを、接続されているランクのＲｔｔ＿Ｎｏｍとして利用する。２つのＯＤＴ制御信号が接続されている場合には、Ｒｔｔ＿Ｎｏｍ＊２がＤＤＲ３仕様の符号を有さない限りにおいて、Ｒｔｔ＿Ｎｏｍ値を接続されているランク両方に利用する（Ｒｔｔ＿Ｎｏｍ＊２がＤＤＲ３仕様の符号を有する場合には、第２のＯＤＴが切断されたものとして、第１のランクはＲｔｔ＿Ｎｏｍを直接利用する）。固有のＯＤＴ規則については、Ｒｔｔ＿Ｎｏｍ及びＲｔｔ＿Ｗｒをこのランクレベルで特定して直接利用する。

【００３６】

図５は、本発明の一実施形態における終端抵抗を決定する方法のフロー図である。図５に示すように、方法４００は、あるポピュレーションパターンについて整合するＯＤＴ規則を発見することから始められてよい（ブロック４１０）。上述したように、ポピュレーションパターンは、あるチャネルについて計算されてよく、次にこのパターンを利用してＯＤＴ規則のリストとの比較を行うことができる。次に制御はダイアモンド４１５に渡され、あるＤＩＭＭについて全てのランクについて第１のループが行われたかを判断してよい。判断結果が否定的である場合には、制御はブロック４２０に行き、公称終端抵抗を所与のランクのディセーブルした値に対して設定して、書き込み終端抵抗をＯＤＴ規則からの書き込み終端抵抗値に対して設定することができる。このことからわかるように、このループは、ランク内の全てのループについて設定が作成されるまで行われてよい。

【００３７】

そして制御がブロック４２５に渡されて、ＯＤＴ制御信号に接続されるランク数が決定されてよい。ダイアモンド４３０で、制御信号の数は０、１、または２と決定されてよい。０である場合には、上述したように、公称終端抵抗をディセーブルして、制御を次のＤＩＭＭに渡す（ブロック４８０）。単一の制御信号のみが接続されている場合には、制御はブロック４７０に渡されて、ここで任意のランクについて公称終端抵抗を、ＯＤＴ規則から得た終端抵抗に設定してよい。次に、上述したようにブロック４８０に制御を渡す。

【００３８】

代わりに２つの制御信号が接続されている場合には、制御はダイアモンド４３５に渡されて、公称終端抵抗の二倍の値が、あるメモリ仕様の符号（例えばＤＤＲ３仕様）を有しているかを判断してよい。判断結果が否定的である場合には、制御をブロック４７０に渡して、第１のランクが、ＯＤＴ規則からの公称終端抵抗を直接利用してよい。判断結果が肯定的である場合には、制御がブロック４４０に渡されて、ここで第１のランクをＯＤＴ規則の抵抗の二倍の公称終端抵抗に設定することができる。そして制御をダイアモンド４４５に渡して、この第１のランク以外の全てのランクをループする。ダイアモンド４５０で、所与のランクが、供給電圧（ＶＤＤ等）に接続されているＯＤＴ制御ピンを有するか（つまり全時点でＯＤＴが存在しているか）を判断する。判断結果が肯定的である場合には制御はダイアモンド４４５に戻る。判断結果が否定的である場合には、制御はブロック４６０に渡され、このランクを、ＯＤＴ規則の値の二倍の公称終端抵抗に設定する。本発明の範囲は、図５の実施形態に示すこの点に限定はされない。この方法を利用すると、終端抵抗を計算することができ、表１に示すように、ＯＤＴ起動マトリックスをＯＤＴ規則の情報から計算することができるので、様々なメモリ構成（ＯＤＴ構成を含む）をコンパクトな形態でパラメータ化することができ、ＥＭＬのバイナリでの配信を可能とすることができる。

【００３９】

従って、本発明の一実施形態を利用することで、ＯＤＴ構成をコンパクトな形態にパラメータ化することができ、本発明の一実施形態に従ってＥＭＬのバイナリでの分配を可能

とすることができる。従って、ＯＥＭは、純粋なデータ（ＯＤＴ規則のデータ構造）を利用してメモリチャネルにメモリ初期化を実行するための情報を提供することができる。

【００４０】

記載してきたように、ほとんどのプラットフォームの初期化はＳＩＣバイナリを利用して、ＯＥＭがソースコードにアクセスしないようにして行うことができる。しかし、ＳＩＣが失敗した場合、特に、新たなプラットフォームの開発中には、ＯＥＭがこれらコードの内部状態をチェックしたい場合もでてくるので、いままでどおりコードの内部状態を、ある程度第三者に開示する必要があるだろう。さらに時折第三者はある段階（例えば１つのステップ）でＳＩＣ実行を停止して、配線が送信する波形をとるためにオシロスコープを接続すること等によって、ハードウェアを診断したいと望む場合があってもおかしくない。一部の電力状態の遷移においては（例えばＳ３からの再開時）、一部の初期化を省略して、Ｓ３に入る前にセーブした情報を利用してシリコンを再構成することができる。

