(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4009437号
(P4009437)
(45)発行日　平成19年11月14日(2007.11.14)　(24)登録日　平成19年9月7日(2007.9.7)

(51)Int.Cl.　　F I
G06K 19/07 (2006.01)　　G06K 19/00 N
G06F 11/00 (2006.01)　　G06F 11/00 350A

請求項の数 1　(全 17 頁)

(21)出願番号　特願2001-138073(P2001-138073)
(22)出願日　平成13年5月9日(2001.5.9)
(65)公開番号　特開2002-334317(P2002-334317A)
(43)公開日　平成14年11月22日(2002.11.22)
審査請求日　平成17年4月6日(2005.4.6)

(73)特許権者　503121103
株式会社ルネサステクノロジ
東京都千代田区大手町二丁目６番２号
(74)代理人　110000350
ポレール特許業務法人
(74)代理人　100068504
弁理士　小川　勝男
(74)代理人　100075096
弁理士　作田　康夫
(72)発明者　渡邊　高志
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内
(72)発明者　遠藤　隆
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内
最終頁に続く

(54)【発明の名称】情報処理装置

(57)【特許請求の範囲】
【請求項１】
プログラムカウンタが指し示すプログラムステップを元に動作を行う情報処理装置であって、
上記プログラムステップは、第１条件分岐命令を実行する第１プログラムステップの後に、第２条件分岐命令を実行する第２プログラムステップが一つないし複数続き、
上記第１条件分岐命令を行い、条件フラグについての第１条件判定が真であった場合には、第１処理へ分岐し、上記第１条件判定が偽であった場合には、次の第２プログラムステップへ進み、
上記第２条件分岐命令を行い、上記条件フラグについての第２条件判定が真であった場合には、エラー処理へ分岐し、上記第２条件判定が偽であった場合には、次のプログラムステップへ進み、
上記第２条件判定と、上記第１条件判定において、同じ上記条件フラグを対象として同じ条件判定を行い、
誤動作の際に、上記プログラムカウンタが、上記第１プログラムステップを無処理のまま通過し、次の第２プログラムステップ以降のプログラムステップへ進む情報処理装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は情報処理装置に関し、特に電磁波や放射線、電源の瞬断その他の物理手段により

誤動作を起こし、その装置内部で使用されている情報を取得しようとする攻撃やその情報取得を目的としたその装置の解析に対抗する手段を持つ耐タンパー装置に関する。

【０００２】

【従来の技術】

電子機器が普及している現代において、各種の機器が様々な周波数の電磁波を放射している。例えば、テレビの隣にスピーカを設置すると、テレビの映像に滲みが生じるなどの現象が起こる。また、車に搭載されるコンピュータに代表されるように、強度の大きい振動や不安定な電源といった環境においても誤動作のない、もしくは規定外動作から確実に復帰するための手段を備えた装置が必要とされている。このような現象について説明するために、「タンパーエビデント(tamper evident)」「タンパーレスポンス(tamper response)」「タンパーフリー(tamper free)(耐タンパー(tamper resistant)とも)」といった用語が用いられ、それぞれ、規定外動作検出、該検出後の応答性、該動作に対する耐性を意味する。

【０００３】

一般の装置には、装置本来の目的を正常に果たすために、実際の運用環境におけるノイズなどの誤動作要因をあらかじめ想定して設定された動作マージンがある。タンパーエビデント性を持つ装置は、その装置があらかじめ想定した動作環境外の誤動作環境にさらされたかどうかの履歴を適当な手段で記録することができるため、装置の管理者は装置を常時監視する必要が無く、定期的に履歴を確認することで、誤動作防止のための適切な処置を施すことができる。さらにタンパーレスポンス性を持つ装置は、設計時に想定した以外の誤動作状況になったかどうかを独自に観測して必要な手段で警報などの応答を行うことができる。装置の管理者は、装置が警告を発した場合のみに確認作業を行えば良い。タンパーフリー性を持つ装置は、規定外の動作に陥らないように十分な配慮が行われているか、仮に規定外の動作に陥ったとしても自動的に復旧することができる。または、タンパーという言葉を悪意者からの攻撃と解釈するならば、悪意者の目的物を完全に保護する機構を備えた装置もタンパーフリー装置と呼ぶことができる。例えば、秘密の情報を保持する記憶装置を備える銀行端末などが不正に開封された場合には、その装置自身が自動的にその記憶装置の記憶内容を消去したり破壊したりすることで保護することができる。タンパーフリー性を持つ装置は、管理者が監視を行う必要が無い。

【０００４】

ここでは、タンパーとは環境の変化や悪意者による積極的な攻撃を意味する。

【０００５】

電磁波や放射線、熱、振動など強度などが設計時に意図した範囲を超えたときに、装置はいわゆる誤動作や停止状態に陥る傾向がある。これは、装置が電気的、機械的に動作していることによる。部品点数の少ない装置であれば、単純な総当りを行っても原因の特定と対策にかかるコストは比較的低く抑えることが可能であるが、部品が相互に組み合わされたときに、その種類と方法をすべて網羅的に確認することは不可能であり、したがって誤動作しないことを保証することは非常に困難である。そのため、多数の部品にそのシステム全体の安全性を依存しなければならない装置ほど、耐タンパー性に高度の注意を払うことが必要である。例えば、数１０万から数１００万個のオーダーで高度に集積化された半導体を使用する中央演算装置（CPU）などのマイクロプロセッサや、CPU、メモリ、外部インターフェースなどを数mm角以下に集積したICカードなどがその一例であろう。従来は、そのようなシステムを運用する上で、上位のシステムは下位のシステムの安全性を全面的に信頼することで設計を行ってきたが、これは下位のシステムの故障や誤動作によりシステム全体が不測の停止や誤動作状態に陥る可能性があることを意味する。

【０００６】

図１に、ICカードを示した。ICカード（１０１）には接触端子（１０２）が装着されており、この端子を経由して外部とのやり取りを行う。Vccには電源を供給し、RSTにはリセット信号を、CLKにはクロック信号を、I/OにはI/O信号を伝達する。GNDは接地端子である。

高度な安全性を要求されるICカードなどの場合、図２の中央演算装置（２０１）を使用

して秘密の鍵を使用して暗号処理などの演算を行う。ICカードでは内部と外部とをつなぐのはI/Oポート（２０７）のみであり、CPUの演算データは外部に出力されないことから、内部で記憶装置（２０４）に秘密に保持しているデータ（２０６）は極めて安全であると考えられてきた。しかし、ICカードに外部から放射線や電磁波などのストレスを与えることで誤動作を起こし、その演算結果を観測することにより内部の秘密の鍵情報を取得することができることをBiham氏とShamir氏が「Differential Fault Analysis of Secret Key Cryptosystems」(Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'97, Springer-Ferlag, pp513-525, 1997)において示した。ここに示されるように、演算値の一部にエラーを生じさせその演算結果から秘密の鍵を特定することが可能である。ゆえに、ICカードのシステムが動作しつづけていることだけでは必ずしも十分でない。このように、装置自身がシステム設計者の意図通りに「正しく」動作しているかどうかを把握する必要がある。

【０００７】

【発明が解決しようとする課題】

従来の装置は、外部からの放射線や電磁波、振動などにより誤動作を起こし、その誤動作の結果として予定外の情報漏洩や故障を招くことがあった。装置の電源を不安定にした場合、装置の中で最も電圧異常に弱い部分から誤動作を始める傾向がある。これは装置の接続関係や設計により様々に異なるため、高度に集積された装置のどの部分が弱いかを決定することは困難である。また、高度に組み合わされた装置の相互関係を追跡することは非常に高コストでもあり、引き起こされる誤動作現象を予測することも困難である。

