(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4094724号
(P4094724)
(45)発行日　平成20年6月4日(2008.6.4)　(24)登録日　平成20年3月14日(2008.3.14)

(51)Int.Cl.　F I
G06F 11/28 (2006.01)　G06F 11/28　D
G06F 11/28　315B
G06F 11/28　L

請求項の数 17　(全 25 頁)

(21)出願番号　特願平10-133256
(22)出願日　平成10年5月15日(1998.5.15)
(65)公開番号　特開平11-110254
(43)公開日　平成11年4月23日(1999.4.23)
審査請求日　平成16年12月16日(2004.12.16)
(31)優先権主張番号　9719173.8
(32)優先日　平成9年9月9日(1997.9.9)
(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】ソフトウェアをデバッグする際に例外を識別するための装置および方法

(57)【特許請求の範囲】
【請求項１】
各フィールドが特定の例外ルーチンに対応し、各フィールドが、対応する例外ルーチンへのアクセスの識別をデバッガーが望むことを表わすようにセット可能である、少なくとも１つのフィールドを有する制御レジスタと、
データ処理装置内のプロセッサコアが例外ルーチンのための命令フェッチコマンドを発生する時第１の信号を受信し、それにより該命令フェッチコマンドと共に発生される命令アドレスからどの例外ルーチンをフェッチしようとしているかを判定するように構成された例外ルーチンキャッチ論理回路とを備え、
該キャッチ論理回路が、判定された例外ルーチンに対応する該制御レジスタのフィールドを参照し、そのフィールドがセットされているかどうかを判断し、かつ該フィールドがセットされているとブレークポイント信号をプロセッサコアに出力するように構成された、データ処理装置とデバッガーを結合するためのデバッガーインターフェースユニット。
【請求項２】
前記第１信号は、実行すべき例外ルーチンのための命令フェッチ動作をリクエストするよう、プロセッサコアにより制御バスに出力される信号である、請求項１記載のデバッガーインターフェースユニット。
【請求項３】
前記第１信号は、命令フェッチ動作が求められていることを表わすよう、プロセッサコアによって出力される信号と命令フェッチ動作が例外ルーチンに関連することを表わすた

めの評価信号との両信号により構成される、請求項２記載のデバッガーインターフェースユニット。

【請求項４】

デバッガーが対応する例外ルーチンへのアクセスを識別したいかどうかを表わすための単一ビットを各フィールドが含む、請求項１記載のデバッガーインターフェースユニット。

【請求項５】

キャッチ論理回路が命令フェッチコマンドと共に発生される命令アドレスの所定ビットから命令フェッチコマンドが関連するのはどの例外ルーチンであるかを判断するように構成された、請求項１記載のデバッガーインターフェースユニット。

【請求項６】

プロセッサコアによって始動されようとしている命令アクセスまたはデータアクセスに関する情報を受信するための入力ターミナルと、

１つ以上のハードウェアブレークポイントユニットとを備え、各ブレークポイントユニットは、プロセッサコアにブレークポイント信号を送るようにする命令またはデータアクセスの属性を記憶するための少なくとも１つのブレークポイントレジスタ、およびブレークポイントレジスタ内の属性と入力ターミナルで受信された情報とを比較するためのブレークポイント比較器を有し、

各ブレークポイントユニットは、ブレークポイント比較器がブレークポイントレジスタ内の属性と入力ターミナルで受信された情報に含まれた対応する属性とが一致すると判断すると、前記ブレークポイント信号を発生するように構成されている請求項１記載のデバッガーインターフェースユニット。

【請求項７】

入力ターミナルがプロセッサコアによって始動されようとしている命令アクセスまたはデータアクセスに関する情報を受信するためのアドレスバス、データバスおよび制御バスに接続されている、請求項６記載のデバッガーインターフェースユニット。

【請求項８】

ブレークポイントユニットまたはステートマシンのいずれかにより発生されるブレークポイント信号を受信し、プロセッサコアに単一信号を出力するための論理ゲートを更に含む、請求項６記載のデバッガーインターフェースユニット。

【請求項９】

前記ブレークポイントレジスタ内に記憶するためのデバッガーからの属性を受信するためのインターフェースを更に含む、請求項６記載のデバッガーインターフェースユニット。

【請求項１０】

制御レジスタがステップ方法を使用すべきことを表わすようにセット可能な別のフィールドを有し、デバッガーインターフェースユニットが更に、

データ処理装置内のプロセッサコアが命令フェッチ動作を行うことをリクエストするとき第２の信号を受信するように構成されたステートマシンであって、プロセッサコアがリクエストした命令フェッチ動作の数の記録を維持するように構成されたステートマシンを備え、

該ステートマシンは更に制御レジスタを参照して別のフィールドがセットされているかどうかを判断し、かつその別フィールドがセットされていると、一旦プロセッサコアによって所定数の命令フェッチ動作がリクエストされると、プロセッサコアにブレークポイント信号を出力するように構成されている、請求項１記載のデバッガーインターフェースユニット。

【請求項１１】

命令フェッチ動作の所定数が２であり、ステップ方法が１回のステップ動作である、請求項１０記載のデバッガーインターフェースユニット。

【請求項１２】

前記第２信号は、命令フェッチ動作がリクエストされていることを示すための、プロセッサコアによって制御バスに出力される信号である、請求項１０記載のデバッガーインターフェースユニット。

