(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4233893号
(P4233893)
(45)発行日　平成21年3月4日(2009.3.4)　(24)登録日　平成20年12月19日(2008.12.19)

(51)Int.Cl.　F I
G06F 11/28 (2006.01)　G06F 11/28 310A

請求項の数 18 (全 20 頁)

(21)出願番号　特願2003-59371(P2003-59371)
(22)出願日　平成15年3月6日(2003.3.6)
(65)公開番号　特開2004-13896(P2004-13896A)
(43)公開日　平成16年1月15日(2004.1.15)
審査請求日　平成17年3月28日(2005.3.28)
(31)優先権主張番号　0213149.8
(32)優先日　平成14年6月7日(2002.6.7)
(33)優先権主張国　英国(GB)

(73)特許権者　594154428
エイアールエム　リミテッド
イギリス国　シービー１　９エヌジェイ　ケンブリッジ，チェリー　ヒントン，フルバーン　ロード　１１０
(74)代理人　100066692
弁理士　浅村　皓
(74)代理人　100072040
弁理士　浅村　肇
(74)代理人　100094673
弁理士　林　鉌三
(74)代理人　100091339
弁理士　清水　邦明

最終頁に続く

(54)【発明の名称】データ処理システムにおける命令のトレーシング

(57)【特許請求の範囲】
【請求項１】
データ処理装置であって、
（ｉ）一連の処理サイクルにわたって動作し、プログラム命令に応じてデータ処理動作を実行するデータ処理回路と、
（ii）前記プログラム命令に応じた前記データ処理回路の動作を示す命令トレースデータを生成するトレーシング回路とからなり、
（iii）前記命令トレースデータは、前記データ処理回路によって連続する処理サイクルにわたって実行される処理から発生する一連のトレース事象を符号化した所定の長さの命令トレースワードを少なくとも１つ含み、前記一連のトレース事象は、
（iv）前記データ処理回路によるプログラム命令の実行に対応する少なくとも１つのトレース事象と、
（ｖ）前記データ処理回路によるプログラム命令の実行以外に対応する少なくとも１つのトレース事象と、
を含む、前記データ処理装置。
【請求項２】
前記一連のトレース事象は前記データトレーシング回路によるプログラム命令の実行にそれぞれ対応する１つあるいはそれ以上のトレース事象と、実行されない条件付プログラム命令に対応するトレース事象との組み合わせからなる、請求項１に記載の装置。
【請求項３】

前記データ処理回路によって実行されるプログラム命令の実行にそれぞれ対応する前記１つあるいはそれ以上のトレース事象は、実行されない条件付プログラム命令に対応する前記トレース事象の前に発生する、請求項２に記載の装置。

【請求項４】

前記一連のトレース事象は前記データ処理回路が待ち状態である前記データ処理回路の待ち処理サイクルにそれぞれ対応する１つあるいはそれ以上のトレース事象と、実行されるプログラム命令に対応するトレース事象の組み合わせからなる、請求項１に記載の装置。

【請求項５】

前記データ処理回路が待ち状態である前記データ処理回路の待ち処理サイクルにそれぞれ対応する前記１つあるいはそれ以上のトレース事象は実行されるプログラム命令に対応する前記トレース事象の前に発生する、請求項４に記載の装置。

【請求項６】

前記データ処理回路は処理パイプラインを含み、前記データ処理回路によって実行されるプログラム命令に対応する前記少なくとも１つのトレース事象は前記プログラム命令が前記処理パイプライン内の所定の位置に達したことを表す、請求項１に記載の装置。

【請求項７】

前記所定の位置は、

（ｉ）前記プログラム命令の実行開始、および

（ii）前記プログラム命令の実行の完了のいずれかである、請求項６に記載の装置。

【請求項８】

前記トレーシング回路は前記データ処理回路によって処理されるデータ値を示すデータトレースデータを生成する、請求項１に記載の装置。

【請求項９】

前記データ値は前記データ処理回路に入力されるデータ値を含む、請求項８に記載の装置。

【請求項１０】

前記データ値は前記データ処理回路から出力されるデータ値を含む、請求項８に記載の装置。

【請求項１１】

前記命令トレースデータおよび前記データトレースデータは共通のトレースデータストリーム中でインターリーブされる、請求項８に記載の装置。

【請求項１２】

前記トレーシング回路は、前記命令トレースデータに各処理サイクルにおける前記データ処理回路の処理動作を示す情報が含まれるサイクルアキュレートモードで動作する、請求項１に記載の装置。

【請求項１３】

前記データ処理回路はプロセッサコアである、請求項１に記載の装置。

【請求項１４】

前記トレーシング回路は前記装置からの出力に先立って命令トレースデータをバッファするためのトレースデータメモリを含む、請求項１に記載の装置。

【請求項１５】

データ処理方法であって、

（ｉ）データ処理回路の一連の処理サイクルにわたって、プログラム命令に応じたデータ処理動作を実行し、

（ii）前記プログラム命令に応じた前記データ処理回路の動作を示す命令トレースデータを生成し、

（iii）前記命令トレースデータは、前記データ処理回路によって連続する処理サイクルにわたって実行される処理から発生する一連のトレース事象を符号化した所定の長さの命令トレースワードを少なくとも１つ含み、前記一連のトレース事象は、

（iv）前記データ処理回路によるプログラム命令の実行に対応する少なくとも１つのトレース事象と、
（ｖ）前記データ処理回路によるプログラム命令の実行以外に対応する少なくとも１つのトレース事象と、
を含む、前記データ処理方法。
【請求項１６】
トレースデータ解析装置であって、
（ｉ）データ処理回路によって連続する処理サイクルにわたって実行される処理から発生する一連のトレース事象を符号化した所定の長さの命令トレースワードを少なくとも１つ含む命令トレースデータに応答するアナライザロジックからなり、前記アナライザロジックは、
（ii）前記データ処理回路によるプログラム命令の実行に対応する少なくとも１つのトレース事象と、
（iii）前記データ処理回路によるプログラム命令の実行以外に対応する少なくとも１つのトレース事象と、
を含む一連のトレース事象を表す命令トレースデータを同定する、前記トレースデータ解析装置。
【請求項１７】
トレースデータ解析方法であって、
（ｉ）データ処理回路によって連続する処理サイクルにわたって実行される処理から発生する一連のトレース事象を符号化した所定の長さの命令トレースワードを少なくとも１つ含む命令トレースデータに応じて、
（ii）前記データ処理回路によるプログラム命令の実行に対応する少なくとも１つのトレース事象と、
（iii）前記データ処理回路によるプログラム命令の実行以外に対応する少なくとも１つのトレース事象と、
を含む一連のトレース事象を表す命令トレースワードを同定する、前記トレースデータ解析方法。
【請求項１８】
トレースデータを解析するコンピュータを制御するためのコンピュータプログラムを有するコンピュータが読み出し可能な記録媒体であって、前記プログラムは、
（ｉ）データ処理回路によって連続する処理サイクルにわたって実行される処理から発生する一連のトレース事象を符号化した所定の長さの命令トレースワードを少なくとも１つ含む命令トレースデータに応じて、
（ii）前記データ処理回路によるプログラム命令の実行に対応する少なくとも１つのトレース事象と、
（iii）前記データ処理回路によるプログラム命令の実行以外に対応する少なくとも１つのトレース事象と、
を含む一連のトレース事象を表す命令トレースワードを同定するための実行可能な識別コードを含む、前記記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明はデータ処理システムに関する。より詳細には、本発明はデータ処理回路によって実行中のプログラム命令のトレーシングに関する。
【０００２】
【従来の技術】
データ処理システムの複雑化に伴って、たとえばシステムデバッギング等の一部としてかかるシステムの挙動を分析する包括的で有効なメカニズムを設けることが重要となってきている。さらに、データ処理システムの高速化の結果、通常それに含まれる集積素子の数