【００４１】

実施形態では、ＳＩＣの内部状態を露呈させて、第三者の処理に適合させるための画一化された方法を提供することができる。こうするために、ＳＩＣは、再上位レベル段階に分割及び定義して、各再上位レベルの段階において１以上の副段階があるようにする。段階の定義によって、ＳＩＣコンポーネントを実行することで行われている処理を特定することができる。１以上の段階レジスタを提供して、読み出されたときのＳＩＣの現在の段階を報告して、書き込まれるときには固有の段階番号でＳＩＣ実行を停止させることができる。そして各段階において、ＳＩＣ実行の全ての内部パラメータに固有のインデックスを割り当ててよい。コマンドインタフェース（例えばＣＳＲ対）を利用して、インデックスを付されたパラメータの読み出し／書き込みを実行することができる。パラメータは、将来のシリコン／ハードウェアに追加／除去することができ、パラメータが別の改定のままとどまる場合には、そのインデックスは変化しない。

【００４２】

本発明の一実施形態におけるＥＭＬの格納定義の一例を以下に記す。１）ＤＩＭＭ検知、２）スタティックな構成、３）トレーニング、及び４）テストである。これら段階は、図３の方法に示す順序で行われてよい。段階の１以上がさらに１以上の副段階を含んでよい。例えばＯＤＴ構成のためには副段階２Ａがあってよい。この副段階に関する情報を段階レジスタに書き込み、スナップショットをとることができる。スナップショットには、全てのチャネルについてＯＤＴ起動マトリックスが含まれてよく、全てのＤＩＭＭの全てのランクについての抵抗値が含まれていてよい。この情報はＥＭＬ実行の内部データであるが、これらパラメータがＤＤＲ３仕様によって定義されていることから、基盤となるＭＣハードウェアの実装の詳細は明らかにされない。

【００４３】

一実施形態では、ＣＳＲ内の特定の命令または特定の制御ビットを利用して、ＳＩＣにある段階／副段階の内部状態のスナップショットをとり、格納するよう命令することができる。例えばストレージを、ＲＡＭとしてのキャッシュ（ＣＡＲ）の位置等のプロセッサの内部メモリに設定することができる。格納されている情報は、ハードウェアの問題のデバッグまたは電力状態の遷移の実装（Ｓ３に入ること、及び、Ｓ３から再開すること等）の際に有用である場合がある。特に、電力状態の遷移の場合には、遷移を始めるプログラムが悪意を有しており、システム管理ＲＡＭ（ＳＭＲＡＭ）その他の保護メモリ等の保護されているメモリ領域へのアクセスを目指してセーブされた状態を傍受しようと試みる場合があるために、注意が必要である。しかし一実施形態では、このような攻撃を回避するためには鍵付キャッシュで十分である場合もあろう。

【００４４】

図６は、本発明の一実施形態におけるプロセッサのブロック図である。図６に示すように、プロセッサ５００はマルチコアプロセッサであってよい。しかし図示の便宜上、１つのコア５０５のコンポーネントしか示していない。図からわかるように、任意のコアに実行論理５１０が含まれていてよく、様々な実施形態では、１以上の実行ユニットが含まれ

てフロントエンドのユニットが提供する処理を実行する。実行の後に、様々なバックエンドのユニットが、結果を受け取り、順序立てて回収する。データに処理を行うべく、実行論理５１０は、レジスタファイル５２０と通信してよく、様々な実施形態では、これは、スカラーレジスタとベクトルレジスタの両方を含む複数のレジスタを含んでよい。

【００４５】

加えて、本発明の一実施形態では、１以上の段階レジスタ５６０を提供してよい。図６の実施形態からわかるように、複数の段階レジスタ565_1－565_nがあってよい。各段階レジスタは、ＳＩＣの任意の段階と関連付けられていてよい。一部の実施形態では、別個のＳＩＣ論理（ＥＣＬ及びＥＭＬ）の各個々のコンポーネントが、最上位の段階として特定されてよく、これらのそれぞれが副段階を含んでよい。わかるように、各段階レジスタ５６５は、任意の最上位段階に加えて、１以上の副段階のストレージを含んでよい。しかし他の実施形態では、読み出されて現在実行中の段階を特定したり、書き込まれて任意の段階の実行を停止したりさせることのできるような１つの段階レジスタのみが提供されてもよい。