【請求項１３】

命令を実行するためのプロセッサコアと、

請求項１に記載のデバッガーインターフェースユニットとを備えたデータ処理装置。

【請求項１４】

命令を記憶するための命令キャッシュと、

データを記憶するためのデータキャッシュと、

命令キャッシュをプロセッサコアに接続するように構成されている命令アドレスバスおよび命令データバスであって、プロセッサコアが命令キャッシュに命令アドレスを出力できるようにする命令アドレスバスおよびプロセッサコアが前記命令アドレスに対応する命令を受信することができるようにする命令データバスと、

データキャッシュをプロセッサコアに接続するように構成されているデータアドレスバスおよびデータデータバスであって、プロセッサコアがデータキャッシュにデータアドレスを出力できるようにする該データアドレスバスと、プロセッサコアが前記データアドレスに対応するデータ値を受信し、またはデータ値をデータキャッシュに出力できるようにする該データデータバスとを更に備えた、請求項１３記載のデータ処理装置。

【請求項１５】

デバッガーインターフェースユニットは、更に

プロセッサコアによって始動されようとしている命令アクセスまたはデータアクセスに関する情報を受信するための入力ターミナルと、

１つ以上のハードウェアブレークポイントユニットとを備え、

各ブレークポイントユニットがプロセッサコアにブレークポイント信号が送られるようにする命令またはデータアクセスの属性を記憶するための少なくとも１つのブレークポイントレジスタ、およびブレークポイントレジスタ内の属性と入力ターミナルで受信した情報とを比較するためのブレークポイント比較器を有し、

各ブレークポイントユニットは、ブレークポイント比較器がブレークポイントレジスタ内の属性と入力ターミナルで受信された情報に含まれる対応する属性とが一致すると判断すると、前記ブレークポイント信号を発生するように構成されており、ブレークポイント比較器がリクエストした命令アクセスに関して必要な情報を１つ以上のハードウェアブレークポイントユニットが受信できるように、デバッガーインターフェースユニットの入力ターミナルは命令アドレスバスおよび命令データバスに接続されている、請求項１４記載のデータ処理装置。

【請求項１６】

ブレークポイント比較器が要求したデータアクセスに関して必要な情報を１つ以上のハードウェアブレークポイントユニットが受信できるようにデバッガーインターフェースユニットの入力ターミナルがデータアドレスバスおよびデータデータバスに接続されている、請求項１５記載のデータ処理装置。

【請求項１７】

（ａ）各々が特定の例外ルーチンに対応する少なくとも１つのフィールドを制御レジスタ内に設ける工程と、

（ｂ）デバッガーが対応する例外ルーチンへのアクセスを識別したいことを表わすよう、前記フィールドのうちの１つをセットする工程と、

（ｃ）データ処理装置内のプロセッサコアが例外ルーチンのための命令フェッチコマンドを発生する時に、第１の信号を受信する工程と、

（ｄ）命令フェッチコマンドと共に発生される命令アドレスからどの例外ルーチンをフェッチしようとしているかを判断する工程と、

（ｅ）前記工程（ｄ）によって判断された例外ルーチンに対応する制御レジスタのフィールドがセットされている場合、プロセッサコアにブレークポイント信号を出力する工程

とを備えた、例外ルーチンへのアクセスを識別するようデータ処理装置とデバッガーを結合するためのデバッガーインターフェースユニットを作動させる方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ソフトウェアをデバッグ（バグ除去）するための装置および方法に関し、より詳細には、データ処理装置のためにソフトウェアをデバッグする際に、コードをパスするステップ動作を容易にする装置および方法に関する。

【０００２】

【従来技術】

代表的なデータ処理装置は、プロセッサコアに供給されるデータ値に適用される、ある命令シーケンスを実行するようになっているプロセッサコアを有することができる。一般に、プロセッサコアが必要とする命令およびデータ値を記憶するためのメモリを設けることができる。更に、メモリに必要なアクセス回数を低減するよう、プロセッサコアが必要とする命令およびデータ値を記憶するために、１つ以上のキャッシュが設けられるケースが多い。

【０００３】

かかるデータ処理装置によって実行されるソフトウェアをデバッグする際に、ソフトウェアを実行する際のデータ処理装置のアクティビティをモニタするよう、コンピュータ上で実行されるデバッガーアプリケーション（以下、デバッガーと総称する）を使用することが知られている。

【０００４】

このデバッガーはデータ処理装置内に設けられた、埋め込まれた「回路内エミュレータ」（ＩＣＥ）ユニットを介してデータ処理装置とインターフェースすることができるようになっている。一般に、かかる埋め込み型ＩＣＥユニットは数個のハードウェアブレークポイントユニットを含み、各ハードウェアブレークポイントユニットは識別時にデバッガーが関心を持つ命令またはデータアクセスの属性を記憶するための多数のブレークポイントレジスタと、プロセッサコアによって開始される命令またはデータアクセスに関するデータ処理装置から受信される情報とブレークポイントレジスタ内の属性とを比較するためのブレークポイント比較器とを有する。ブレークポイントレジスタは一般にデータ、アドレスまたは制御信号の属性をそれぞれ記憶するためのデータレジスタ、アドレスレジスタおよび制御レジスタを含む。更に、例えばアドレスの所定ビットに関心がない時に「ドントケア（don't care）」ステータスを設定できるようにするためのマスクフィールドを設けてもよい。