も増大し、それに伴い、生成されるトレースデータの量およびその生成速度も増大している。

【０００３】

データ処理システム用にオンチップトレース装置を設けることが知られており、これは集積回路にその回路の動作をトレースし、トレースデータを転送する、あるいはこのトレースデータをバッファし後に転送するための専用回路を設けるものである。ここで、生成可能なトレースデータの量が大きく、そのトレースデータのバッファリングに要するオンチップメモリの増大および／またはトレース機能に専用のピンの数の増大といった、チップからのトレースデータの送出に要する装置の増大が発生するという制約がある。そこで、生成されるトレースデータの量を低減する対策がきわめて有益となる。

【０００４】

ある特定の命令が実行されているかどうかを示す命令トレースデータが命令トレースワードで符号化されるシステムを設けることが知られている。これは、分岐命令で終わる順次の命令の実行を符号化することによって行われる。したがって、４つの命令の順次実行の後に分岐命令が続くことの検出がある特定の命令トレースワードを符号化する。

【０００５】

【発明が解決しようとする課題】

本発明はデータ処理装置を提供するものであり、前記装置は、

（ｉ）一連の処理サイクルにわたってプログラム命令に応じたデータ処理動作を実行するデータ処理回路と、

（ii）前記プログラム命令に応じた前記データ処理動作を示す命令トレースデータを生成するトレース回路とからなり、

（iii）前記命令トレースデータは、前記データ処理回路によって連続する処理サイクルにわたって実行される処理から発生する一連のトレース事象を符号化した所定の長さの命令トレースワードを少なくとも１つ含み、前記一連のトレース事象は、

（iv）前記データ処理回路によるプログラム命令の実行に対応する少なくとも１つのトレース事象と、

（ｖ）前記データ処理回路によるプログラム命令の実行以外に対応する少なくとも１つのトレース事象とを含む。

【０００６】

【課題を解決するための手段】

本発明は、命令トレースワードとしての分岐によって終了する一連の実行を符号化する従来技術には、手書きあるいはコンパイルされたコードに短いループがよく用いられまた、かかるループの最後の分岐間の実行は比較的短いため、命令トレースワードによってこれらのループを圧縮された形態で表現することは非効率的であるという問題があるとの認識に立つものである。この問題に鑑み、本発明は、連続する処理サイクルにわたって実行され、命令の実行に対応する１つ以上のトレース事象とプログラム命令の実行以外のトレース事象に対応する１つ以上のトレース事象とを含む一連のトレース事象を表す命令トレースワードを提供することでこれを解決する。

【０００７】

一方、他に多くの方法が可能であり、好適な方法の１つとして、プログラム命令の実行に対応する１つ以上のトレース事象と実行されない条件付プログラム命令に対応するトレース事象との組み合わせからなるトレース事象のシーケンスを符号化するという方法がある。これは多くの実在のシステムにおける命令トレース動作を符号化する効率的かつ有効な方法であることがわかっている。

【０００８】

実行されない条件付プログラム命令は実行される１つ以上のプログラム命令の前後いずれでもよいことは明らかであるが、本発明の実施例においては、実行されない条件付プログラム命令は実行される１つ以上のプログラム命令の後に続くものとする。これによって、実施が容易となる。

【０００９】
プログラム命令へのデータ処理回路の応答を現す他の好適な方法として、待ち処理サイクルに対応する１つ以上の命令トレースワードと実行されるプログラム命令に対応するトレース事象との組み合わせからなるトレース事象シーケンスを符号化する命令トレースワードを使用する方法がある。
【００１０】
データ処理システムの複雑化、特に多数の異なる機能素子を用いたシステムオンチップ設計が多く用いられるようになった結果、サイクルアキュレート（cycle-accurate）な命令トレーシングを実現しうることが強く求められており、本発明の方法は、かかるサイクルアキュレートなトレーシングを命令トレースワードで効率的に表すことを可能にする。
【００１１】
かかる符号化は、実行されるプログラム命令に対応するトレース事象が、トレースされる１つ以上の待ちサイクルの後に続くようにすることによって簡単に実施することができる。
【００１２】
パイプライン処理システムにおいては、命令パイプライン内の異なる地点で複数のプログラム命令が有効に並列処理される。本発明の実施例では、プログラム命令が処理パイプライン内の所定の位置に到達したとき、そのプログラム命令に対応するトレース事象を処理することによってこれに対処する。
【００１３】
これはその命令の取り出し、復号、実行、あるいはパイプラインの他の地点でありうることは明らかである。プログラム命令はパイプライン内の所定の位置に到達した後打ち切られる場合があり、これは命令の実行が実際には完了しなかったことを示すデータを挿入する必要のある実行に対応するものと解釈される。
【００１４】
特に、パイプライン内でプログラム命令の実行に対応するとみなしうる簡便な位置としては、パイプラインの実行段内におけるそのプログラム命令の実行が開始される位置である。パイプライン内でプログラム命令の実行に対応するとみなしうる簡便な他の位置としては、パイプラインの実行段内におけるそのプログラム命令の実行が終了する位置がある。
【００１５】
命令トレースデータは他のトレースデータとは別のストリームとして設けることができることは明らかであるが、本発明の実施例ではデータ処理回路によって（たとえば、入力あるいは出力として）処理されるデータ値を示すデータトレースデータを生成し、このデータトレースデータと命令トレースデータとを共通のデータストリーム中でインターリーブする。これには、システムからのトレースデータの転送に要するリソースが低減され、また命令トレースデータをデータトレースデータと相関させることによって、データ処理回路の動作をより詳細に分析しうるという利点がある。
【００１６】
データ処理回路は様々な形態をとりうることは明らかであるが、本発明の技術は比較的複雑なプログラム命令と、プロセッサコアに関係付けられたかかる命令への応答とに関するトレースデータの生成に特に適している。
【００１７】
トレース回路は命令トレースデータを生成後ただちにシステムから直接転送するようにすることができるが、本発明の実施例ではトレース回路は命令トレースデータをバッファするトレースデータメモリを含む。これによってトレースデータを実動速度で迅速に収集することができ、また本発明の技術による命令トレースデータの符号化によれば、かかるトレースデータメモリを効率的に利用することができる。
【００１８】
また、本発明はデータ処理方法を提供するものであり、その方法は、
（ｉ）プログラム命令に応じてデータ処理回路の一連の処理サイクルにわたってデータ処