【０００８】

本発明の目的のひとつは、情報処理装置が誤動作の種類に応じて適切に処理を継続もしくは停止することが可能な情報処理装置を提供することにある。

【０００９】

【課題を解決するための手段】

Ｉ.条件分岐命令を複数回連続的に記述する。

【００１０】

グリッチアタックへの対応策の一つとして、条件分岐命令を複数回連続的に記述する構成を本発明の実施例では採る。

【００１１】

グリッチアタックに遭遇してある条件分岐命令が正しく実行されず、実質的にはその命令が実行されなかった場合への対応策である。

【００１２】

それに関連する構成は次の通りである。

１．ICカード用チップ内の一のアプリケーションまたは一の情報へのアクセス権を得るための入力パスワードの正誤を判定するために前記入力パスワードと正しいパスワードとを比較し、両者が一致した場合にはその次の命令を実行し、不一致の場合には分岐命令を実行する第一の分岐命令が記憶装置内に格納され、その第１の分岐命令の後にはそれと同じ命令である第２の分岐命令が前記記憶装置内に格納されていることを特徴とする情報処理装置。

２．前記第１および第２の分岐命令の判定結果に基づき実行される分岐命令での分岐先にはエラー処理命令が設けられていることを特徴とする上記１．記載の情報処理装置。

３．前記第２の分岐命令の判定結果に基づき分岐せずにその次の命令が実行される場合にはその第２の分岐命令の次の命令またはその先の命令には前記アクセス権を付与するルーチンが含まれていることを特徴とする上記１記載の情報処理装置。

４．前記入力パスワードは複数個の文字、数字、符号または記号の列を有し、前記文字、数字、符号または記号の一単位と正しいパスワードの一単位とを比較することにより入力パスワードの正誤判定を行うことを特徴とする請求項１記載の情報処理装置。

５．前記第一の分岐命令に引き続いて前記第二の分岐命令が実行される場合の前記第一の分岐命令の実行開始時刻から前記第二の分岐命令の実行開始時刻までの時間間隔が１マイ

クロ秒以下となるように構成されていることを特徴とする上記１．記載の情報処理装置。
６．前記正しいパスワードがn個(nは自然数とする。)の文字列よりなる場合、前記第１および第２の分岐命令はそれぞれn個あり、前記第１の分岐命令と前記第２の分岐命令とが交互に実行されるようにコーディングされていることを特徴とする上記１．記載の情報処理装置。
７．前記第１の分岐命令と前記第２の分岐命令との間にはそれ以外の命令が配置されていることを特徴とする上記６．記載の情報処理装置。
８．分岐条件を満足する場合にはその分岐命令を実行し、その分岐条件を満足しない場合にはその次の命令を実行する第一の分岐命令が記憶装置内に格納され、その第１の分岐命令の後にはそれと同じ命令である第２の分岐命令が前記記憶装置内に格納されていることを特徴とする情報処理装置。
９．前記第２の分岐命令は前記第１の分岐命令に引き続いて前記記憶装置内の所定アドレスに記憶されていることを特徴とする上記８．記載の情報処理装置。
１０．前記第１および第２の分岐命令は入力パスワードが正しいか否かの判定を行うために用いられるものであり、入力パスワードが正しいとの判定がなされた場合にはその判定の後にICカード用チップ内の一のアプリケーションまたは一の情報へのアクセスする処理が実行されるように構成されたことを特徴とする上記８．記載の情報処理装置。
【００１３】
上記したものは基本的に同じ分岐命令を連続して記載するものである。これに対して次のものは異なる分岐命令を連続的に記載するものである。
１１．ある条件フラグF1の値が０または１のときに分岐命令が実行される第１の分岐命令と、前記条件フラグF1の値が１または０のときに分岐命令が実行される第２の分岐命令とを有し、前記第１の分岐命令実行時に前記条件フラグF1の値に基づき分岐処理が行われないときには、それに続く処理で前記条件フラグF1の値を反転させた後に前記第２の分岐命令が実行されるように構成されたことを特徴とする情報処理装置。
１２．前記第１および第２の分岐命令は入力パスワードが正しいか否かの判定を行うために用いられるものであり、入力パスワードが正しいとの判定がなされた場合にはその判定の後にICカード用チップ内の一のアプリケーションまたは一の情報へのアクセスする処理が実行されるように構成されたことを特徴とする上記１１．記載の情報処理装置。
１３．ある条件フラグF1の値が０または１のときに分岐命令が実行される第１の分岐命令と、前記条件フラグF2の値が0または1のときに分岐命令が実行される第２の分岐命令とを有し、前記第１の分岐命令実行時に前記条件フラグF1の値に基づき分岐処理が行われないときには、それに続く処理で前記条件フラグF1の値を前記条件フラグF2に複写し、その後に前記第２の分岐命令が実行するように構成したことを特徴とする情報処理装置。
ＩＩ.つぎのものは分岐経路の把握に関するものである。分岐命令に従い分岐する場合、分岐前にどの分岐先へ分岐すべきかは遅くともその分岐命令実行直前に判明している。一方、その分岐命令が正しく実行されたか否かの確認が必要な場合がある。この場合、個々の分岐先で特有の処理を行い、その処理結果と事前に記憶した分岐先とを比較照合することにより、正しく分岐したか否かをチェックする機能を有するのが以下の構成である。
１４．分岐条件に基づき複数個の分岐先のうちのいずれか一を指定する分岐命令と、その分岐先にコーディングされている一連の命令を実行した場合には前記複数個の分岐先のうちのどの分岐先経路を通過したかを第１のデータとして記憶し、前記分岐条件の判断の基準となる分岐先データと前記第１のデータとを比較する手段を有することを特徴とする情報処理装置。
１５．前記分岐条件の判断の基準となる分岐先データと前記第１のデータとが一致する場合にはそれに引き続く処理を行い、不一致の場合にはエラー処理を行うように構成されたことを特徴とする上記１４．記載の情報処理装置。
ＩＩＩ.これは．ループ処理の処理回数を制限する機能に関するものである。
【００１４】
これは、ループ処理の処理回数を制限することにより、情報処理装置の異常な動作を抑制

する処理に関するものである。カウンタを複数個設け、ループ内で例えば、同様のカウントダウン処理を行い、各カウンタ値が０となるか複数個のカウンタの値に不一致が生じた場合にはループ処理から抜ける処理を行う。これにより規定回数以上のループ処理実行を抑制するものである。以下のものはその具体的な構成例である。

１６．第１および第２のカウンタの値を初期化し、ループ内で前記第１および第２のカウンタ値をカウントダウンする処理を行い、前記第１および第２のカウンタに対するカウントダウン後の前記第１および第２のカウンタ値が不一致の場合または前記第１および第２のカウンタに対するカウントダウン後の前記第１および第２の両方のカウンタ値が共に第１の所定値の場合には前記ループから抜ける処理を行い、前記第１および第２のカウンタに対するカウントダウン後の前記第１および第２の両方のカウンタ値が共に第２の所定値の場合には前記ループ内の処理を繰り返すように構成されていることを特徴とする情報処理装置。