【０００５】

各ハードウェアブレークポイントユニットは、ブレークポイント比較器がブレークポイントレジスタの属性とデータ処理装置から受信される命令またはデータアクセスに関する情報に含まれる対応する属性とが一致すると判断した場合、プロセッサコアにブレークポイント信号を発生するようにできる。

【０００６】

ハードウェアブレークポイントユニットによって発生されるブレークポイント信号は、プロセッサコアが処理命令を停止するようにさせ、よってデバッガーがデータ処理装置のその時のステートを分析できるようにする。

【０００７】

かかるデータ処理装置によって実行されるソフトウェアをデバッグする際に、コードの実行中に例外が生じた場合、トラップできるようにすることは極めて好ましい。かかる例外の一例としては、データのアボート（data abort）、例えば存在していないメモリロケーションへのデータメモリのアクセスが行われる場合がある。この場合、メモリシステムはエラーを戻し、プロセッサはベクトルアドレスとして知られる例外アドレスを介して例外ルーチンとして知られる特別ルーチンへ分岐する。この例外ルーチンはデータアボートの

発生を取り扱うように設けられている。

【０００８】

かかる例外ルーチンへのアクセスを識別するための公知の技術は、ハードウェアブレークポイントユニットのうちの１つのブレークポイントレジスタに、その例外ルーチンの第１命令を識別する属性を含ませることである。このような方法によりプロセッサコアがその命令のアドレスを含む命令フェッチリクエストを発生すると、ハードウェアブレークポイントユニット内のブレークポイント比較器は命令フェッチリクエストとブレークポイントレジスタ内の属性との一致を判断し、一致している場合、ブレークポイント信号を発生する。

【０００９】

【発明が解決しようとする課題】

しかしながら上記方法には主な２つの不利点がある。第１に、ほとんどのマイクロプロセッサには多数の異なるタイプの例外があり、よってこの例外を処理するために多数の異なる例外ルーチンが設けられており、各例外ルーチンは異なる命令アドレスを有していることである。よって、一般に上記方法が例外ルーチンのうちのいずれかへのアクセスを効果的に識別する場合、ブレークポイントを必要とする異なる命令アドレスが多くなる。しかしながらコスト要素（例えばシリコンコスト）により埋め込み型ＩＣＥユニットは一般に少数のハードウェアブレークポイントユニットしか含まず、よってそれらの用途を例外ルーチンへのアクセスの識別に限定することは望ましいことではなく、この方法は当技術分野では、ベクトルトラッピング（Vector Trapping ）またはベクトルキャッチング（Vector Catching ）と称されることが多い。

【００１０】

例えばアドバンスＲＩＳＣマシンズリミテッドによって開発されたＡＲＭマイクロプロセッサは、８つのタイプの例外を有し、よって８つの例外ルーチンを有し、各例外ルーチンはユニークな命令アドレスを有するが、一般に２つのハードウェアブレークポイントしか有しない８つのタイプの例外はそれらのアドレスと共に下記のように示される（次表）。

【００１１】

【表１】

アドレス	例外ルーチン
０ｘ００００００００	リセット
０ｘ０００００００４	定義なし命令
０ｘ０００００００８	ソフトウェアによる割り込み
０ｘ０００００００Ｃ	プリフェッチアボート
０ｘ００００００１０	データアボート
０ｘ００００００１４	保　留
０ｘ００００００１８	インターラプト
０ｘ０００００１Ｃ	高速インターラプト

【００１２】
上記表から判るように、アドレスの所定のビットは各例外ルーチンに対して同じとなる。これら種々のタイプの例外のすべてを、単一のハードウェアブレークポイントユニットによって識別できるようにするには、一般にブレークポイントアドレスレジスタ内に「ドントケア」ステートをセットし、ハードウェアブレークポイントユニットが例外ルーチンのいずれかへのアクセスを識別できるように保証しなければならない。しかしながら、かかる「ドントケア」ステートを使用することは、必要以上に多い頻度でプロセッサコアが停止されることを意味する。
【００１３】
代替方法として、ソフトウェアのブレークポイント技術を使用する方法がある。このソフトウェアブレークポイント技術を使用する際は、ハードウェアブレークポイントユニットのブレークポイントデータレジスタ内に固定パターンのビットを入れ、ブレークポイント比較器が常に命令アドレスに関して受信した情報とその固定パターンとを比較する。次にメモリから各例外ルーチンの第１命令を取り出し、これを固定パターンのビットに置換する。この方法によれば、プロセッサコアが、そのアドレスへの命令フェッチリクエストを発生すると、固定パターンのビットが検索され、ハードウェアブレークポイントユニット内のブレークポイント比較器が、そのパターンのビットとブレークポイントレジスタに記憶されていたビットパターンとの一致を比較し、よってプロセッサコアにブレークポイント信号を発生する。このブレークポイント信号によりプロセッサコアは命令の処理を停止させられるので、デバッガーはプロセッサおよび／またはメモリのステートを検査できる。
【００１４】