理動作を実行し、
（ii）前記プログラム命令に応じた前記データ処理回路の動作を示す命令トレースデータを生成し、
（iii）前記命令トレースデータは、前記データ処理回路によって連続する処理サイクルにわたって実行される処理から発生する一連のトレース事象を符号化した所定の長さの命令トレースワードを少なくとも１つ含み、前記一連のトレース事象は、
（iv）前記データ処理回路によるプログラム命令の実行に対応する少なくとも１つのトレース事象と、
（ｖ）前記データ処理回路によるプログラム命令の実行以外に対応する少なくとも１つのトレース事象とを含む。
また、上述した命令トレースデータ生成技術を補足する観点から、本発明は生成されたかかるデータを分析するメカニズムおよび技術を提供する。
【００１９】
したがって、本発明はまたトレースデータ分析装置を提供し、前記装置は、
（ｉ）データ処理回路によって連続する処理サイクルにわたって実行される処理から発生する一連のトレース事象を符号化した所定の長さの命令トレースワードを少なくとも１つ含む命令トレースデータに応答するアナライザロジックからなり、前記アナライザロジックは、
（ii）前記データ処理回路によるプログラム命令の実行に対応する少なくとも１つのトレース事象と、
（iii）前記データ処理回路によるプログラム命令の実行以外に対応する少なくとも１つのトレース事象とを含む一連のトレース事象を表す命令トレースワードを識別する。
【００２０】
生成されたトレースデータの分析はまた分析の方法とみなすことができ、適当なコンピュータプログラムの制御のもとに動作する汎用コンピュータを用いて簡便に実行することができる。
【００２１】
本発明の上記の目的、特徴および効果、またその他の目的、特徴および効果は以下の実施例の詳細な説明および添付図面から明らかとなろう。
【００２２】
【発明の実施の形態】
図１はメモリ６に記憶されたプログラム命令を実行するプロセッサコア４を内蔵するシステムオンチップ集積回路２を示す。メモリ６には実行すべきプログラム命令とプロセッサコア４への入力として用いられ、プロセッサコア４からの出力として生成されるデータ値の両方が記憶される。トレーシング回路８はプロセッサコア４および集積回路２の他の部分に関係付けられており、各回路の動作を示すとレースデータを生成する。かかるトレーシング回路８の一般的機能については、英国ケンブリッジのＡＲＭ社のＥＴＭ製品の機能に準じるものとする。かかるＥＴＭ回路はトレーシングを開始するためのトリガポイント等のオプションや、命令とデータの両方をトレースする機能を有する。本技術は命令トレースデータをかかるシステム内で表現する方法の改良に関する。
【００２３】
トレーシング回路８によって生成されるトレースデータは、バッファメモリ１０に記憶される。このメモリはトレースデータメモリとして機能し、生成されるトレースデータが適当なインターフェース回路を介して集積回路２から分析コンピュータ１２に送出される前に、通常は最高速度でこれを一時記憶する。バッファメモリ１０の一般的な動作については、英国ケンブリッジのＡＲＭ社の埋め込み型トレースバッファ（ＥＴＢ）製品の提供する機能に準じるものとする。
【００２４】
簡略化のため図１には示さないが、集積回路２はシステムオンチップ設計の一部としてその他多くの機能素子を有することは明らかであり、かかる異なる要素の動作を詳細かつ効

率的にトレースすることはシステムオンチップ設計全体の開発およびデバッギングにとって非常に有益である。

【００２５】

図２はプロセッサコア４によって実行される一連のプログラム命令（この場合ＡＲＭ命令）を示す。トレーシングはユーザーの条件に応じて、サイクルノンアキュレートモードあるいはサイクルアキュレートモードで実行される。サイクルアキュレートモードは、より多くの情報を生成し、待ち状態が符号化されるように、この情報を表すためにより大きなスペースを要する。

【００２６】

命令トレースワードで表される命令処理動作のトレース事象をアトム（atom）と呼び、アトムはさまざまな形態をとりうる。ノンサイクルアキュレートモードでトレースされるアトムは、ある特定の命令が実行されたこと、あるいはある特定の条件付命令がその条件を満たさないために実行されなかったことを示すものである。したがって、図２の例ではＭＯＶ命令およびＳＵＢＳ命令の両方が実行されてＥ個のアトムが得られ、ＡＤＤＮＥ命令は不等条件コードを満たさないため実行されない。２つの命令の実行とそれに続く条件不満足によるＡＤＤＮＥ命令の不実行とに対応するトレース事象シーケンスは符号化すべきシーケンスとして取り扱われ、所定の長さの命令トレースワードとして表される。

【００２７】

その後分岐命令Ｂが実行され、処理はＬＤＲ命令に移行する。ＬＤＲ命令が実行され、データワード<data1>が転送される。Ｂ命令とＬＤＲ命令の実行は、これらを複合したデータストリームに埋め込まれたあるデータ事象の発生によって終了する連続する２つの実行として取り扱われる。

【００２８】

最後の２つの命令は、実行されるＣＭＰ命令と実行されないＢＥＱ命令であり、ＢＥＱ命令は等条件を満たさず、したがってこの命令が実行されないため分岐が発生する。これらＥ個およびＮ個のアトムはシーケンスとして取り扱われ、命令トレースワードＥＮとして符号化される。

【００２９】

サイクルアキュレートモードでも同じ命令シーケンスが実行される。このモードのアトムには、ある命令が実行され処理サイクルを消費することを示すＷＥアトム、ある命令が条件を満足せず実行されないが処理サイクルを消費することを示すＷＮアトム、およびたとえば多処理サイクル命令のＷＥアトムに待ちサイクルが設けられることを示すＷアトムがある。

【００３０】

最初の３つの命令はＷＥＷＥＷＮを表す命令トレースワードとして符号化される。分岐命令Ｂは３つの処理サイクルを消費する。最初のサイクルはＷＥサイクルであり、それ自体の命令トレースワードとして符号化される。次の２つのサイクルは待ちサイクルであり、次のＬＤＲ命令のＷＥサイクルと組み合わせられ、ＷＷＷＥを表す命令トレースワードとして符号化される。最後の２つの命令はＷＥＷＮを表す命令トレースワードとして符号化される。

【００３１】

命令トレースワードの詳細な符号化はさまざまな形態をとりうることは明らかであるが、使用できる符号化の一例を次に説明する。

【００３２】

１　Ｐ－ヘッダ

１．１　初めに

ここでは、（上述した命令トレースワードを用いた）ETMv3におけるプロトコル変更のすべてを説明する。ETMv3は（英国ケンブリッジのＡＲＭ社の提供する）ETMv2に対する互換性を有しない。

【００３３】

ｐ－ヘッダはETMv2パイプスタットによって現在与えられる情報からなり、他のデータと同じパケットストリームに現れる。その意味を正確に記述するには新たな技術を導入しなければならない。ETMv2中の16のパイプスタットは以下の情報を表す。
-命令がいつ実行されるか
-実行されるそれぞれの命令が条件コードをパスしたか否か
-データがいつＦＩＦＯに入れられたか
-上述の事象がそれぞれどのサイクルで発生したか
-トリガの発生
-トレース捕捉をいつ不能にすべきか

【００３４】

分岐ファントムパイプスタットはあるサイクルで２つの命令が実行されたことを示すものとみなすことができる。

【００３５】

ETMv3は不能化されたトリガとトレースを示す代替的なメカニズムを提供する。それらは、ここでの目的上それぞれパイプスタットとＷＴに代わるものとみなすべきものである。トレースストリームに現れたデータは常に直近のサイクルまたは命令に対応する。その結果、それに続くデータがあるかどうかを示す必要はなく、したがって残る14のパイプスタットをさらに7つの可能性に絞り込むことができる。それらは以下の３つのアトムの組み合わせを表すものと見ることができる。
-Ｗ：サイクルの境界（すべてのパイプスタットはこれを表す）
-Ｅ：条件コードをパスした命令
-Ｎ：条件コードを満たさなかった命令