【００４６】

さらに図からわかるように、他のコンポーネント（様々な実施形態でＳ３状態等の低電力状態への入退出を制御することのできる電力管理ユニット（ＰＭＵ）５５０等）がプロセッサ内に存在していてよい。図６の実施形態では、プロセッサはさらに、システムメモリ（図６には不図示）と通信してよい統合メモリコントローラ５３０を含んでよい。上述したように、このメモリコントローラの構成は、ＥＭＬを利用して実行することができる。加えて、１以上のキャッシュメモリを提供してよい。示されている実施形態では、キャッシュメモリ５４０が、一時的なデータストアとして利用されてよい。一部の実施形態では、これらキャッシュが、スナップショットデータのストレージの位置を含んでいてよい。インデックスとパラメータデータとを含む形態で格納されていてよいスナップショットデータを格納するものであってよいエントリ５４５が図示されている。パラメータには、ＩＭＣの構成データ、またはＭＣが報告するステータスコードが含まれてよい。これらは、ハードウェアの内部にあってよく、ここでいうインデックス／データのＣＳＲ対は、これらパラメータにソフトウェアがアクセスするためのインタフェースである。スナップショットは、ステータスレジスタがＩＭＣ状態を電力状態遷移に復元する必要がないために、これらインデックス／データ対のダンプであり、包括的である必要はない。一部の実施形態では、ＳＩＣ／ＥＭＬが別のインデックス／データレジスタ対を提供して、ソフトウェアの行う、メモリまたはプロセッサの内部ストレージからのＩＭＣスナップショットの解釈を助けることができる。図６の実施形態にはこの特定の実装例を示したが、本発明の範囲がこの点に限定されないことは理解されたい。

【００４７】

例えば低電力モード（Ｓ３）からの再開は、スナップショットを利用しうる利用モデルであってよい。しかし、このスナップショットをとる動作は、ＳＩＣがプラットフォームＢＩＯＳ前に動作していることから、ソフトウェアにより開始することができない。この代わりに、スナップショットは、自動的に行われ、ＳＩＣによってメモリに格納され（例えばＥＭＬによるトレーニング及びテストが完了した後で）、ＣＡＲは、ＢＩＯＳの開始前に終端されてよい。一実施形態では、スナップショットを格納するアドレスは予め定義されている（例えばチャネル０のランク０のアドレス０）。例えばＯＥＭコードを介して後に、スナップショットを不揮発性ストアに移し、ポインタをＦＩＴに書き込み、ＥＭＬがこれを後で発見できるようにする（例えばＳ３の再開の開始時に）。Ｓ３の再開中に、ＳＩＣは、ＦＩＴに格納されているアドレスを利用して不揮発性ストレージからスナップショットを読み出すことができる（図３の方法に示すように）。従って低電力状態（Ｓ３）から再開すると、この状態にアクセスして、性質上時間のかかりうる様々なメモリ構成パラメータの計算を回避することができる。セキュリティを考慮すると、アドレスは定数であってよく（例えばＦＩＴの署名が保護されていてよい）、ＢＩＯＳがプラットフォームの不揮発性ストレージ内の同じアドレスにスナップショットを格納してよい。

【００４８】
図７は、低電力状態再開中のスナップショットを生成してその後で利用する方法のフロー図である。図７からわかるように、方法６００は、メモリを初期化することで開始されてよい（ブロック６１０）。このＥＭＬを利用したメモリ初期化は、システムの最初の電力投入時に行われてよい。この実施形態では、ＳＩＣは（及びより詳しくはＥＭＬは）、メモリ構成のスナップショットをとるコードを含んでよい（ブロック６２０）。例えばこのスナップショットは、上述したように様々な構成情報を有してよく、最初にはプロセッサキャッシュメモリのＣＡＲ部分に格納されていてよい。ＳＩＣが完結して制御がＯＥＭＢＩＯＳに渡されると、ＢＩＯＳはこのスナップショットを不揮発性ストレージにセーブしてよい（ブロック６３０）。例えばスナップショットはＯＥＭフラッシュに格納されてよい。