しかしながらソフトウェアのブレークポイント方法を採用する場合、このことはＲＯＭのコードをデバッグすることは不可能であることを意味する。その理由はＲＯＭ内への命令は固定パターンのビットによりオーバーライトできないからである。更に、データ処理装置がキャッシュを使用している場合、および固定パターンのビットによりオーバーライトすべき命令がキャッシュ内にある場合、キャッシュ内のその命令をオーバーライトできないことがあり、よってキャッシュからその命令をフラッシュ（flush ）しなければならない。その理由は、メモリへのアクセスを行うことなく、よって固定パターンのビットを検索することなく、キャッシュから命令が戻されるからである。このことは、特にコードがキャッシュのロックダウン部分に記憶されている場合に不利であり、このロックダウン部分は一般に、決定可能な動き（すなわち決定可能な実行時間）を必要とするプロセッサコアおよび／またはルーチンによって定期的に使用され、よってメモリから検索された新しい命令によってオーバーライトすべきでない多数の命令を記憶するのに使用される。

【００１５】

上記技術の別の問題として、ある種のマイクロプロセッサは例外ルーチン、すなわち時々ベクトルと称される例外ルーチンのロケーションを変えることを認める点が挙げられる。例えばＭＩＰＳアーキテクチャは「ブートストラップ例外ベクトル」（ＢＥＶ）ビットのステートに基づき、ベクトルが位置を変えることを認めている。このＭＩＰＳアーキテクチャに関して更に情報を望む読者は、刊行物であるゲリー・ケイン著「ＭＩＰＳ　Ｒｉｓｃアーキテクチャ」（プレンティスホール出版、ＩＳＤＮ０－１３－５８４２９３－Ｘ）を参照されたい。ベクトルの基底はデバッガーが変更することなくソフトウェアの制御により変更し得る。このことは上記ブレークポイント技術を使用する場合、ベクトルへのアクセスはキャッチされないことを意味する。

【００１６】

従来の方法の上記課題を前提とし、本発明の目的はソフトウェアをデバッグする際に例外を識別するための改良された技術を提供することにある。

【００１７】

【課題を解決するための手段】

本発明の第１の態様によれば、各フィールドが特定の例外ルーチンに対応し、各フィールドが、対応する例外ルーチンへのアクセスの識別をデバッガーが望むことを表示するようにセット可能である、多数のフィールドを有する制御レジスタと、データ処理装置内のプロセッサコアが例外ルーチンのための命令フェッチコマンドを発生する時に第１信号を受信し、命令フェッチコマンドと共に発生される命令アドレスからどの例外ルーチンをフェッチするかを判断するようになっている例外ルーチンキャッチ論理回路とを備え、該キャッチ論理回路が決定された例外ルーチンに対応する制御レジスタのフィールドを参照し、そのフィールドがセットされているかどうかを判断し、フィールドがセットされている場合にブレークポイント信号をプロセッサコアに出力するようになっている、データ処理装置のためのデバッガーインターフェースユニットが提供される。

【００１８】

本発明によれば、デバッガーインターフェースまたは先に述べたような埋め込み型ＩＣＥユニットは、ソフトウェアをデバッグする際の例外の識別を容易にするための別の要素を含むようになっている。これら別の要素としてはデータ処理装置内のプロセッサコアが例外ルーチンのための命令フェッチコマンドを発生する際に、第１信号を受信するようになっている例外ルーチンキャッチ論理回路が挙げられる。更にデバッガーインターフェースの制御レジスタ内には多数のフィールドが設けられ、各フィールドは特定の例外ルーチンに対応し、各フィールドはデバッガーが対応する例外ルーチンへのアクセスの識別を望むことを表示するようにセット可能である。この例外ルーチンキャッチ論理回路は命令フェッチコマンドと共に発生される命令アドレスおよび制御レジスタのフィールド内に記憶された情報を使用して、どの例外ルーチンにフェッチしているかを判断し、対応するフィールドがセットされている場合にプロセッサコアにブレークポイント信号を出力する。

【００１９】

かかる例外ルーチンキャッチ論理回路を設けたことにより、他の目的に使用するためにデバッガーインターフェースユニット内に設けられる一般的なハードウェアブレークポイントユニットが不要となる。その理由は、例外ルーチンへのアクセスを識別するのにハードウェアブレークポイントユニットを保留する必要がないからである。更に、かかる例外ルーチンへのアクセスの識別は、例外ルーチンのベースアドレスと無関係である。

【００２０】

好ましい実施例では、第１信号は実行すべき例外ルーチンのための命令フェッチをリクエストするよう、プロセッサコアにより制御バスに出力される信号であり、好ましくは第１信号は命令フェッチが求められていることを表示するよう、プロセッサコアによって出力される信号（以下InMREQ信号と称す）と、命令フェッチが例外ルーチンに関連することを表示するための評価信号とから成る。代表的なデバッガーインターフェース、例えば埋め込み型ＩＣＥユニットは既にInMREQ信号を受信するようになっているので、デバッガーインターフェースユニットは評価信号を受信するようになっているだけでよい。

【００２１】

当業者であれば制御レジスタ内のフィールドのサイズを任意に選択し、デバッガーインターフェースユニット内で充分なスペースが利用できると見なすことができる。しかしながら好ましい実施例では、各フィールドはデバッガーが対応する例外ルーチンへのアクセスを識別したいかどうかを表示するための単一ビットを含む。これにより制御レジスタ内のスペースを最も効率的に使用できる。