１７．前記第１の所定値は0であり、前記第２の所定値は自然数であることを特徴とする上記16．記載の情報処理装置。

１８．第１および第２のカウンタの値を初期化し、処理ループ内で前記第１および第２のカウンタ値を更新する処理を行い、前記更新処理後の前記第１のカウンタ値と第２のカウンタ値とが不一致の場合または前記更新処理後の前記第１および第２の両方のカウンタ値が共に第１の所定値の場合には前記ループから抜ける処理を行い、それ以外の場合には前記ループ内の処理を繰り返すように構成されていることを特徴とする情報処理装置。

１９．前記カウンタ値の更新は更新前のカウンタ値からカウントダウンする処理であるか又はカウントアップ処理するものであることを特徴とする上記１８．記載の情報処理装置。

２０．前記第１の所定値は0であることを特徴とする上記１８．記載の情報処理装置。

ＩＶ.情報処理装置の一部を切り離してその部分を使用し続けるような行為を阻止するための手段について説明する。これはそのような部分に関連する独自のカウンタをその装置に設け、その部分を使用するたびにその独自カウンタのカウンタ値を更新し、そのカウンタ値が規定値以下となった場合にはその部分の使用を阻止する機能を設けるものである。以下に具体的な構成を示す。

２１．その装置の一部の使用限度回数を記憶するためのカウンタと、その装置の一部の使用回数に応じて前記カウンタ値を更新する手段と、その装置の一部の累積使用回数が前記使用限度回数に至った場合にはその装置の一部のその後の使用を制限する手段と、その装置の一部の使用が制限された場合にはその装置の他の部分からの指示で前記カウンタ値を更新することによりその使用制限を解除する手段を有することを特徴とする情報処理装置。

２２．その装置の一部の累積使用回数が前記使用限度回数に至った場合にはその装置の一部のその後の使用を拒否するものあることを特徴とする上記２１．記載の情報処理装置。

２３．その装置へのコマンド入力により前記使用制限が解除されるように構成されていることを特徴とする上記２１．記載の情報処理装置。

２４．前記コマンド入力と併せてそのコマンド入力の正当性を確認する手段が設けられていることを特徴とする上記２３．記載の情報処理装置。

【００１５】

【発明の実施の形態】

今、演算装置を考える。電源を不安定にした場合、演算装置の全体ではなく一部分で誤動作が起こることがある。つまり、ある命令を受け取った演算装置の命令実行部分が沈黙状態に陥り、しかしそれ以外の部分が正常に動作しているならば、外部からはある命令を実行しない演算装置と見ることができる。このような誤動作は条件分岐を行う際に顕著な問題となる。

（１）同様な条件分岐命令を複数個並べることによる誤動作防止対策

図３は一般的に用いられるパスワード照合の処理フローである。パスワード判定装置（３００）は、判定部分（３０１）でパスワードが不正解であると判定すると、もう一度パス

ワード入力処理に戻るように動作しようとする（３０２）。この時に上記の誤動作(条件分岐命令の沈黙)が起こったならば、判定部分（３０１）の処理を行わずに次の処理に遷移するため、パスワードの正否に関わらず、最終的に正解時の処理（３０３）に進む可能性がある。

【００１６】

また、図４に示すように、装置がエラーを起こした場合、装置のリセットをその装置の利用者に対して促すために無限ループ（４００）に遷移させることが一般的に行われる。その装置が無限ループ（４０１）内で動作中に誤動作した場合を考える。すると、無限ループに留まるための無条件ジャンプ命令（４０２）が実行されないことが起こり得る。この場合、装置は無限ループを抜け出し、それに続くアドレスに記述されているプログラム（４０３）へ遷移し、以降に記述されているプログラムを順次実行していく可能性がある。これはこのプログラムの作成者が想定していない動作であり、望ましくないものである。

【００１７】

図５に示すように、条件分岐命令を複数回繰り返す（５０２，５０３，５０４）ことで、図４を用いて説明した問題点の発生を防止することができる。同一の条件での分岐命令を複数回繰り返すことは処理フローに何ら影響を与えず、突発的な誤動作であれば、複数回のうちのいずれかが正常に処理される。ここで、２回目以降の条件分岐先（５０３，５０４）を装置の処理を停止する制御を行う装置、あるいは誤動作を警告する装置（５０６）に設定することもできる。というのは、２回目以降の条件分岐で分岐するのは上記のような誤動作による場合に他ならないからである。この時、装置はタンパーエビデント性を持つといえる。なお、誤動作なき場合には処理A（505）または処理B（507）が実行される。

【００１８】

一般的な情報処理装置はプログラムカウンタを元に動作を行っており、そのプログラムカウンタが指し示す場所にあるプログラムを順次実行する。プログラムやデータを記憶する記憶装置は一般に読み書き時に電力を消費する。また複雑な手順を経るため、電源が不安定になるなどにより誤動作が起こった場合にプログラムやデータを正常に読み込めない場合がある。特別な配慮のされていない装置ではこの誤動作の際に、プログラムカウンタの指し示すプログラムステップを無処理のまま通過し、しかしプログラムカウンタは記憶装置に比較して電力消費が少なく簡単な演算で更新されるため、そのまま次のプログラムステップに進むことがある。条件分岐処理を無処理のまま通過することは、条件判定で条件が偽であった場合と等価であり、設計者の意図しない結果をもたらす。例えば、入力値が負の数であれば再度入力処理に戻り、そうでなければ次の処理に進むという処理を考える。誤動作により入力値の判定が行われないまま次の処理に進んでしまうと、後のプログラムは入力値が正の数であることを想定しているため、予期しない誤動作を引き起こす。本実施例のように、２回以上の条件分岐命令を設置することで、条件分岐命令を無処理のまま通過して次の状態に移行してしまう確率を減らすことができる。条件分岐処理は一般的にシステムに変化をもたらさないため、複数回実行してもなんら新たな問題は起きない。また、条件を反転した命令を使用することや、他の条件フラグによる条件判定を行うことで、異なる条件分岐命令を混在させることができる。ある環境要因による誤動作が命令そのものに依存するものであるとき、本発明により連続する同質の誤動作の影響を受ける確率を効果的に減らすことができる。

【００１９】

図5に示した情報処理装置は、条件フラグＺ（ゼロフラグ）を参照し、Ｚ＝０であれば分岐処理（５０２）を行う。本発明では、分岐処理（５０２）の後に再度、同じ分岐処理（５０３）を行う。本実施例ではさらに続けて同じ分岐処理（５０４）を行っている。これは、処理装置の誤動作により分岐処理が行われなかった場合の保護機構であり、条件分岐命令は処理装置の内部状態を変化させない命令であるため、さらに誤動作確率を減らすために、本例に見られるように3回処理を行うのに留まらず、任意の回数だけ行うことができる。2回目以降の分岐処理により分岐が起こるのは、誤動作により1回目の分岐処理を通過した場合のみであるため、2回目以降の分岐処理の分岐先をエラー処理（５０６）とす

ることができる。エラー処理とせず、1回目の分岐処理と同様に処理Ｂ（５０７）に接続すれば、誤動作環境下においても処理を継続することができる。図5に示した条件Ｚ＝０とは、演算の結果がゼロであったことを示すものであり、一般の演算器にはこのような演算の結果を示す条件フラグが用意されている。本実施例ではＺフラグを参照したが、その他のフラグを用いることも可能である。