【００４９】
図７は、システムの動作中に低電力状態が開始されると仮定している。従って、プロセッサ及びシステムメモリを含む様々なシステムコンポーネントが、低電力状態とされてよい。この低電力状態から再開すると（例えばＳＩＣに従って）、例えばＦＩＴに格納されているポインタからスナップショットの位置にアクセスして、スナップショットを不揮発性ストレージから読み出す（ブロック６４０）。そして、スナップショットを利用して、メモリコントローラ構成を復元することができる（ブロック６５０）。このときに、制御をＯＥＭコード（ＢＩＯＳ）に、またはＯＳに、Ｓ３再開ベクトルの識別情報を介して渡すことができる（ブロック６５０）。図７の実施形態にはこの特定の実装例を示したが、本発明の範囲がこの点に限定されないことは理解されたい。

【００５０】
このスナップショットをＳ３からの再開処理に利用することに加えて、実施形態ではさらに、スナップショットをＥＭＬの実行速度を上げるためにも利用する。つまり上述したように、特にメモリ技術では周波数が増えることから、メモリトレーニングは時間集約的な作業になるので、メモリコントローラは、メモリモジュールと適切な通信を行うためにより正確なタイミングパラメータを利用する。しかしこれらのパラメータは普通ハードコード化されておらず、ランタイムにおいて決定される。従ってメモリトレーニングは、一定のパラメータの一定の範囲内で適切な値を探す処理を含む。周波数が上がると、より多くのトレーニングが必要となるので、ＥＭＬ完了まで時間が長くかかるようになる。同様に、Ｓ３の再開には、プラットフォームをＳ３に入る前の状態に復旧することが含まれる。プラットフォームに対してメモリＤＩＭＭが追加、除去されていないことを想定する場合にはトレーニングは不要であり、通常の起動中にＳ３処理でセーブされたパラメータを単純復旧するだけで、ＭＣ状態を機能するよう復旧することができる。

【００５１】
上述したように、Ｓ３をサポートするためには、ＭＣ状態のスナップショットをプラットフォームのフラッシュにセーブして、ＥＭＬがＳ３の再開動作中にロードできるようにする。しかしユーザがＳ３を開始しておらず、プラットフォームを直接シャットダウンしてしまった場合には、スナップショットが破棄されており、通常起動時に全てのパラメータを始めから決定していく必要がある。つまり、無効であれば、またはユーザが完全起動を望んだ場合にはスナップショットが破棄される、ということである。例えばユーザは、ＢＩＯＳがスナップショットを不揮発ストアに成功裏に移動する前に、プラットフォームの停止を望む場合があり、この場合には無効なスナップショットとなる。

【００５２】
本発明の一実施形態を利用すると、ＥＭＬはＳ３のスナップショットの利用を先ず試行することができる。ＤＩＭＭは追加、削除、または置き換えられているので、メモリテストを実行してパラメータを有効化することができる。テストが失敗したということは、ＤＩＭＭポピュレーションが変化したことを示しているので、ＥＭＬは全てのランクを最初からトレーニングするべく後退処理を実行することができる。これらメモリテスト及びトレーニングは、ランクごとに行われてよい。失敗したランクのみをトレーニングすること

で、消費する時間は全てのランクのトレーニングを行う場合と比べて格段に短くなる。しかし、全ての場合にこの増分的なトレーニングが適しているわけではない。特に、論理から物理へのランクマッピングが変化した場合などがその一例である。一部の実施形態では、プラットフォームのユーザが利用可能な構成可能なフラグを利用して、増分的なトレーニングではなくて完全なトレーニングを強制試行する。

【００５３】

一部の実装例では、Ｓ３のシャットダウンからセーブされたハードウェア状態は、不揮発性ストレージにセーブされた後に、システムの次の通常起動時に再利用されることもある。このようにすると、より迅速な起動が行われるだろう。さらに一定のテストを実行してこれらの安全な設定を有効化することもできる。そして、このテストに失敗したメモリランクについては、増分的なトレーニングを行うようにすることでトレーニングにかかる時間を短くすることができる。従い、システムの再起動からメモリポピュレーションに変化があまりない、または全くない場合には、起動をより迅速に行うことができるようになる。

【００５４】

一実施形態では、タイミングパラメータを物理ランクごとに格納することができる。全ての論理ランクは、「有効」ビットを有して、前の起動時の自身の存在を示し、「存在」ビットは、現在の起動において存在していることを示している。セーブされた構成をロードした後に、ＳＰＤを読み出して、ポピュレーションの変化を検知して、存在フラグをこれに従って更新する。論理から物理へのランクマッピングが変化した場合には、新たなポピュレーションのパターンに従ってタイミングパラメータを移行させることができる。例えば、今は同じランクにマッピングされているが別の物理ランクにマップするために利用される同じＤＩＭＭに２つのランクがあるとすると、元のタイミング設定の平均を、これら論理ランクの両方についての新たな設定として利用することができる。