【００２２】

キャッチ論理回路は命令フェッチコマンドがどのルーチンに関連しているかを判断するように、命令フェッチコマンドと共に発生される命令アドレス全体を検査するようにできる。しかしながら好ましい実施例では、キャッチ論理回路は命令フェッチコマンドと共に発生される命令アドレスの所定ビットから命令フェッチコマンドが関連するのはどの例外ルーチンであるかを判断するようになっている。どの例外ルーチンにアクセスしているかを判断するのに、命令アドレスのビットの所定のサブセットしか必要でない状況では、上記方法は命令アドレス全体を検査するよりも、より効率的である。

【００２３】

好ましい実施例では、デバッガーインターフェースユニットは、プロセッサコアによって開始される命令またはデータアクセスに関する情報を受信するための入力ターミナルと、１つ以上のハードウェアブレークポイントユニットとを備え、各ブレークポイントユニットがプロセッサコアにブレークポイントユニット信号を送るようにする命令またはデータアクセスの属性を記憶するための多数のブレークポイントレジスタ、およびブレークポイントレジスタ内の属性と入力ターミナルで受信された情報とを比較するためのブレークポイント比較器と有し、

ブレークポイント比較器がブレークポイントレジスタ内の属性と入力ターミナルで受信された情報に含まれた対応する属性とが一致すると判断した場合、各ブレークポイントユニットが前記ブレークポイント信号を発生するようになっている。

【００２４】

先に述べたように、所定の事象が生じた時に、かかるハードウェアブレークポイントユニットを使用してブレークポイント信号を発生させることができる。制御バスに出力されるアドレス、命令、データ値または信号の特定の属性を記憶する外に、ブレークポイントレジスタは所定の属性をマスクし、実際に「ドントケア」機能を提供するようにマスク情報を記憶することもできる。従って、ブレークポイント比較器が戻されたデータ値の所定部分と関連すべきでなく、むしろブレークポイント比較器は比較のための他の属性、例えばプロセッサコアが発生する命令またはデータ値のアドレスを使用することを表示するように、ブレークポイントレジスタにデータマスクを加えることができる。

【００２５】

先に述べたように、例外ルーチンへのアクセスの識別を行うのに、キャッチ論理回路を使用することにより、他の目的のために設けられたかかるハードウェアブレークポイントユ

ニットを自由に使用でき、例外ルーチンのアクセスを識別するのにハードウェアブレークポイントを保留する必要がなくなる。

【００２６】

好ましい実施例では、入力ターミナルはプロセッサコアによって開始される命令またはデータアクセスに関する情報を受信するためのアドレスバス、データバスおよび制御バスに接続されている。ブレークポイントレジスタ内に記憶される属性はアドレス、（命令データを含む）データまたは制御バスに出力される信号に対する属性に関連付けでき、よって好ましい実施例では入力ターミナルはハードウェアブレークポイントユニットが必要な情報すべてにアクセスするのに、アドレスバス、データバスおよび制御バスからの情報を受信する必要がある。

【００２７】

好ましい実施例では、デバッガーインターフェースユニットは、ブレークポイントユニットまたはステートマシンのいずれかにより発生されるブレークポイント信号を受信し、プロセッサコアに単一信号を出力するための論理ゲートを更に含む。ブレークポイント信号が論理「１」の値をとる場合、論理ゲートはＯＲゲートであることが好ましい。

更に好ましい実施例では、デバッガーインターフェースユニットは、デバッガーからの属性を受信し、ブレークポイントレジスタ内に記憶するためのインターフェースを更に含む。このインターフェースはシリアルインターフェースの形態、例えばＩＥＥＥ規格１１４９．１－１９９０ＪＴＡＧシリアルインターフェースであることが好ましく、よってデバッガーインターフェースはブレークポイントレジスタに加えるべきデータ内でスキャンできる。

【００２８】

好ましい実施例では、制御レジスタはステップ方法を使用すべきことを表示するようにセット可能な別のフィールドを有し、デバッガーインターフェースは更に、データ処理装置内のプロセッサコアが命令フェッチを行うことをリクエストした際に、第１信号を受信するようになっており、プロセッサコアがリクエストした命令フェッチの数の記録を維持するようになっているステートマシンを備え、該ステートマシンが更に制御レジスタを参照し、フィールドがセットされているかどうかを判断し、フィールドがセットされている場合において、プロセッサコアによって所定の数の命令フェッチがリクエストされると、プロセッサコアにブレークポイント信号を出力するようになっている。

【００２９】

従来のデバッガーインターフェースユニットでは、かかるステップ方法はハードウェアブレークポイントユニットのうちの１つを使用することによって、一般に実行される。しかしながら上記のようなステートマシンを設けたことにより、他の目的のために使用すべきハードウェアブレークポイントユニットが不要となる。その理由は、ステップ方法を実行するのにハードウェアブレークポイントユニットのうちの１つを保留する必要がないからである。

【００３０】

好ましくは、命令フェッチの所定の数は２であり、ステップ方法は１回のステップ動作である。代表的なデバッガーインターフェースユニット、例えば埋め込み型ＩＣＥユニットは、命令フェッチが生じたことを表示する信号に応答できる。従って、ステートマシンはそれら信号の記録を維持するだけでよく、好ましい実施例によればこれら信号のうちの第２信号を受信すると、ブレークポイント信号がアサートされる。このような方法により、次の命令が実行され、プロセッサはデバッグステートになった後に他のことを行う。