（２）他の分岐命令の併用による誤動作防止対策

先に述べたように、電源を不安定にしても装置全体が必ずしも不安定になるわけではなく、最も弱いところから誤動作し始める傾向がある。そのため、受ける影響が演算装置の命令毎に異なり、ある命令は正常動作してもある命令は誤動作する場合がある。ところで、演算装置には各条件フラグの０／１に応じて２種類ずつの命令（図６の６０２，６０４）が実装することができる。例えば、Ｚフラグ(ゼロフラグ)が０であるときに分岐する命令（６０２）と、Ｚフラグが１であるときに分岐する命令（６０４）である。そこで、条件フラグのビット値を反転して（６０３）再度違う命令で条件分岐を行う。これは、一時的な誤動作であれば同じ分岐命令を列挙することで対策可能であるが、定常的な誤動作であるときには同じ命令の系列はすべて通過してしまうため、違う命令を織り交ぜることで誤動作検出確率を上げることができる。主な手順は以下の通りである。

１．条件分岐命令を実行。
２．条件フラグの値を反転。
３．上記1.の条件を反転した条件分岐命令を実行。

【００２０】

同等の効果を得るために、ある条件フラグを保存している状態レジスタの値を、別の条件フラグを保存している状態レジスタに複写することで、別命令を用いて条件判定処理を行ってもよい。この場合、

１．条件分岐命令を実行。
２．上記１.で使用した条件フラグ値を、別の条件フラグを保持する状態レジスタに移動。
３．上記２.での移動先に対応する条件分岐命令を実行。

の手順で処理する。ここでも先と同じ理由で、２回目以降の分岐命令の分岐先を警告装置や停止装置（６０８）にすることでタンパーエビデント性を得ることができる。

ところで、異なる命令間で誤動作の発生確率や発生状態が異なる。そのため、同等の処理を多種の方法で行うことが誤動作への対策となる。図6のように、分岐処理（６０２）の後で分岐条件を反転する（６０３）ことで、分岐処理に異なる命令（６０４）を使用することができ、誤動作を効果的に回避することができる。ここでは条件フラグＺについて述べたが、他の条件フラグを用いても同様に処理を行うことができる。

【００２１】

さらに、ビットシフト命令などにより状態レジスタに保存されているある条件フラグの結果を他の条件フラグ位置に反映させ、その結果を利用して多種の条件分岐命令で条件判定を行うことも可能である。状態レジスタ(コンディションコードレジスタ：CCR)とは、演算器で取り扱うデータの状態を反映するレジスタである。一般にＨフラグ、Ｎフラグ、Ｚフラグ、Ｖフラグ、Ｃフラグがあり、それぞれハーフキャリ、ネガティブ、ゼロ、オーバーフロー、キャリを表している。ハーフキャリには、演算で中央ビットについてキャリまたはボローが発生したときに１がセットされ、ネガティブフラグには、演算結果が負の数であるときに1がセットされ、ゼロフラグには、演算結果がゼロであるときに１がセットされ、オーバーフローフラグには、演算結果がオーバーフローを生じた際に1がセットされ、キャリフラグには、演算でキャリが生じた際に1がセットされる。これらは、状態レジスタの中で各1ビットの位置を占め、順番に配置されている。配置はＣＰＵにより異なる場合があるため、ここでは、ＨＮＺＶＣの順番に配置されているものとし、8ビットの状態レジスタの上位3ビットはここでは常に０であるとする。図7は、ゼロフラグが０である場合に処理Ｂ（７０９）を行いたい場合を示している。最初の分岐処理では、通常通りゼロフラグを利用して条件分岐を行う（７０２）。次に状態レジスタ(配置は上記したよ

うに「０００ＨＮＺＶＣ」である)を1ビット右にシフトすることにより、Ｚフラグの位置にあったビットをＶフラグの位置に移動する（７０３）。つまり、Ｖフラグの判定を行うことで実質的に先のＺフラグの判定を行っていることになる（７０４）。さらに状態レジスタを1ビット右にシフトすると、Ｖフラグの位置にあったビットはＣフラグの位置に移動する（７０５）。Ｃフラグの判定を行うことで、先のＺフラグの判定を行うことができる（７０６）。このように複数の異なる命令を使用することで、一部の命令の誤動作を検出することができる。

（３）処理経路の把握

ノイズによる誤動作などによりＣＰＵのプログラムカウンタが異常動作した場合、条件分岐が必ずしも設計者の意図どおりに当該条件に従って処理されないことや、通常先頭から順番に処理されるべき処理順序が、誤動作により変化することが起こりうる。このような事態の発生を前提として、各分岐経路での処理が正しく行われたかどうかを判定する手段を設ける必要がある。その判定は以下の手順に従って行う。

１．分岐情報（どこへ分岐するのかの分岐先の情報）を保存。

２．分岐命令を実行。

３．分岐先での処理命令を実行。

４．分岐先に依存した情報をメモリに保存。

５．分岐先での処理終了後に、上記１．で保存した分岐情報と上記４．で保存した情報を比較。

【００２２】

上記処理５．において比較結果が不一致ならば、分岐処理が期待通りに処理されなかったことを意味する。よってその場合はさらにエラー処理として装置を停止したり、分岐前に遡って再処理を行うなどの後処理を適切に行うことができる。

【００２３】

図8に本実施例に係る処理フローを示す。分岐命令を実行するにあたり、分岐先を決定するための情報を記憶装置に記憶する（８０１）。分岐命令は分岐先情報を元に分岐経路を決定する（８０２）。今、分岐先情報により経路Ａを選択したとすると、分岐経路では、経路に依存した処理ｆａ（８０３）を行い、さらに分岐経路に特有の情報Iaを記憶装置に記憶する（８０４）。処理終了後、分岐先情報と分岐経路情報を比較することにより、正しい分岐経路が選択されたことを確認する。分岐経路が正しくなかったと判定されれば、処理を直ちに終了したり、エラーメッセージを表示するなどの処理を行う（８０７）。

【００２４】

今、例えば、銀行端末機内のプログラムが入金処理を実行する場合を考える。分岐処理に先立ち、分岐先を決定するための情報を保存する。例として、文字列“入金処理”を記憶装置に記憶する。次に分岐命令は文字列“入金処理”を読み、入金処理経路を選択する。入金処理経路では、実際に入金処理を行い、さらに入金処理経路を処理したことを示す文字列“入金処理経路選択”を記憶装置に記憶する。以上の処理経路の処理が終了した後に、分岐処理に先立ち記憶しておいた文字列“入金処理”と実際の処理経路で記憶した文字列“入金処理経路選択”を比較して、望んだ経路が正しく選択され実行されたことを確認することができる。もし、分岐命令実行時の誤動作により、希望していない処理経路である、例えば、出金処理経路に分岐したならば、当該経路処理中に文字列“出金処理経路選択”が記憶される。そのため、処理経路の処理が終了した後に行う比較において、“入金処理”と“出金処理経路選択”が不整合であることを発見することができ、たとえば処理を中断したり、再度分岐処理をやり直すなどの対策を施すことができる。ここでは、処理経路独自の情報として文字列を保存したが、後で処理経路を特定できるならば、数値を使用しても良い。

【００２５】

図9に条件分岐処理においてその条件分岐が正常に行われたかどうかを検知するための手段の一例を示す。入金処理経路では、経路情報として“１”を保存する（９０３）。出勤処理経路では、経路情報として“２”を保存する（９０５）。照会処理経路では、経路情

報として“３”を保存する（９０７）。装置の使用者は経路情報として１，２，３いずれの値が保存されているかを確認することにより意図した通りの処理経路が選択されたかを知ることができる（９０９）。このように、単純な値の設定により経路選択の結果を保持することもできる。また、経路情報を装置固有の値に限定することで、装置使用者が本情報処理装置の出力結果を使用する際の手順を定型化することができるため利便性を向上させることができる。この定型化により検査処理の共通化を行い、プログラムメモリの節約が可能であり、汎用検査処理としてハードウェア化することで高速化も可能である。