【００５５】

図８は、本発明の一実施形態におけるメモリ初期化に対する増分的なトレーニングを行う方法のフロー図である。図８に示すように、方法７００は、ＥＭＬ実行期間中であって、前に起動され、そのためのメモリ構成情報またはスナップショットが不揮発性ストレージに前に格納されていたシステムのメモリを初期化するときに、実装されてよい。図８に示すように、方法７００は、不揮発性ストレージ（例えばＯＥＭフラッシュメモリ）にセーブされている状態をロードすることで開始されてよい（ブロック７１０）。この構成情報には、前の起動で存在していた全てのランクについて設定された有効なフラグが含まれていてよい。そして現在のメモリポピュレーションを検知してよい（ブロック７２０）。このようにして、追加／除去されたＤＩＭＭを特定することができる。現在のポピュレーションのポピュレーション化していない（unpopulated）ランクについては、検知に基づいて存在フラグをクリアする。

【００５６】

次に制御はダイアモンド７３０に渡され、ここでループを全てのメモリランクについて実行する。特にダイアモンド７４０では、このランクについて有効なフラグが設定されているかを判断してよい。つまり、セーブされている状態からの有効なフラグをチェックして、設定されているかを判断することができる（前の起動で所与のランクが存在していたことが示される）。判断結果が肯定的な場合には、ブロック７５０で生じたリンクのトレーニングをバイパスすることができ、制御は直接ブロック７６０に渡されて、ランクのテストが行われる。テストにパスすると（ダイアモンド７７０で決定される）、制御はブロック７７５に渡されて、これが最終的に残っているランクであるかが判断される。判断結果が否定的である場合には、ダイアモンド７４０から始まるループの実行を続ける。ダイアモンド７７０で、テストが失敗したと判断されると、制御はブロック７８０に渡されて、そのランクについての有効なフラグをクリアすることができる。

【００５７】

図８の参照を続け、全てのランクがダイアモンド７４０から始まるループを通過すると

、制御はダイアモンド７９０に渡されて、ここで有効なフラグをクリアされたランクがあるか（つまり、そのランクが最後の起動に存在していないことを意味する）を判断する。判断結果が肯定的である場合には、制御がダイアモンド７３０から始まるブロックに戻る。判断結果が否定的である場合には図８の方法は完了してよい。

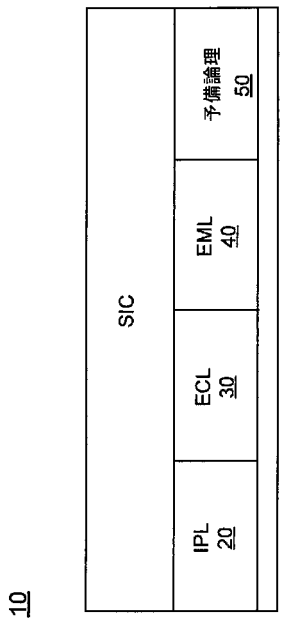
【００５８】

実施形態はコードで実装されてよく、システムを、命令を実行するようプログラミングするために利用されうる命令を格納した格納媒体に格納されてよい。格納媒体には、これらに限定はされないが、フロッピー（登録商標）ディスク、光ディスク、固体ドライブ（ＳＳＤ）、ＣＤ－ＲＯＭ、ＣＤ－ＲＷ、及び光磁気ディスクなど、あらゆる形態のディスク、読み出し専用メモリ（ＲＯＭ）、動的ランダムアクセスメモリ（ＲＡＭ）及び静的ＲＡＭなどのＲＡＭ、消去可能プログラム可能型ＲＯＭ（ＥＰＲＯＭ）、電気的消去可能ＲＯＭ（ＥＥＰＲＯＭ）、磁気カード、光カードなどのあらゆる形態の半導体デバイス、ならびに、その他電気的命令の格納に適したあらゆる形態の媒体などが含まれる。