【００３６】

これらをマッピングすると次のようになる。

パイプスタット	アトム
IE, DE	**WE**
IN, DN	**WN**
WT, DW	**W**
TR, TD	なし
PTIE, PTDE	**WEE**
PTIN, PTDN	**WEN**
PNIE, PNDE	**WNE**
PNIN, PNDN	**WNN**

【００３７】

１．２　エンコード

１．２．１　Ｐ－ヘッダの意味

次に、かかるパイプスタットのシーケンスをこれら３つのアトムのシーケンスとして表すことができる。サイクルアキュレートなトレーシングが要求されない場合、Ｗアトムを除くことによってよりわかりやすいストリームを生成することができる。この場合、ほとんどの命令が条件コードを満たすため、データトレーシングが不能化される場合、Ｅのストリングが発生する可能性が高い。データトレーシングがイネーブルされると、トレースストリームへのデータの挿入を可能にするためにこのシーケンスを終了しなければならない。しかし、この場合、データトレースのために生成されるトレース量がいずれにしてもパイプスタットのトレースを上回る。サイクルアキュレートなトレーシングが要求される場

合、連続するサイクルで実行される命令の場合のＷＥのストリングと同様にＷのストリングが発生する可能性が高い。

【００３８】

１．２．２　配置

分岐アドレス以外のETMv2のあらゆるヘッダには、同じサイクルに後続のパケットがあったか否かを示すＣビットが含まれていた。トレースストリームにパイプスタット情報を埋め込むことによって、このビットは不要になる。これによって新たなヘッダを設けるスペースが得られる。新たなヘッダの符号化は次の通りである。

【００３９】

値	サイクルアキュレート	新ヘッダ	ペイロード（最大バイト）	説明
Cxxxxxx1		no	追加アドレスバイト(4)	分岐アドレス
00000000		yes	なし（繰り返し）	A-sync
00000100	! cycacc			予備
00000100	cycacc	yes	サイクル数(5)	サイクル数：$1\text{-}2^{32}$xW（下記）
00001000		yes	下記(14)	I-sync
00001100		yes	なし	トリガ
0TT0SS00		no	データ(4)	ロードミスデータ　, TT = tag (1-3)
01x10000				予備
01A1TT00		no	アドレス(5)	ロードミス発生, TT = tag (1-3), A = 後続アドレス有り（アドレストレーシングの場合）
00x1xx00				予備
00AMSS10		no	アドレス(5) データ(4)	通常データ, A = 最初のデータパケット （予想アドレス）
010xxx10				予備
01100010		yes	なし	データ抑制 (DSup)
01100110		yes		無視
01101010		no	なし	値をトレースせず (VnT)
01101110		no	コンテキスト ID(4)	コンテキスト ID
0111xx10				予備
1NEEEE00	! cycacc	yes	なし	フォーマット 1: 0-15xE, 0-1xN
1000FF10	! cycacc	yes	なし	フォーマット 2: 1x(N/nE), 1x(N/nE)[a]
1N0EEE00	cycacc	yes	なし	フォーマット1: 0-7x(WE), 0-1xWN
1E1WWW00	cycacc	yes	なし	フォーマット3: 1-8xW, 0-1xE
1000FF10	cycacc	yes	なし	フォーマット2: 1xW, 1x(N/nE), 1x(N/nE)[a]
1001xx10				予備
101xxx10				予備
11xxxx10				予備

[a]：ビット１は最初の命令を表し、ビット２は２番目の命令を表す。

連続するサイクルにおけるＥのランレングスはサイクル境界を無視する場合のランレングスより小さいことが多いため、７つ以上の命令の実行に対応するヘッダスペースが再使用される。Ｗの平均ランレングスはこれより小さい。

【００４０】

１．２．３　ｐ－ヘッダの不在

サイクルアキュレートモードではＮで終わる多数のＷを有するフォーマットはない。条件コードを満たさない命令はインターロックされそれらと前の命令との間で待ち状態が発生することはない。一般的に条件付命には、一般的に同じ条件コードを有する他の条件付命令か、条件コードを設定する命令のいずれかが先行する。これらは通常１サイクルで実行されるデータ処理命令である。

その結果、現在のところ、このフォーマットを使用する場合にヘッダスペースが正当化されることは少ないものと見られ、フォーマット１で十分である。

【００４１】

１．３　サイクルカウント

サイクルカウントｐ－ヘッダが設けられ、それに１～５バイトのデータが続く。各バイトのビット７の１は、分岐アドレスと同様に他のバイトが続くことを示す。このようにして最大で３２ビットが出力され、不明の高位ビットはいずれも０である。この値はＷの数である。非周期的なi-syncの後にこのパケットは発生した場合、これはそのi-syncの前のトレースストリームにさらに挿入すべきＷの数を示す。その後このパケットが発生した場合、他のｐ－ヘッダと同様にＷはその出力ポイントで挿入しなければならない。

【００４２】

前者は必ず使用しなければならず、トレース領域間のサイクルの数を効率的に出力することができる。後者の使用は任意であり、あるトレース領域内の長いギャプをより効率的にトレースすることが可能となり、またトレース出力を不能化しながら、トレースを再度イネーブルする前に（ロードミスデータ等の）他のパケットを出力することができる。

【００４３】

サイクルカウントが０である場合、これはカウンタのオーバーフローを示す。この場合、ギャップの長さは不明となる。

【００４４】

１．４　ｐ－ヘッダの生成

以下にｐ－ヘッダの生成規則を示す。これらの規則はアーキテクチャではなく具体化の過程で定義される。ｐ－ヘッダは正しいアトム群にアンパックされるものであればよいが、十分ではない。これは、極端な場合、装置によって１サイクル当たり１つのｐ－ヘッダが生成されることを意味するが、それが好適な実施態様とはいえないことに注意しなければならない。下記の規則はETM10Jで実施されるものである。将来の実施においては改良されるものと考えられる。

【００４５】

分岐ファントムとは他の実行される命令と並列する正しく予測された分岐を指し、それによってETMv2においてはＰ*パイプスタットが作成される。これは他の命令と並列に実行されない折り返し分岐を指すものではない。

【００４６】

各サイクルで、ＥＴＭは出力されるｐ－ヘッダを計算する。これはただちに出力されるか、あるいは次のサイクルまで保管される。保管される場合、次のサイクルで生成されるｐ－ヘッダはそれに基づくものでなければならず、したがってアトムが失われることはない。

【００４７】

１．４．１　ノンサイクルアキュレートモード

あるサイクルで２つの命令が実行される（すなわち、分岐ファントム（phantom）が発生し）、第1の命令が条件コードを満たさない場合、フォーマット２が用いられる。これはＰＮ*パイプスタットに対応する。これ以外のすべての場合にフォーマット１が用いられ