【００２６】

処理経路独自の情報を保存するための処理は、処理経路に分岐した後のどのタイミングで何度行っても良い。図10に一例として、処理経路中に2回の数値演算を行う場合を示す。

【００２７】

入金処理の先頭では数値“１”を設定し（１００３）、出金処理の先頭では数値“２”を設定し（１００６）、照会処理の先頭では数値“３”を設定する（１００９）。また、入金処理の末端では数値を３倍し（１００５）、出金処理の末端では数値を４倍し（１００８）、照会処理の末端では数値を５倍する（１０１１）とする。

このとき、最後に得られた数値が“３”ならば、入金処理経路が正しく処理されたことを示し、数値が“４”ならば、入金処理経路の途中で誤動作により出金処理経路に移行したことを示し、数値が“５”ならば、入金処理経路の途中で誤動作により照会処理経路に移行したことを示し、数値が“６”ならば、出金処理経路の途中で誤動作により入金処理経路に移行したことを示し、数値が“８”ならば、出金処理経路が正しく処理されたことを示し、数値が“１０”ならば、出金処理経路の途中で誤動作により照会処理に移行したことを示し、数値が“９”ならば、照会処理経路の途中で誤動作により入金処理に移行したことを示し、数値が“１２”ならば、照会処理経路の途中で誤動作により出金処理に移行したことを示し、数値が“１５”ならば、照会処理が正しく行われたことを示す。それ以外の数値は起こりえない。起こりえない数値が得られた場合は、誤動作が起きていることを意味する。今、入金処理を実行したいとする。分岐先を決定するための情報である文字列“入金処理”を保存（１００１）した後、分岐命令を実行する（１４０２）。分岐命令により入金処理経路に処理を移行し、入金処理経路の先頭で、数値“１”を保存する（１００３）。入金処理を終了し（１００４）、入金処理経路の末端で、先ほどの数値“１”を３倍する処理を行い、結果“３”を保存する（１００５）。処理経路の処理を終了した後で、分岐処理前に保存した文字列“入金処理”から、比較するべき数値は“３”であることを得る。処理経路の処理中に設定された数値が“３”であることを確認すれば（１０１２）、確かに入金処理が行われたことを確認することができる。

このように、起こり得る全ての処理経路が各々独自の値を持つように処理方法を選択することで、処理経路の終了後にどのような経路で処理したかを確認することができるため、経路処理の途中での誤動作も検出することが可能である。分岐先での経路処理中に多くの検査用処理を行えば、より細かく処理経路を把握することができる。

【００２８】

図11は、図10などで示した情報処理装置Aを内蔵する情報処理装置Bであって、情報処理装置Aの出力を検査して演算が正しく行われた場合に正しい演算結果を出力し、演算が正しく行われなかった場合にはエラーであることを示す情報を出力するものである。分岐処理の検算には、情報処理装置Bへの入力値（本実施例では、そのまま情報処理装置Aへの入力値となる）と情報処理装置Aにおける分岐処理の結果の出力データK、Lを使用する。検算処理をハードウェアで実装すれば、データJに複雑な処理を施しても検算に要する時間を節約することができる。また、装置使用者は情報処理装置Bが誤動作を起こした場合にその旨の出力結果を得てそれに応じて再実行などの適切な後処理を行うことができるため、情報処理装置として条件分岐の検算処理までを含めることには大きな効果がある。情報処理装置Bは入力データ（１１０１）をそのまま情報処理装置Aの入力部に入力し（１１０２）、情報処理装置Aの出力（１１０６）と合わせて検算処理を行う。処理の結果、情報処理装置Aの結果が正しいものと判断されればデータ出力部から正常なデータを出力するよ

うにセットし（１１１１）、不正であると判断されればデータ出力部から不正を示すデータを出力するようにセットし（１１１０）、データ出力手段によりセットされたデータを出力する（１１１２）。今、入力データＩ、Ｊが与えられたとき、情報処理装置Aにより Ｉ、Ｊに対して演算が行われる。例えば処理ｆが演算経路に応じた方式でのIを暗号化演算であるとし、処理G_0はＪに０を、処理G_1はＪに１を、処理G_2はＪに２を加える演算とする。入力Ｊの値に依存して経路が正しく選択されたかどうかを情報処理装置Bはその情報処理装置Aの入出力値から判断する。検算処理（１１０７）は、情報処理装置Aにおける経路選択処理（１１０３）と同等の演算を用いて想定される経路を決定し、その場合の処理ＧをＪに対して行い、判定処理（１１０８）で情報処理装置Aの出力値であるKと比較を行い、結果が一致すれば正しく経路が選択されたと判定し、（１１１１）で出力データを正しいデータにセットする。結果が不一致であれば、処理が正しく行われなかったとして（１１１０）により処理の不正を表すデータをセットする。情報処理装置Bは、情報処理装置Aの結果から誤動作の有無を検出する処理を付加したものであり、これによりユーザーは独自に誤動作の有無を検出する必要がなくなる。図11ではさらに上記検算処理を複数回行い、すべての検算に合格した場合（１１０９）に正しい演算結果を出力し、演算が正しく行われなかった場合にはエラーであることを示す情報を出力する情報処理装置である。情報処理装置の設置環境や悪意者からの攻撃の種類により誤動作の発生確率やタイミングは変化するため、必ずしも１度の検算処理が正常に処理されるとは限らない。そのため、複数回の検算処理を行うことで効果的に検算処理自体の誤動作の確率を減らすことができる。例えば、１回の検算処理が確率５０％で誤って演算が正しく行われたと判断する場合では、２回の検算処理が間違いを出力する確率は２５％であり、３回ならば１２．５％となり、７回の検算処理で間違える確率は１％以下となる。また、上記情報処理装置が誤動作を検出した際に、演算がエラーである旨の出力をする代わりに装置全体を停止させることができる。環境の変化や悪意者の意図により誤動作が集中した場合に、誤動作を表す出力データを正常に外部に伝達できなくなることがある。このような場合、情報処理装置側で装置全体に停止命令を発行することがシステムの誤動作を最低限に抑える最善の方法を与える。装置の停止は、一時的であることも恒久的なものである場合もある。

（４）カウンタ処理

カウンタは、あるサービスを提供する回数などの限度を規定する上で重要であり、処理装置使用を提供するサービス運用システムでは不可欠である。誤動作によりカウントダウン機能が誤動作すると、利用者は規定の回数を超えてその装置を利用し続けることができるため、確実にカウント処理を行う必要がある。また、カウントダウン機能の誤動作により、利用者が規定回数以下しか利用できなければ、サービスとしての運用ができなくなる。外部処理の終了待ち時間調整には一定回数のループ処理を利用することが一般的であるが、誤動作によりループ処理の回数に変化が起こるとやり取りの整合性が取れず致命的な問題となる。また、ストリーム暗号を利用する際に、途中で同期がずれるとそれ以降の暗号文全体が解読されてしまう問題が知られている。

【００２９】

そこで、カウンタを２つ以上用意し、それぞれについてカウント処理を行い、すべてのカウンタが終了条件に達した場合にのみ処理を終了する。途中で、一部のカウンタのみが終了条件に達した場合は、誤動作を意味するため、エラー警告を発したり動作を停止するなどの処理を行う。手順を図12および以下に示す。