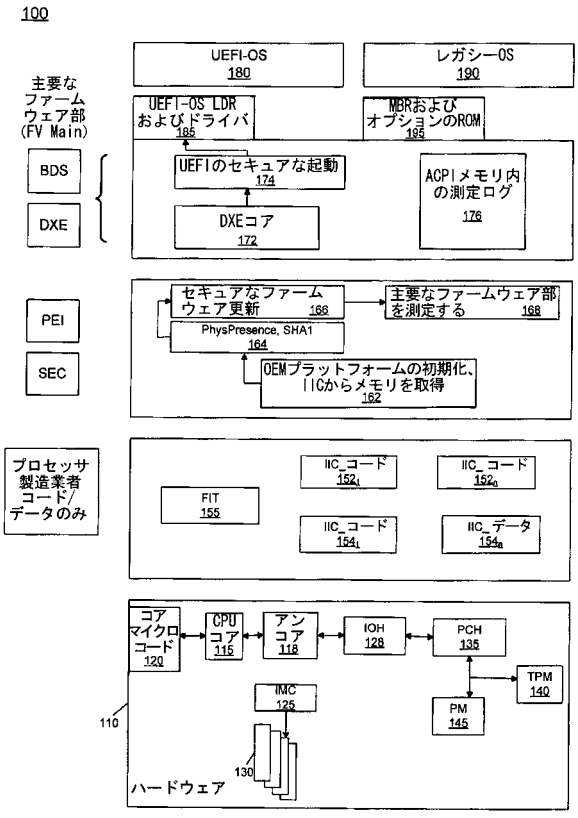
【００５９】

本発明を限られた数の実施形態を参照して説明してきたが、当業者であれば各種の変形例及び変更例を想到するだろう。添付請求項は、本発明の真の精神及び範囲内に含まれるこれら全ての変形例及び変更例を含むことを意図している。