る。
【００４８】
フォーマット２のｐ－ヘッダは常にただちに出力される。保管されたフォーマット１のｐ－ヘッダは以下の事象が発生したときに出力される。
-条件コードを満たさなかった命令が実行される（Ｎビットがセットされる）
-他のデータパケットが生成される
-Ｅのカウント値が15に達する
-Ｅのカウント値が14に達し、かつ次のサイクルに分岐ファントムが含まれる
-次のサイクルに条件コードを満たさない分岐ファントムが含まれる（ＰＮ*パイプスタット）
-トレーシングが不能になる
-トリガが発生する
Ｅのカウント値が１４に達したとき常にフォーマット１のＰ－ヘッダを出力することが簡便である。
【００４９】
次のサイクルに（条件コードを満たさない）分岐ファントムが含まれるか否かを見るための別途のパイプライン段は、かかる分岐ファントムをトレースするか否かは問題ではないため不要である。これをトレースしない場合、トレーシングは不能となり、いずれにせよｐ－ヘッダを出力しなければならない。
【００５０】
１．４．２　サイクルアキュレートモード
条件コードを満たさない分岐ファントムが発生した場合、前述したようにフォーマット２が用いられる。１つの命令が実行された場合、フォーマット１が用いられる。命令が実行されない場合、フォーマット３が用いられる。
【００５１】
フォーマット２のｐ－ヘッダは常にただちに出力される。保管されたフォーマット１のｐ－ヘッダは以下の事象が発生したときに出力される。
-条件コードを満たさなかった命令が実行される（Ｎビットがセットされる）
-他のデータパケットが生成される
-Ｅのカウント値が７に達する
-次のサイクルに分岐ファントムが含まれる（Ｐ*パイプスタット）
-トレーシングが不能になる。
-次のサイクルで命令が実行されない
-トリガが発生する
【００５２】
保管されたフォーマット３のｐ－ヘッダは以下の事象が発生したときに出力される。
-命令が実行される
-他のデータパケットが生成される
-Ｅのカウント値が８に達する
-次のサイクルに分岐ファントム（Ｐ*パイプスタット）すなわち条件コードを満たさない命令が含まれる。
-トレーシングが不能になる。
-トリガが発生する
【００５３】
次のサイクルで実行される命令があるか否かを見るための別途のパイプライン段は、分岐ファントムを見る場合と同じ理由で不要である。
【００５４】
サイクルカウント値は非周期的なi-syncの後にのみ出力される。これは、i-syncの後の最初のサイクルで出力され、かかるサイクルでは他のトレースは出力されない。理論的には他のｐ－ヘッダと平行して、あるいは分岐アドレス付きで出力されうるが、実施し易いも

のではない。カウンタ自体はあるトレース領域の最後のｐ－ヘッダに続くサイクルに１（０ではない）にリセットされる。これによって、カウンタを出力に先立ってリセットすることが不要になる。

【００５５】

１．５　同期

同期には、a-sync（アラインメント）、i-sync（命令フロー）、およびd-sync（データアドレス）の３つの形態がある。適正な同期を行うにはこの３つが周期的に発生しなければならない。

【００５６】

１．５．１　度数

各同期形態は一般にはトレースが出力されるｎサイクルにつき（すなわちＴＤサイクルはカウントされない）少なくとも一度は発生しなければならない。ｎは同期度数レジスタの値である（０×78）。この実施態様では特殊なケースでは、通常はオーバーフローを防止するために、さらに最大でｎサイクル同期が遅れる可能性がある。そのアルゴリズムは実際には実施態様によって定義される。ETM10Jは（トレースが実際に出力されるサイクルだけでなく）トレーシングがイネーブルされている間のすべてのサイクルをカウントするものと考えられ、したがって必要最小限より高い頻度でトレースを出力されるときこの定義に適合する。

【００５７】

これは、ツールがETMv2の場合と同様にポート幅に基づいて同期度数カウンタの値を変化させようとすることを意味することに注意しなければならない。これは、ツールはバッファ内にある設定された数の同期ポイントを設けようとすると考えられ、また幅の狭いポートを用いるより広いポートを用いるときより少ないトレースサイクル数でバッファを満たすことができるためである。

【００５８】

１．５．２　A－sync（配列同期）

５つ以上のa-sync　ｐ－ヘッダ（00000000）のシーケンスが周期的に出力され、それに１つの空白ｐ－ヘッダ（10000000）が続く。これは、４７以上の０ビットに１が続くストリングと等価である。A-sync（ヌル）バイトは圧縮解除プログラムには通常無視される。

【００５９】

同期をとるには、圧縮解除プログラムはこのシーケンスを探さなければならず、これ以外の方法では発生し得ない。トレース捕捉装置は通常バイトに位置合わせされるが、サブバイトポートは必ずしもそうではない。したがって、圧縮解除プログラムは必要に応じてa-syncシーケンスに続くすべてのデータを位置合わせし直す必要がある。

【００６０】

次のバイトはヘッダであるが、このヘッダは任意のタイプである。すなわち、i-syncヘッダである必要はなく、他のｐ－ヘッダであってもよい。

【００６１】

たとえば、バイトに合わせたシステムにおいては、a-syncシーケンスとそれに続く単一のＥｐ－ヘッダは00　00　00　00　00　80　84と表すことができる。しかし、捕捉の結果として、この同じシーケンスが１ビットポートから１ビットずれると、01　00　00　00　00　40　42と表される。これはポートサイズが１、２あるいは４ビットである場合には発生せず、トレース捕捉装置はトレースの全実行過程にわたって完全に正確である。しかし、１サイクルでもあやまって捕捉すると、すなわちTRACECTLピンの不安定性のために余分なサイクルを捕捉するか１サイクルを捕捉しそこなうと、それ以降の捕捉にはすべてずれが生じる。各バイト内で新たに位置合わせを行うことによって、このエラーを局所に限定することができる。また、これによって３あるいは６ビットといったより一般的でないポートサイズで実施することが可能になる。

【００６２】

１．５．２．１　桁そろえ

47の０ビットのストリングには少なくとも５つの連続するゼロバイトがなければならない。連続するゼロバイトが４つしかない場合、構成しうるゼロビットの最大長はそれらのゼロバイトについては46：32、その周囲の各バイト（それ自体はゼロではない）については７である。したがって。５以上の連続するゼロバイトが生成されないことを証明しなければならない。

【００６３】

a-sync以外のゼロバイトが発生しうる場所は次の通りである（そのパケットタイプとともに示す）。

-命令アドレスの最後のバイト（分岐アドレス、i-sync）
-データアドレスの最後のバイト（通常のデータ、ロードミス発生）
-サイクルカウントの最後のバイト（サイクルカウント）
-データ（通常のデータ、ロードミスデータ）
-コンテキストＩＤ（コンテキストＩＤ、i-sync）
-ＬＳＭアドレス（LSiP　i-sync）

【００６４】

すべてのパケットがヘッダによって分離されていなければならないため、単一のパケット内のバイトのシーケンスのみを考慮すればよい。上記のケースには可能な最大長である５バイト以上からなるものはない。上記のリスト中二度以上発生するパケットタイプは２つだけである。そのうち、i-sync中のコンテキストＩＤは一方の側にi-syncヘッダを、また他の側にi-sync情報バイトを有し、そのどちらのゼロではありえないため放棄することができる。残るのは次の２つのケースである。

-通常のデータ：データアドレスの最後のバイトはゼロであり、それにゼロデータが続く。データはゼロである場合圧縮される。圧縮を行うため、ゼロでありうるのはデータの最初のバイトだけであり、したがってこのケースは起こりえない。