【００３１】

第２信号は、実行すべき命令フェッチをリクエストするための、プロセッサコアによって制御バスに出力される信号（以下、InMREQ信号と称す）である。上記のように代表的なデバッガーインターフェースユニット、例えば埋め込み型ＩＣＥユニットは既にInMREQ信号を受信するようになっており、よってステートマシンは受信したInMREQ信号の数の記録を維持するだけよい。

【００３２】
本発明の第２の態様によれば、命令を実行するためのプロセッサコアと、本発明の第１の態様に係わるデバッガーインターフェースユニットとを備える。
【００３３】
本発明の第２の態様によれば、デバッガーインターフェースユニットおよびプロセッサコアは、１つの回路内に組み合わされ、この回路は他の要素、例えばメモリまたはキャッシュと別個に設けることができる。
【００３４】
データ処理装置は多数の他の特徴、例えばプロセッサコアによって直接アクセスされるメモリ、または１つ以上のキャッシュと組み合わせて使用されるメモリを含むことができる。フォンノイマンアーキテクチャによれば、命令およびデータの双方を記憶するために、単一のキャッシュを設けることができる。しかしながら、好ましい実施例ではハーバードアーキテクチャが使用され、従って、データ処理装置は更に、命令を記憶するための命令キャッシュと、データを記憶するためのデータキャッシュと、命令キャッシュをプロセッサコアに接続するようになっており、プロセッサコアが命令キャッシュに命令アドレスを出力できるようにする命令アドレスバスおよびプロセッサコアが前記命令アドレスに対応する命令を受信することができるようにする命令データバスと、データキャッシュをプロセッサコアに接続するようになっており、プロセッサコアがデータキャッシュにデータアドレスを出力できるようにするデータアドレスバスおよびプロセッサコアが前記データアドレスに対応するデータ値を受信し、またはデータ値をデータキャッシュに出力できるようにするデータデータバスとを更に備える。
【００３５】
好ましくはデバッガーインターフェースユニットの入力ターミナルは、命令アドレスバスおよび命令データバスに接続され、ブレークポイント比較器がリクエストした命令アクセスに関する必要な情報を１つ以上のハードウェアブレークポイントユニットが受信することができるようにしている。更に、デバッガーインターフェースユニットの入力ターミナルはデータアドレスバスおよびデータデータバスに接続され、ブレークポイント比較器が要求したデータアクセスに関する必要な情報を１つ以上のハードウェアブレークポイントユニットが受信することができるようにしている。
【００３６】
更に、入力ターミナルは命令制御バスおよびデータ制御バスの双方に接続され、これら制御バス上に出力される信号に関する、ハードウェアブレークポイントユニットが求めることがある必要な情報を受信できる。
【００３７】
本発明の第３の態様によれば、（ａ）各々が特定の例外ルーチンに対応する多数のフィールドを制御レジスタ内に設ける工程と、（ｂ）デバッガーが対応する例外ルーチンへのアクセスを識別したことを表示するよう、前記フィールドのうちの１つをセットする工程と、（ｃ）データ処理装置内のプロセッサコアが例外ルーチンのための命令フェッチコマンドを発生する時に、第１信号を受信する工程と、（ｄ）命令フェッチコマンドと共に発生される命令アドレスからどの例外ルーチンをフェッチしているかを判断する工程と、（ｅ）前記工程（ｄ）によって判断された例外ルーチンに対応する制御レジスタのフィールドがセットされている場合、プロセッサコアにブレークポイント信号を出力する工程とを備えた、例外ルーチンへのアクセスを識別するようデータ処理装置のためにデバッガーインターフェースユニットを作動させる方法が提供される。
【００３８】
添付図面に示した本発明の好ましい実施例を参照して、単なる例により本発明について更に説明する。
【００３９】
【発明の実施の形態】
図１は、本発明の一実施例に係わるデータ処理装置を示し、この実施例ではデータ処理装

置は別個の命令キャッシュと別個のデータキャッシュを有するハーバードアーキテクチャを使用している。

【００４０】

図１に示されるように、プロセッサコア１００は命令バスラインを介して命令キャッシュ１１０に接続され、かつデータバスラインを介してデータキャッシュ１２０に接続されている。命令キャッシュ１１０とデータキャッシュ１２０の双方を制御するための単一キャッシュコントローラ１３０が示されている。しかしながらこれとは異なり、一つのコントローラが命令キャッシュ１１０用であり、一つのコントローラがデータキャッシュ１２０用であるような別個のキャッシュコントローラを設けてもよい。

【００４１】

プロセッサコア１１０が命令を求めるとき、このコア１１０は命令アドレスバス１６０にその命令のメモリアドレスを発生し、更に命令制御バス１７０上にプロセッサの制御信号を発生する。このプロセッサ制御信号はアドレスが読み出しリクエストまたは書き込みリクエストのいずれに対応するのか、アクセスのタイプ（例えばシーケンシャルタイプ）、アクセスのサイズ（例えばワード、バイト）、プロセッサの作動モード（例えばスーパーバイザーまたはユーザーモード）等に関する情報を含む。このプロセッサ制御信号はキャッシュコントローラ１３０によって受信され、必要な命令を命令キャッシュ１１０に記憶するかどうかをキャッシュコントローラに判断させるように促す。