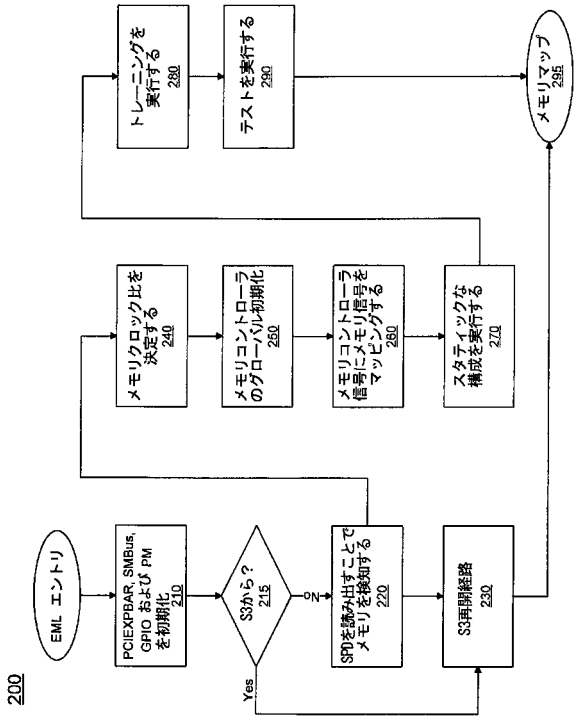
【図１】



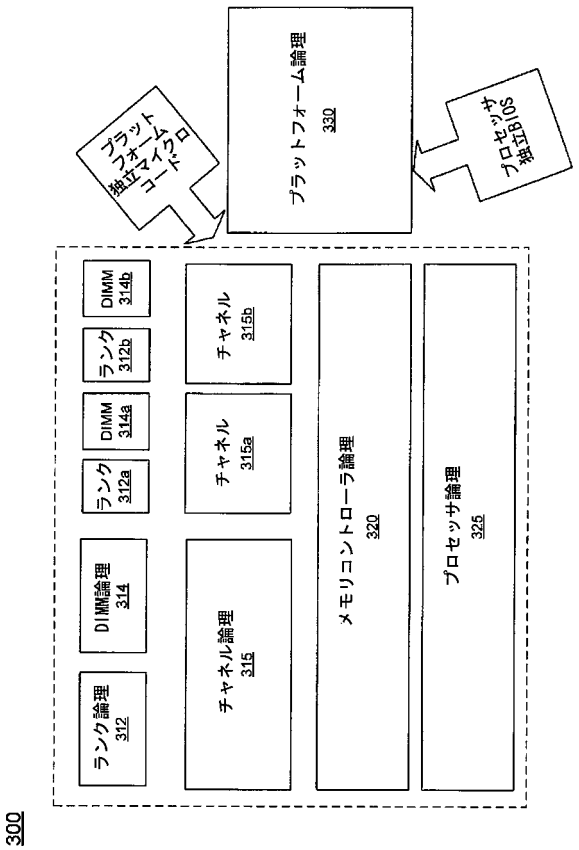
【図２】



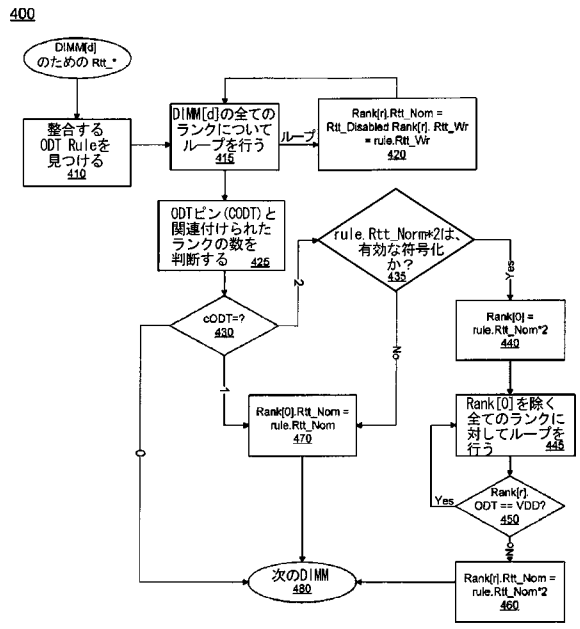
【図３】


200
EML エントリ
PCIEXPBAR, SMBus, GPIO および PM を初期化 210
S3から? 215
Yes
No
SPDを読み出すことでメモリを検知する 220
S3再開経路 230
メモリクロック比を決定する 240
メモリコントローラのグローバル初期化 250
メモリコントローラ信号にメモリ信号をマッピングする 260
スタティックな構成を実行する 270
トレーニングを実行する 280
テストを実行する 290
メモリマップ 295

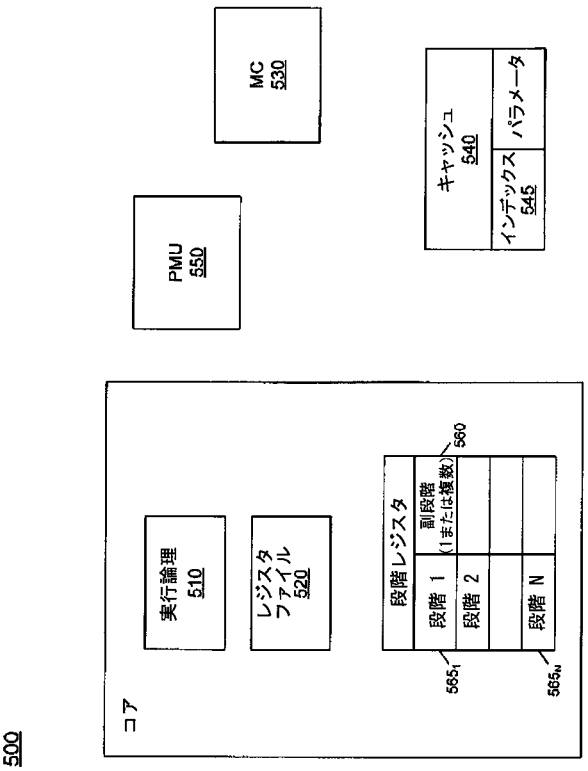
【図４】


300
プラットフォーム論理 330
プラットフォーム独立マイクロコード
プロセッサ独立BIOS
ランク論理 312
DIMM論理 314
ランク 312a
DIMM 314a
ランク 312b
DIMM 314b
チャネル論理 315
チャネル 315a
チャネル 315b
メモリコントローラ論理 320
プロセッサ論理 325