-LSiP i-sync：ＬＳＭアドレスがゼロであり、それにゼロ分岐アドレスが続く。これは分岐を生じなかったリセットベクトルのデータ命令を示し、それに他の例外ベクトルからこのリセットベクトルへの直接分岐が続く。これが発生する可能性はきわめて低い。さらに、ＬＳＭアドレスの前にはそれがLSiPであることを示すようにセットされたビット７を有するi-syncバイトがある。最悪の場合、つまり次のアドレスが空白ｐ－ヘッダ（0x80）である場合、ゼロビットの最大ランが47に減少する（５つのゼロバイトが40、ｐ－ヘッダが７）。これはいずれにせよ有効なa-syncとして機能する。理由はｐ－ヘッダは無視することができ、トレースはこのシーケンスに位置合わせされていることである。次のヘッダが0x80である場合、ゼロの最大ランは46に減少し、これはa-syncに対しては無効である。したがって、このケースでは偽のa-syncが発生する可能性はない。

位置合わせ同期が達成された後は、a-syncバイトは圧縮解除プログラムに無視されるが、６以上の取り消しバイトが必要である場合（８バイトのポートサイズが実施される場合）これらを取り消しバイトとして用いることはできないことに注意しなければならない。したがって、別途の取り消しコードが必要である。空白ｐ－ヘッダをこの目的に用いることができるが、それにかわって圧縮解除エラーの検出を容易にするために0x66が選ばれている。

【００６５】

１．５．３　i-sync（命令同期）

圧縮解除プログラムはa-syncシーケンスを発見すると、i-syncヘッダを探さなければならない。これをまとめると次の通りである。

通常の i-sync	LSiP i-sync
I-sync ヘッダ (1 バイト)	I-sync ヘッダ(1 バイト)
コンテキストID (0-4 バイト)	コンテキストID(0-4 バイト)
情報バイト(従来のTFOヘッダ)	情報バイト(従来のTFOヘッダ)
次の命令のアドレス(4 バイト)	LSM アドレス(4 バイト)
	次命令のアドレス(1-5 バイト)

【００６６】
これは以下の点を除いてETMv2におけるＴＦＯパケットと同様である。
-i-sync ｐ－ヘッダの後に来ること
-コンテキストＩＤは情報バイトの前に来ること（これは、a-sync値が５回連続して発生することを防止するためである）
-情報バイトがゼロになり、a-syncと競合することを防止するために情報バイトのビット０は常にセットされること
【００６７】
ETMv2の場合と異なり、周期的なi-syncパケットが存在することは命令の実行を意味しない。むしろ、i-syncはいつでも発生しうる。命令アドレスは周期的な同期を行う場合にも常に実行すべき次の命令のアドレスを示す（従来は周期的同期は実行中の命令のアドレスを与えるものであった）。
【００６８】
前述した同期頻度の２倍以内で周期的同期が発生しない場合、オーバーフローが起こる。
【００６９】
１．５．４　d-sync（データアドレス同期）
これはETMv2の場合と同様に５バイトのデータアドレスを検索することによって行われる。
【００７０】
１．６　他の新しいヘッダ
１．６．１　トリガ
トリガが発生すると、必要な他のヘッダとともにトリガヘッダが出力される。ＴＰＡは機構を介してトリガを検出する。
【００７１】
１．６．２　無効
無効ヘッダには効果はない。これは、ポート全体を満たすだけのトレースがないときにトレースを出力しなければならない場合にトレースポートの未使用バイトに使用される。
【００７２】
１．７　トレース捕捉装置（ＴＣＤ）への信号
上述したように、捕捉されたトレースは純粋なビットストリームである。しかし、ＴＣＤはこの情報からどのサイクルでデータを捕捉すべきか、またいつトリガが発生するかを判断することはできない。Error! Reference source not foundおよびError! Reference source not found..の項で説明したように、この情報をTRACEDATA[0]と関連づけて有する信号TRACECTLが出力される。
【００７３】
１．８　プログラマのモデルチェンジ
以下の場所にトレースポートサイズを記述するために３ビットのフィールドが設けられる。
-制御レジスタのビット６：４－使用するポートサイズの選択

-PORTSIZE出力－使用されるポートサイズを示す

【００７４】

より幅の広いトレースポートを可能とするために新たな符号化が定義される。

価	新符号？	ポートサイズ
000		4 ビット
001		8 ビット
010		16 ビット
011	新符号	24 ビット
100	新符号	32 ビット
101	新符号	48 ビット
110	新符号	64 ビット
111		予備

48ビットおよび64ビットのポートはETM10Jによってサポートされない。ある特定の装置のFIFOの幅より大きなポートサイズ（たとえば、ETM7の場合の64ビット）はそのポートサイズがFIFOの幅に等しいことを示す。かかる装置はFIFOを有しない場合もある。

【００７５】

図３はノンサイクルアキュレートモードにおけるトレーシング回路８の動作の概略を示すフローチャートである。

【００７６】

最初にこのモードに入った時、すなわち命令トレースワードを生成したとき、処理はステップ１４に進む。このとき考察対象である命令が、同じサイクルにおいて（たとえば分岐予測の結果）他の命令として発生する条件コードを満たさない分岐ファントムであると考えられる場合、処理がステップ１４に戻る前に、ステップ１６においてフォーマット２の命令トレースワードの出力がトリガされる。同じサイクルにおける他の命令としての条件コードを満たさない分岐ファントムが検出されない場合、処理はステップ１８に進み、現在考察中の命令が条件コードを満たすか否かのテストが実行される。条件コードを満たさない場合、処理はステップ２０に進み、フォーマット１の命令トレースワードが出力され、それに続いてステップ２２で実行された命令のカウント値が０にリセットされ、処理はステップ１４に進む。ステップ１８においてフォーマット１の命令トレースワードの出力がトリガされない場合、処理はステップ２４に進み、トレースデータストリームにおけるインターリーブを必要とする処理動作によって生成されたデータパケットがあるかどうかが判定される。かかるインターリーブが発生すると、これはステップ２０において命令トレースワードを出力するためのトリガとして機能する。

【００７７】

データパケットが生成されていない場合、処理はステップ２６に進み、カウント値がインクリメントされる。ステップ２８においてカウント値が１５に達したか否かが判定される。これはフォーマット１の命令トレースワード中で符号化することのできる最大の値であり、したがって１５に達していた場合、ステップ２０において命令トレースワードを出力しなければならない。ステップ２８においてこの条件が満たされない場合、処理はステップ３０に進み、カウント値は現在１４であり、次のサイクルが分岐ファントムであるか否かが判定される。この条件が満たされると、処理はステップ２０に進む。次に、ステップ３２、３４および３６が実行され、各ステップは、それに対応する次のサイクルの条件が条件コードを満たさない分岐ファントムであること、トレーシング動作の不能化、あるいはトレーシングトリガ事象の発生であることを検出すると、フォーマット１の命令トレー

スワードのトリガとして機能する。ステップ３２あるいは３４でテストされる事象の１つが発生すると、フォーマット１の命令トレースワードがトリガされるため、後続のテストは実行されないことが理解されよう。