【００４２】

好ましい実施例ではプロセッサコア１００が検索すべき命令を求めた時に、命令アドレスバス１６０に発生されたプロセッサ制御信号は命令メモリリクエスト（InMREQ）信号であり、この信号は命令キャッシュ１１０が命令アドレスバス１６０上のアドレスと命令キャッシュ１１０内のアドレスとを比較し、キャッシュ内にそのアドレスに対応する命令が記憶されているかどうかを判断しなければならないことを、キャッシュコントローラ１３０に通知する。記憶すると判断した場合、この命令は命令キャッシュ１１０から命令データバス１６５に出力され、ここでプロセッサコア１００によってこの命令が読み出される。アドレスに対応する命令がキャッシュ１１０にない場合、キャッシュコントローラ１３０は外部バス２２０を介してメモリ１４０から命令を検索させる。次に、検索された命令はキャッシュ１１０内に記憶され、次に命令データバス１６５を通してプロセッサコア１００へ戻される。

【００４３】

同様にデータ値のためのアドレスをデータアドレスバス１８０上に出力することができ、データフェッチが必要なことをメモリシステムに表示するための信号（以下、データメモリリクエスト（DnMREQ）信号と称す）がデータ制御バス１９０を通してキャッシュコントローラ１３０に出力される。これによりデータキャッシュ１２０はデータ値がデータキャッシュ内に記憶されている場合、データデータバス１８５上のそのアドレスに対応するデータ値を戻すようにさせるか、またはキャッシュコントローラ１３０はメモリ１４０からデータ値を検索し、この点でデータ値がデータキャッシュ１２０に記憶され、データデータバス１８５を通してプロセッサコア１００に戻される。

【００４４】

本発明の好ましい実施例によれば、データ処理装置には更に埋め込み型ＩＣＥユニット１５０が設けられ、このユニットはコンピュータ、例えばパソコンで実行される外部デバッガーアプリケーションに対するデバッガーインターフェースユニットとして使用される。このデバッガーアプリケーションを以下、デバッガーと称す。好ましい実施例では埋め込み型ＩＣＥユニット１５０は２つのハードウェアブレークポイントユニット２００および２０５を含み、各ハードウェアブレークポイントユニットはデバッガーが識別に関心を有する命令またはデータアクセスの属性を記憶するための多数のブレークポイントレジスタおよびブレークポイントレジスタ内の属性とプロセッサコアによって開始される命令またはデータアクセスに関する、データ処理装置から受信される情報とを比較するためのブレークポイント比較器を有する。データ処理装置からのこの情報は、種々の命令バスライン

１６０、１６５、１７０および／または種々のデータバスライン１８０、１８５、１９０を通して得られる。各ハードウェアブレークポイントユニットはブレークポイント比較器がブレークポイントレジスタの属性とデータ処理装置から受信される命令またはデータに関する情報に含まれる対応する属性とが一致すると判断した場合、プロセッサコアにブレークポイント信号を発生することができる。

【００４５】

デバッガーがハードウェアブレークポイント２００または２０５内のブレークポイントレジスタを更新できるようにするためにシリアルインターフェース、例えばＪＴＡＧシリアルインターフェースを設け、デバッガーインターフェースユニットがブレークポイントレジスタに対かすべきデータ内でスキャンできるようにする。ブレークポイントレジスタ内に記憶すべき新しい属性をデバッガーインターフェースユニットに与えることができ、これらパラメータはインターフェースをブレークポイント比較器によってその後使用できるようにブレークポイントレジスタに記憶される。

【００４６】

各ブレークポイント比較器は、任意の時間において命令バス１６０、１６５、１７０からの命令情報またはデータバス１８０、１８５、１９０からのデータ情報のいずれかを見ることができるようになっている。ブレークポイント比較器が命令バスからの命令情報を分析し、これがブレークポイントレジスタの属性と一致していると判断した場合、パス２３０を介してプロセッサコアにブレークポイント信号が送られる。しかしながらブレークポイント比較器がデータバス１８０、１８５、１９０からのデータ情報を解析し、ブレークポイントレジスタの属性と一致していると判断した場合、パス２４０を介してプロセッサコア１００にウォッチポイント信号が送られる。好ましい実施例の次の説明のために、パス２３０を通して発生されるブレークポイント信号およびパス２４０を通して発生されるウォッチポイント信号の双方が、プロセッサコアの命令処理をするのを停止させ、よってデバッガーがブレークポイント命令の実行に先立ち、データ処理装置の現在ステートをデバッガーが分析できるようにする際に、これら双方の信号をブレークポイント信号と見なすことができる。しかしながら好ましい実施例では、受信された信号が命令に関するブレークポイント信号であるか、またはデータに関するウォッチポイント信号であるかを判断できるように、別個のパス２３０、２４０が設けられている。

【００４７】

プロセッサコア１００がパス２３０または２４０を介してブレークポイント信号を受信し、命令処理を停止するとプロセッサコアはパス２５０を通して埋め込み型ＩＣＥユニット１５０にデバッグアクノーレッジ信号を送り、プロセッサコアがブレークポイント信号を受信し、命令の処理を停止したことを埋め込み型ＩＣＥユニット１５０に伝える。

埋め込み型ＩＣＥユニット１５０はハードウェアブレークポイントユニット２００および２０５の他に埋め込み型ＩＣＥユニット１５０の作動を制御するために使用されるデータの種々のアイテムを含む多数の制御レジスタ２１０も含む。しかしながら本発明の好ましい実施例によれば、制御レジスタ２１０に多数のエクストラフィールドが追加されており、これら制御レジスタは所定の指定事象が生じた際に、ブレークポイント信号の発生を取り扱うよう、埋め込み型ＩＣＥユニットに加えられた別の要素によって使用される。これら別の要素は図４に示されており、図４は埋め込み型ＩＣＥユニット１５０内の構成部品のより詳細な概観を示すものである。