１．カウンタを初期化する。
２．ループ内の処理を行う。
３．カウンタのカウントダウンを行う。
４．各カウンタの終了判定を行う。
５．各カウンタの終了判定結果を保存する。
６．すべてのカウンタについて、終了判定がNoであれば、2.へ戻る。
７．すべてのカウンタについて、終了判定がYesであれば、処理を終了する。
８．上記6.、7.いずれでもなければエラー処理を行う。

【００３０】
本対策処理では、カウンタを２つ以上用意し、まずそれらを初期化する（１２０１）。次に、ループ処理に入り、ループ内処理を行う(１２０２)。各カウンタをカウントダウンし（１２０３）、カウンタ毎に終了判定を行う（１２０４）。各カウンタの終了判定結果を保存する（１２０５）。この終了判定結果を読み込み、すべての判定結果が一様にNoになっていれば、ループ処理を継続する（１２０２）。すべての判定結果が一様にYesになっていれば、ループ処理を終了する（１２１０）。これ以外のケースでは、一部のカウンタのみが終了状態に達していることになるため、誤動作をしていると判定し、エラー処理を行う（１２０９）。
【００３１】
従来の繰り返し処理装置は、入力されたカウンタ値を減算し、演算結果が０になるまで繰り返し演算を行う。繰り返し処理時の誤動作によりカウンタ値が変化した場合に、カウンタ値が規定値になる前に繰り返し処理を終えてしまうことを防止する実施例を図13を用いて説明する。ここでは、入力データIは２５６未満の数値であるとする（１３０１）。Jを Ｊ＝２５６－Ｉとして生成する（１３０２）。カウントダウン処理は、Ｉ＝Ｉ－１（１３０３）、Ｊ＝Ｊ＋１（１３０４）、つまりコンピュータで一般的なデクリメント，インクリメントであるとする。通常のループ処理と同様にカウンタＩが０であるかを検査する（１３０６）。Ｉが０でなければ、次に演算Ｉ＋Ｊを行い、０である場合にはエラーが起きていないと判断して、次のループへと進む（１３０９）。ここでエラーがあると判断された場合（Ｉ＋Ｊ≠０）は速やかに、装置がそれ以上の処理を行わないように、必要に応じて異常である旨のデータを出力し、リセットなどの処理を行う（１３１０）。１８０６でＩ＝０であった場合も、Ｉ＋Ｊが０であるかを確認し、０であれば終了処理に進み（１３０８）、０でなければエラー処理に進む（１３１０）。カウンタを２５６未満としたのは、実際のＣＰＵにおいて８ビットの数値が利用しやすいためであり、この制限は重要ではない。
【００３２】
ここでは、カウンタを２つ使用する例を挙げたが、３つ以上の場合についても同様に処理することができる。また、カウントダウン処理としてデクリメント、インクリメントを使用したが、その他の演算を用いても良い。
【００３３】
一例として、図14に乗算、除算を使用する場合を示す。ここでは、２５６回のループ処理を行う。まず、０でない数Ｉを生成する（１４０２）。ＪとしてＩの２５７を法とした逆数を設定する（１４０３）。カウント処理では２５７を法として、Ｉは２倍算、Ｊは２での除算を行う（１４０４，１４０５）。カウンタの終了条件はＩ＝１である（１４０７）。カウント終了条件のチェックの後にもう１つのカウンタＪとの整合性をチェックするためにＩ×Ｊｍｏｄ２５７が１であるかどうかを確認し（１４０８,１４１０）、１でなければ誤動作を意味するため、エラー処理を行う（１４１１）。カウント終了しており、カウンタＩとＪの整合性が正しければループ処理を終了する（１４０９）。本実施例では、オイラーの定理を利用している。同定理によると、pを素数として1≦y<pであるyについて $y^{\phi(p)} \bmod p=1$ となる。ここでφ(x)はオイラー数を表す。つまり、本実施例中で用いた法２５７について、２５７と互いに素な数Ｉ(２５７が素数であるため、256以下で０以外の任意の数は２５７と互いに素である)をφ(２５７)＝２５６乗すると、１になる。
（５）装置独自のカウンタ
使用者からの呼び出し回数を制限した情報処理装置について以下、説明する。情報処理装置は独自にカウンタを保持し、呼び出し毎にカウントダウンし、呼び出し回数が一定回数を超えた場合、それ以降の呼び出しに対して処理実行を拒否する。これにより、設計者が意図しない誤動作により想定回数以上の処理実行を防止することができる。例えば、データを出力する処理装置を呼び出す際に誤動作を起こせば、必要以上のデータを外部に対して出力することがある。これはセキュリティ上重要な情報を保持する装置（例えばＩＣカードや現金処理端末装置など）においては重要な情報の漏洩に繋がり、致命的な問題とな

る可能性があるため、本発明による保護が必要である。さらに、実行可能回数を制限することにより、処理装置部分を単独で取り出してきて別装置に内蔵して使用するという不正使用を防止することができる。装置の運用は、以下の手順による。
１．初期化データにより、装置内のカウンタをある値に初期値化する。
２．処理装置は呼び出される毎にカウントダウンし、カウンタが終了条件に達しているかどうかをチェックする。
３．カウンタが終了状態でなければ処理を行う。
４．カウンタが終了状態であれば、処理を拒否して終了する。
５．制御システムは、外部からの要求に応じて初期化データを装置に送る。

【００３４】

図15に示すように、ユーザ（１５０１）は、制御システム（１５０５）に対して装置の初期化要求を行う。制御システムはその要求に応じて装置内部カウンタ（１５０６）を初期化するための初期化コードを、装置に対して送る。装置は初期化コードに応じて内部カウンタを初期化する。ユーザが処理装置（１５０２）を呼び出すと、装置はカウントダウンを行い（１５０３）、カウンタが終了条件に達したかを判定する（１５０４）。カウンタが終了条件に達していなければ、処理を実行し（１５０７）、終了条件に達していれば処理を実行せずに終了する。該装置のカウンタを初期化するには、セキュリティを考慮する場合、暗号化したパスワードの照合などを利用する。

【００３５】

図15に示すように、制御システム（１５０５）は、ユーザ（１５０１）からの装置初期化要求を受ける。制御システムはユーザ要求の正当性を検証した後に、装置（１５０２）に対してカウンタ（１５０６）初期化コードを送信する。装置はカウンタ初期化コードの正当性を検証した後に、カウンタを要求値に設定する。次に、ユーザは装置の実行呼び出しを行う。装置はそれを受けて、内部カウンタのカウントダウンを行い（１５０３）、カウンタが終了していなければ（１５０４）処理を実行する（１５０７）。カウンタが終了していれば、そのまま終了する。呼び出しに対して処理を行わなくなれば、ユーザが制御システムに対して再度、初期化要求を行う。以下同様にして運用する。

【００３６】

装置の内部カウンタの初期化は、ユーザの要求が無くとも制御システムが特定間隔で行っても良い。この場合、図16のように、制御システム（１６０７）は独自に（例えばクロックに同期して)カウントダウンする（１６０７）ことで初期化タイミングを決定し、特定間隔でカウンタ初期化コードを送信する。
装置のカウンタ初期化コードとして、暗号化パスワードなどで保護された方法を用いれば、装置の不正使用に対して安全性を向上させることができる。これにより、装置提供者は装置使用者に対して利用資源の制限を行うことができる。装置提供者は装置使用者が適当な資格を得ることでカウンタ初期化データの値の大小を調整したり、適当なタイミングで再度初期化を行うことができる。