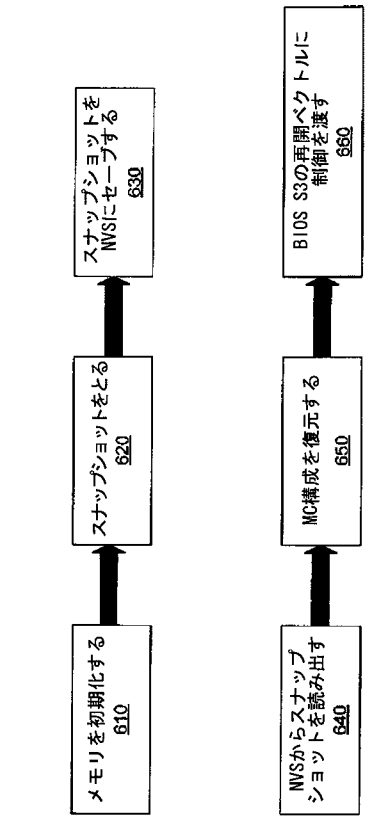
【図５】


400
DIMM[d] のための Rtt_*
整合する ODT Ruleを見つける 410
DIMM[d]の全てのランクについてループを行う 415
ループ
Rank[r].Rtt_Nom = Rtt_Disabled Rank[r].Rtt_Wr = rule.Rtt_Wr 420
ODTピン(CODT)と関連付けられたランクの数を判断する 426
rule.Rtt_Nom*2は、有効な符号化か? 435
Yes
No
cODT=? 430
2
1
0
Rank[0] = rule.Rtt_Nom*2 440
Rank[0].Rtt_Nom = rule.Rtt_Nom 470
Rank[0]を除く全てのランクに対してループを行う 445
Rank[r].ODT == VDD? 450
Yes
No
Rank[r].Rtt_Nom = rule.Rtt_Nom*2 460
次のDIMM 480

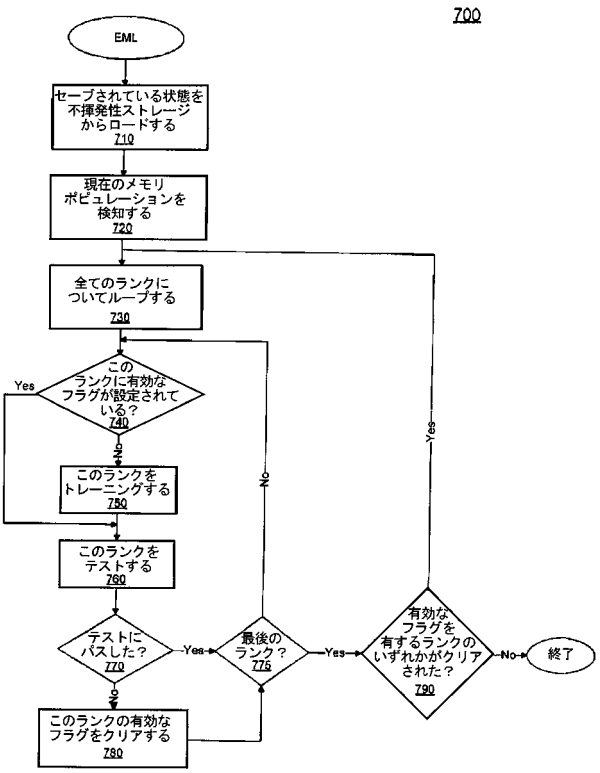
【図６】


500
コア
実行論理 510
レジスタファイル 520
段階レジスタ
副段階 (1または複数) 560
段階 1
段階 2
段階 N
565₁
565N
PMU 550
MC 530
キャッシュ 540
パラメータ
インデックス 545

【図７】


メモリを初期化する
610
スナップショットをとる
620
スナップショットを
NVSにセーブする
630
NVSからスナップ
ショットを読み出す
640
MC構成を復元する
650
BIOS S3の再開ベクトルに
制御を渡す
660
600

【図８】


700
EML
セーブされている状態を
不揮発性ストレージ
からロードする
710
現在のメモリ
ポピュレーションを
検知する
720
全てのランクに
ついてループする
730
この
ランクに有効な
フラグが設定されて
いる？
740
Yes
No
このランクを
トレーニングする
750
このランクを
テストする
760
テストに
パスした？
770
Yes
No
このランクの有効な
フラグをクリアする
780
最後の
ランク？
775
Yes
No
有効な
フラグを
有するランクの
いずれかがクリア
された？
790
Yes
No
終了

フロントページの続き

(72)発明者　ジマー、ヴィンセント　ジェイ.
アメリカ合衆国　カリフォルニア州・サンタクララ・ミッション　カレッジ　ブーレバード・２２００　インテル・コーポレーション内
(72)発明者　ツムジンスキー、クリストフ　シー.
アメリカ合衆国　カリフォルニア州・サンタクララ・ミッション　カレッジ　ブーレバード・２２００　インテル・コーポレーション内

審査官　平井　誠

(56)参考文献　特開２００９－２１１２０４（ＪＰ，Ａ）
特表２００９－５４０６３３（ＪＰ，Ａ）
特開２０１２－００９０２８（ＪＰ，Ａ）
Intel Platform Innovation Framework for EFI Pre-EFI Initialization Core Interface Specification (PEI, ２００３年　9月１６日, Version 0.9, pp.67, 185-190, ＵＲＬ, http://www.intel.co.jp/content/dam/www/public/us/en/documents/reference-guides/efi-pei-cis-v09.pdf

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ　２１