【００７８】

全体として見ると、図３の処理は終了事象が発生する前に、すなわちステップ１８で条件コードを満たさない命令が見つかる前に（カウント値１５で示される）最大で１６までの隣接して実行される命令を有効にカウントする機能を有する。

【００７９】

図４はサイクルアキュレートモードでトレース回路８によって実行される処理の概略を示す。ステップ３８で条件コードを満たさない分岐ファントムの初期テストが実行された後、ステップ４０でテストが実行され、考察中のサイクルが待ち処理サイクルであるか否かが判定される。待ち処理サイクルである場合、処理は、命令の実行、データパケットの生成、カウント値が８に到達すること、次のサイクルでの分岐ファントムの発見、トレーシングの不能化、あるいはトリガ事象の発生といった終了事象が発生するまで待ちサイクルＷをカウントするループに進み、終了事象が発生すると、フォーマット３の命令トレースワードの出力がトリガされ、待ちサイクルのカウント値は０にリセットされる。

【００８０】

ステップ４０のテストで待ちサイクルが検出されない場合、終了事象が発生するまで実行される命令をカウントするループに入り、終了事象が発生すると、フォーマット１の命令トレースワードが生成され、実行カウント値が０にリセットされる。終了事象は、データパケットの生成、カウント値が７に達すること、次の分岐ファントムが検出されること、トレーシングの不能化、次のサイクルで命令が実行されないこと、あるいはトリガ事象の発生である。このループはおおむね図３のループと同様であるが、符号化が多少異なり、特に最大実行命令カウント値は、ランを短い長さに制限する３つのビットによってのみ表される。

【００８１】

図５は、上述した技術にしたがって（コンピュータ１２等で）命令トレースワードを含むトレースデータを解析する際に実行される処理の概略を示すフローチャートである。ステップ４２において、システムはトレースデータワードが受信されるまで待機する。ステップ４４において、受信されたトレースワードに一連のマスクがかけられ、比較動作が実行されて、そのトレースワードが上述した符号化にしたがった命令トレースワードであるかどうかが識別される。この種の動作は命令の復号と同様であるとみなすことができる。

【００８２】

ステップ４４で命令トレースワードが同定されると、処理はステップ４６に進み、その命令トレースワード中のフィールド値が読み出され、ステップ４８において、その命令トレースワードによって表され、したがって分析者に利用可能となるべき命令トレース事象動作アトムを再生するのに用いられる。ステップ４４でのテストで命令トレースワードが同定されない場合、処理はステップ５０に進み、そのトレースワードはデータトレースワードとして取り扱われる。

【００８３】

図５の処理はこの方法にしたがって動作する特殊なハードウエアあるいはコンピュータプログラムの制御下で動作する汎用コンピュータによって実行しうることは明らかである。図５に示す処理の実行に使用可能なかかる汎用コンピュータの一例のアーキテクチャを以下に説明する。

【００８４】

図６は上述した技術の実施に使用しうるタイプの汎用コンピュータ２００の概略を示す。汎用コンピュータ２００は中央演算処理装置２０２、ランダムアクセスメモリ２０４、リードオンリーメモリ２０６、ネットワークインターフェースカード２０８、ハードディスクドライブ２１０、表示ドライバ２１２およびモニタ２１４、およびキーボード２１８とマウス２２０を有するユーザー入出力回路２１６を含み、これらはすべて共通バス２２２

を介して接続されている。動作時において、中央演算処理装置２０２はコンピュータプログラム命令を実行し、それらの命令はランダムアクセスメモリ２０４、リードオンリーメモリ２０６、ハードディスクドライブ２１０のうち１つ以上に記憶されるか、あるいはネットワークインターフェースカード２０８を介して動的にダウンロードされる。この処理の実行結果は表示ドライバ２１２およびモニタ２１４を介してユーザーに表示することができる。汎用コンピュータ２００の動作を制御するためのユーザー入力は、キーボード２１８あるいはマウス２２０からユーザー出力回路２１６を介して受信することができる。コンピュータプログラムはさまざまなコンピュータ言語で記述しうることは明らかである。コンピュータプログラムは記録媒体で記憶および分配するか、汎用コンピュータ２００に動的にダウンロードすることができる。適当なコンピュータプログラムの制御下で動作しているとき、汎用コンピュータ２００は上述した技術を実行することができ、上述した技術を実行する装置を形成するとみなすことができる。この汎用コンピュータ２００のアーキテクチャにはかなりの変更が可能であり、図６は一例に過ぎない。

【００８５】

本発明の実施例を添付図面を参照して詳細に説明したが、本発明はこれら詳細な実施例には限定されないことは明らかであり、特許請求の範囲に述べる本発明の範囲および要旨から逸脱することなくさまざまな変更が可能である。

【図面の簡単な説明】

【図１】生成されるトレースデータを解析する汎用コンピュータに接続された、プログラム命令に応答するデータ処理回路およびオンチップトレーシング回路を含むデータ処理装置の概略図である。

【図２】一連のプログラム命令と、かかる命令をノンサイクルアキュレートモードおよびサイクルアキュレートモードの両方で命令トレースワードに符号化する方法を示す。

【図３】ノンサイクルアキュレートモードにおけるトレーシング回路の符号化動作の概略を示すフローチャートである。

【図４】サイクルアキュレートモードにおけるトレーシング回路の符号化動作の概略を示すフローチャートである。

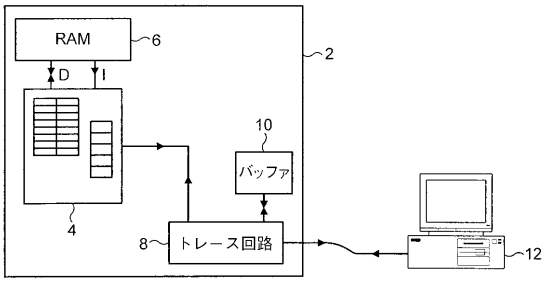
【図５】命令トレースワードを含むトレースデータの解析の概略を示すフローチャートである。

【図６】図５の解析技術の実施に用いることのできる汎用コンピュータのアーキテクチャの概略図である。

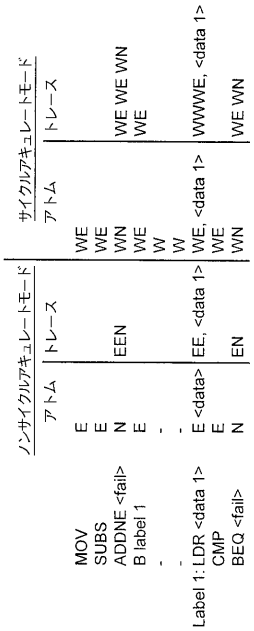
【符号の説明】

２　集積回路
４　プロセッサコア
６　ランダムアクセスメモリ
８　トレース回路
１０　バッファ
１２　分析コンピュータ
２０２　中央演算処理装置
２０４　ランダムアクセスメモリ
２０６　リードオンリーメモリ
２０８　ネットワークインターフェースカード
２１０　ハードディスクドライブ
２１２　表示ドライバ
２１４　モニタ
２１６　ユーザー入出力回路
２１８　キーボード
２２０　マウス