【００３７】

以上、警告や制御停止などによるタンパーエビデント性を強調したが、代わりに再計算処理などへ制御を移すことで、装置は動作を続けることが可能である。異常環境が継続すれば、無限のループに陥る可能性があるが、環境が常に変化しつづける場合、最終的に処理を正常に継続すると期待される。本発明の実施例における異常時に動作を停止するか継続するかの選択はシステム設計者の手にゆだねることができる。高セキュリティシステムであれば警告を発して停止しても良く、高度な無人システムであれば、最大限に復旧継続処理の努力を行うように設定することもできる。以上を必要に応じて組み合わせて使用することで、誤動作に対してより高い検出可能性を得ることができる。

【００３８】

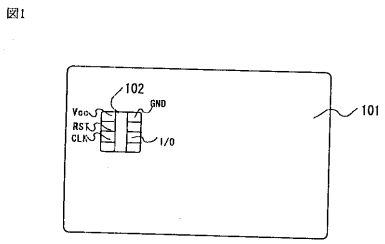
【発明の効果】

本発明の実施例によれば、情報処理装置の誤動作を検出することが可能である。

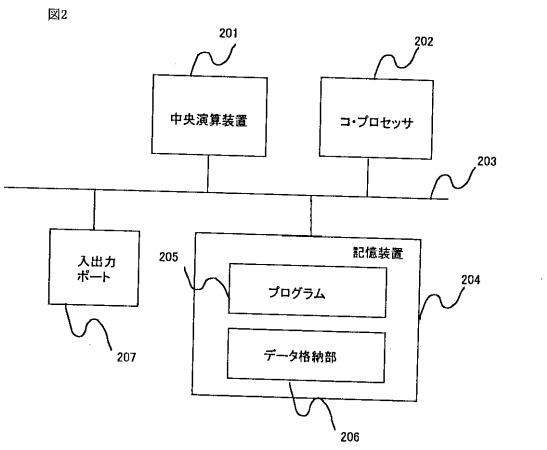
【図面の簡単な説明】

【図１】　ICカード外観図。
【図２】　ICカードのシステム図。
【図３】パスワード照合処理フロー。
【図４】無限ループと、誤動作による無限ループ抜けを示す図。
【図５】条件分岐を複数回実行することによる誤動作検出フロー。
【図６】二種類の分岐命令を併用した誤動作検出フロー。
【図７】分岐条件データを他フラグに移動することを利用した誤動作検出フロー。
【図８】分岐経路に依存した情報を保存することで、経路選択が正しく行われたかどうかを検査する機構を備えた分岐処理フロー。
【図９】分岐経路情報として経路に固有の数値を設定した処理フロー。
【図１０】分岐経路情報として、経路先頭と末尾に検査用処理を設けることで、経路処理中の経路遷移を検出する処理フロー。
【図１１】分岐処理の正しさを複数回検査することにより、異常検出率を向上させる処理フロー。
【図１２】複数のカウンタを用いたループ処理フロー。
【図１３】２つのカウンタの和を使用することにより、カウンタ誤動作を検出するループ処理フロー。
【図１４】２つのカウンタの積を使用することにより、カウンタ誤動作を検出するループ処理フロー。
【図１５】装置独自のカウンタを用いた、処理回数制限機構を備える処理装置の運用方法を表す図。
【図１６】自動的にカウンタを初期化する制御システムと、カウンタによる実行回数制限を持つ処理装置の運用方法を示す図。