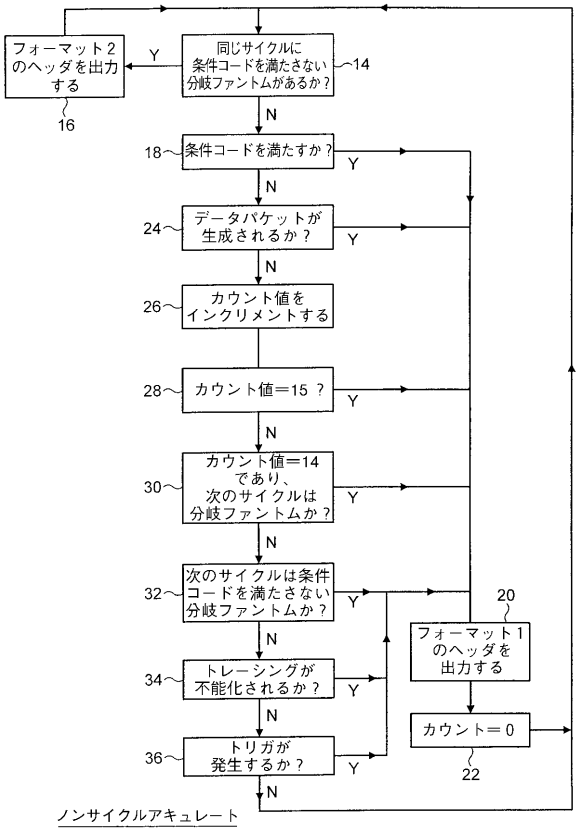
【図１】


RAM
6
D
I
2
10
バッファ
4
8
トレース回路
12

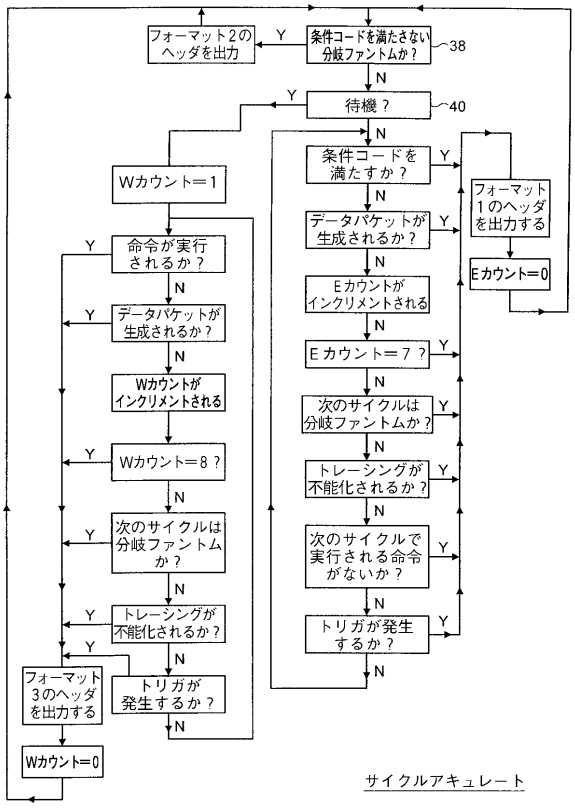
【図２】


ノンサイクルアキュレートモード
アトム
トレース
MOV
SUBS
ADDNE <fail>
B label 1
Label 1: LDR <data 1>
CMP
BEQ <fail>
E
E
N
E
EEN
E <data>
E
N
EE, <data 1>
EN
サイクルアキュレートモード
アトム
トレース
WE
WE
WN
WE
W
W
WE, <data 1>
WE
WN
WWWE, <data 1>
WE WN

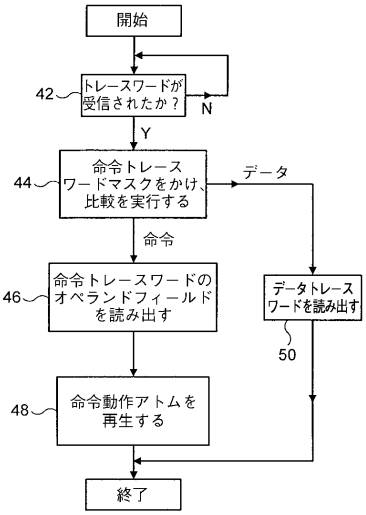
【図３】


フォーマット2のヘッダを出力する
16
Y
同じサイクルに条件コードを満たさない分岐ファントムがあるか？
14
N
18
条件コードを満たすか？
Y
N
24
データパケットが生成されるか？
Y
N
26
カウント値をインクリメントする
28
カウント値＝15？
Y
N
30
カウント値＝14であり、次のサイクルは分岐ファントムか？
Y
N
32
次のサイクルは条件コードを満たさない分岐ファントムか？
Y
N
34
トレーシングが不能化されるか？
Y
N
36
トリガが発生するか？
Y
N
20
フォーマット1のヘッダを出力する
カウント＝0
22
ノンサイクルアキュレート

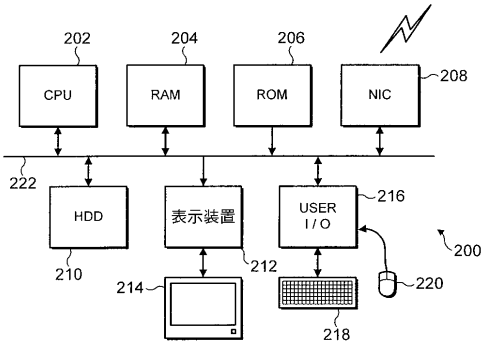
【図４】


フォーマット2のヘッダを出力
Y
条件コードを満たさない分岐ファントムか？
38
N
Y
待機？
40
N
Wカウント＝1
Y
命令が実行されるか？
N
Y
データパケットが生成されるか？
N
Wカウントがインクリメントされる
Y
Wカウント＝8？
N
Y
次のサイクルは分岐ファントムか？
N
Y
トレーシングが不能化されるか？
Y
N
フォーマット3のヘッダを出力する
トリガが発生するか？
N
Wカウント＝0
条件コードを満たすか？
Y
N
データパケットが生成されるか？
Y
N
Eカウントがインクリメントされる
N
Eカウント＝7？
Y
N
次のサイクルは分岐ファントムか？
Y
N
トレーシングが不能化されるか？
Y
N
次のサイクルで実行される命令がないか？
Y
N
トリガが発生するか？
Y
N
フォーマット1のヘッダを出力する
Eカウント＝0
サイクルアキュレート

【図５】


開始
42
トレースワードが受信されたか？
N
Y
44
命令トレースワードマスクをかけ、比較を実行する
データ
命令
46
命令トレースワードのオペランドフィールドを読み出す
データトレースワードを読み出す
50
48
命令動作アトムを再生する
終了

【図６】


202
204
206
208
CPU
RAM
ROM
NIC
222
HDD
表示装置
USER I/O
216
200
210
212
214
220
218

フロントページの続き

(72)発明者　アンドリュー　ブルックフィールド　スウェイン
イギリス国　ウェルウィン　ガーデン　シティ、ターモア　デイル　２９

審査官　多胡　滋

(56)参考文献　特開2002-149442（ＪＰ，Ａ）
特開2000-194581（ＪＰ，Ａ）
特開2000-163281（ＪＰ，Ａ）
特開2000-003295（ＪＰ，Ａ）
特開平11-353205（ＪＰ，Ａ）
特開平05-298144（ＪＰ，Ａ）
特開平03-025631（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　11/28
G06F　11/36