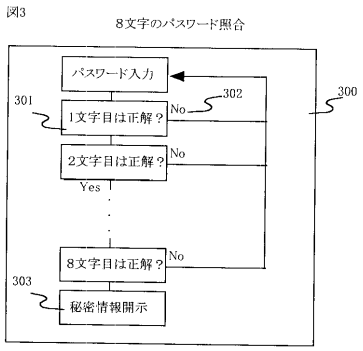
【図１】



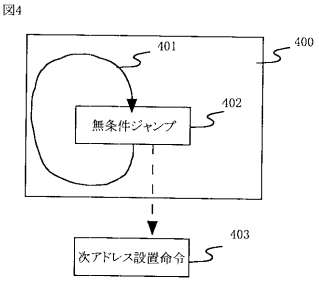
【図２】



【図３】

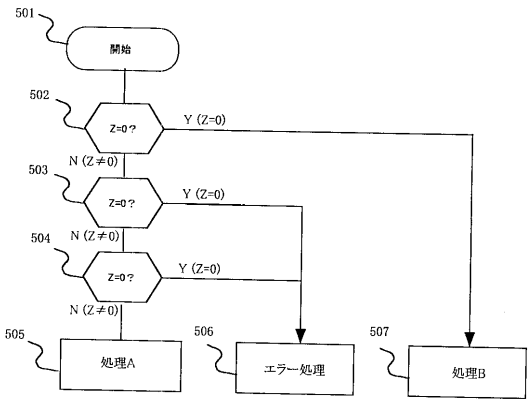


【図４】



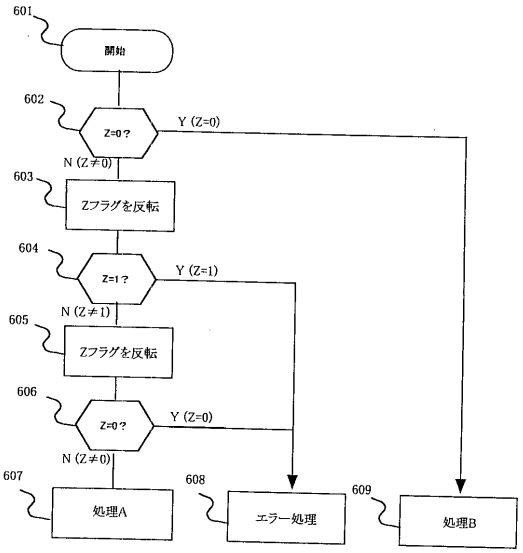
【図５】

図5



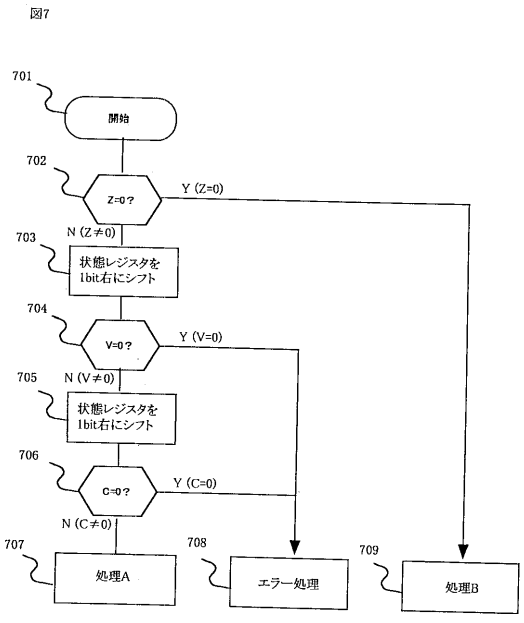
【図６】

図6



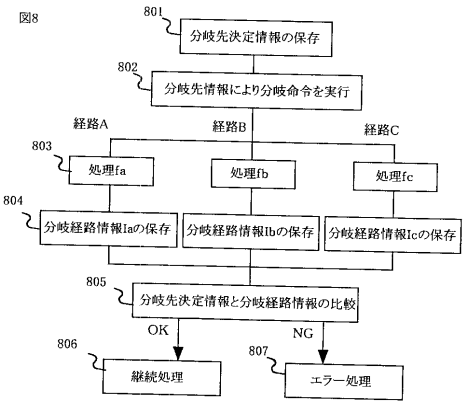
【図７】

図7



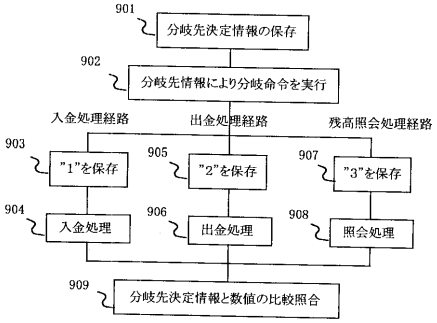
【図８】

図8



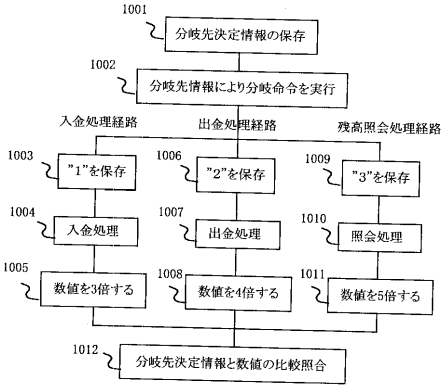
【図９】

図9



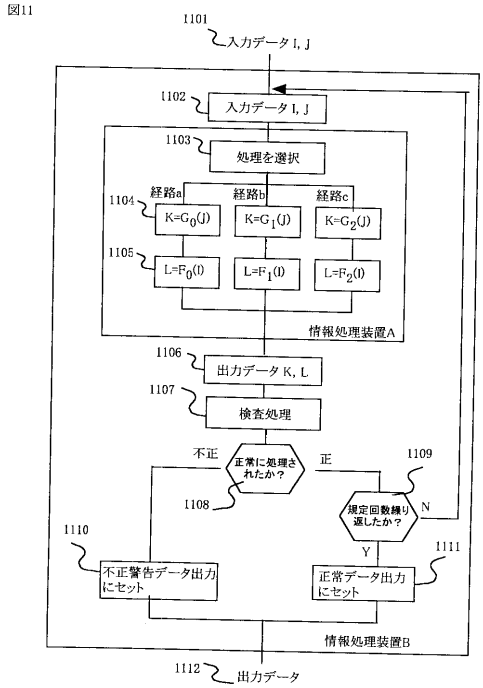
【図１０】

図10



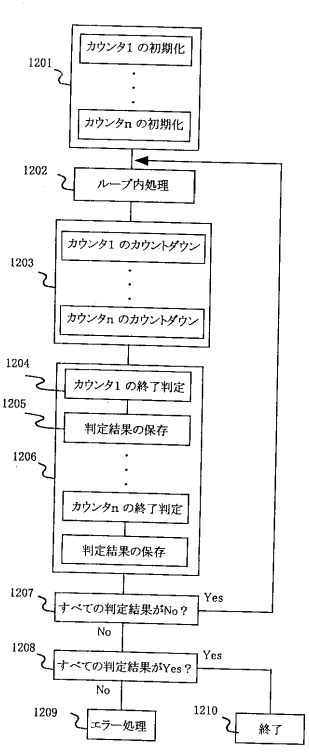
【図１１】

図11



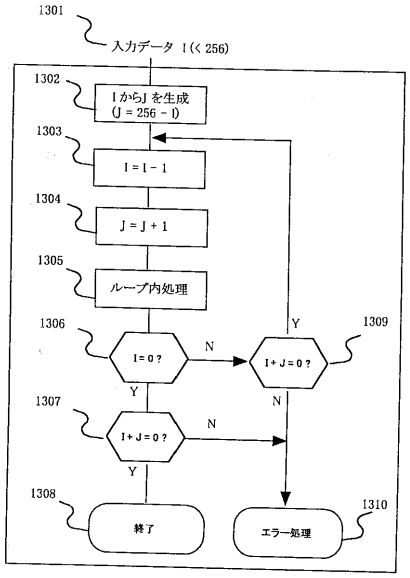
【図１２】

図12



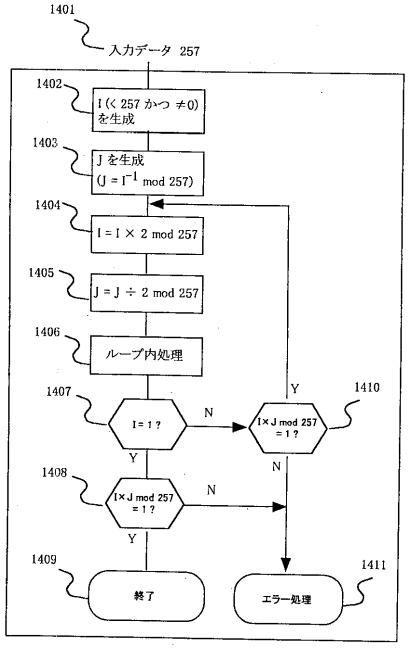
【図１３】

図13



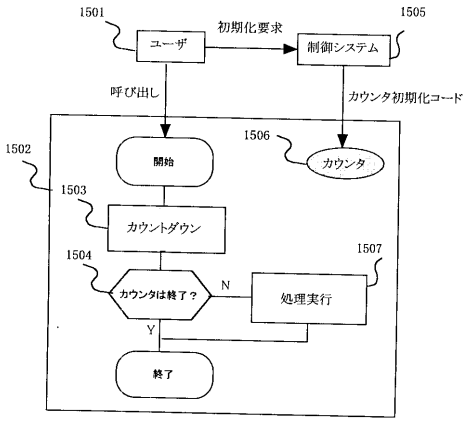
【図１４】

図14



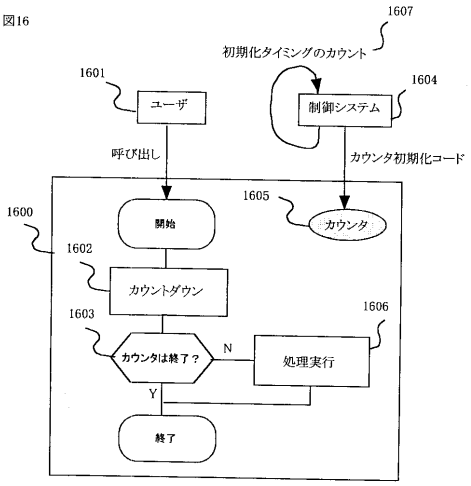
【図１５】

図15



【図１６】

図16



フロントページの続き

(72)発明者　神永　正博
東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内
(72)発明者　中田　邦彦
東京都小平市上水本町五丁目２０番１号　株式会社日立製作所半導体グループ内
(72)発明者　成吉　雄一郎
東京都小平市上水本町５丁目２２番１号　株式会社日立超エル・エス・アイ・システムズ内
(72)発明者　谷本　千晶
東京都小平市上水本町五丁目２０番１号　株式会社日立製作所半導体グループ内

審査官　大塚　良平

(56)参考文献　特開昭５９－１９３６４１（ＪＰ，Ａ）
特開平０２－１６５３４３（ＪＰ，Ａ）
特開昭６２－２４９２３８（ＪＰ，Ａ）
特開昭６２－２９３４４１（ＪＰ，Ａ）
特開平１１－０９６１２０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06K 19/00-19/10
G06F 11/00