【００４８】

図４に示されるように、データ処理装置で実行中のコードをデバッガーが１ステップごとに実行できるようにし、よって所定の数の命令を実行し、次にプロセッサおよび／またはメモリのステートを検査し、その後、次の所定の数の命令を実行するようにするためのステートマシン３００が設けられている。好ましい実施例では、コードを通過する１回のステップ動作を実行するのに、このステートマシン３００が使用され、よって一度に１つの命令でコードが実行され、各命令の間でプロセッサおよび／またはメモリのステートが検査される。ステートマシン３００によってアクセス可能であり、かつステップ方法を使用

すべきことを表示するように設定可能な命令レジスタ２１０内にエキストラフィールドが設けられている。このフィールドがセットされている場合、ステートマシンはプロセッサコアによって発生されるInMREQ信号の数の記録を維持するようになっており、所定の数のInMREQ信号が受信された後、ステートマシン３００はパス３３０を通してＯＲゲート３２０にブレークポイント信号を発生するようになっている。ステートマシン３００が一回のステップ動作を実行するようになっている場合、ステートマシン３００はプロセッサコアによって２つのInMREQ信号が発行されると、ステートマシン３００はブレークポイント信号を１回発生する。このステートマシン３００の作動については図５～７を参照して後により詳細に説明する。

【００４９】

本発明の好ましい実施例によれば、埋め込み型ＩＣＥユニット１５０はベクトルアドレスとも称されることが多い例外アドレスにて、例外ルーチンへのアクセスを識別するようになっているベクトルキャッチ論理回路３１０も含む。制御レジスタ２１０には多数の別のフィールドが追加されており、各フィールドは特定の例外ルーチンに対応し、各フィールドはデバッガーが対応する例外ルーチンへのアクセスを識別したがっていることを表示するように設定できる。

【００５０】

ベクトルキャッチ論理回路３１０はプロセッサコアによって発生されるInMREQ信号を受信するようになっており、更にInMREQ信号と称される命令フェッチリクエストが実際に例外ルーチンの命令のための命令フェッチリクエストであること表示するようにセットされた評価信号を受信するようにもなっている。ロジックキャッチ論理回路３１０はこれら信号を受信すると、InMREQ信号に関連するアドレス内の所定の数のビットおよびアドレスが、どの例外ルーチンに対応するかを判断するよう、各例外ルーチンに対応する種々の別の制御レジスタ内に記憶されたビットを使用するようになっている。命令フェッチコマンドが例外ルーチンに対応し、この例外ルーチンの、制御レジスタ内の対応するフィールドがセットされている場合、ベクトルキャッチ論理回路３１０はパス３４０を通してＯＲゲート３２０にブレークポイント信号を発生するようになっている。ベクトルキャッチ論理回路３１０の作動については図８～１０を参照して後により詳細に説明する。

【００５１】

ＯＲゲート３２０はステートマシン３００およびベクトルキャッチ論理回路３１０によって発生されるブレークポイント信号の他にＯＲゲート３２０のハードウェアブレークポイントユニット２００および２０５からパス３５０および３６０を通って発生されるブレークポイント信号を受信するようになっている。従って、埋め込み型ＩＣＥユニット１５０の部品によってブレークポイント信号が発生されると、ＯＲゲート３２０はパス３７０を通してプロセッサコア１００にブレークポイント信号を発生する。

ステートマシン３００およびベクトルキャッチ論理回路３１０は特に１回のステップおよび例外ルーチンへのアクセスの識別を特に処理するように設けられているので、ハードウェアブレークポイントユニット２００および２０５は他の目的のために自由に使用できる。よってデバッガーは一部の別のデバッグ方法を実行しながら、１回のステップまたはベクトルキャッチプロセスも実行できる。

【００５２】

当業者であれば、本発明の好ましい実施例に係わる上記埋め込み型ＩＣＥユニット１５０は、図１に示されるようなハーバードアーキテクチャを有するデータ処理装置と共に使用することだけに限定されているわけではないことが理解できよう。必要なことは、埋め込み型ＩＣＥユニット１５０がプロセッサコアによって開始される命令アクセスおよび／またはデータアクセスに関する情報を受信するようにすることである。

【００５３】

従って、図２に示されるように、好ましい実施例の埋め込み型ＩＣＥユニット１５０は命令およびデータ値の双方を記憶するための単一キャッシュ４００が設けられているフォンノイマンアーキテクチャを有するデータ処理装置でも使用できる。このような構造では、

プロセッサコア１００が命令またはデータ値を必要とする時には、このプロセッサコアはプロセッサのアドレス（ＰＡ）バス４１０に、その命令またはデータ値のメモリアドレスを入れる。更に、プロセッサコア１００はプロセッサ制御（ＰＣ）バス４２０にプロセッサ制御信号を発生する。このプロセッサ制御信号はキャッシュコントローラ１３０によって受信され、この制御信号は必要な命令またはデータ値がキャッシュ４００内に記憶されているかどうかをキャッシュコントローラが判断するように促す。更に、命令フェッチとデータフェッチとを区別する別の制御信号が設けられている。キャッシュコントローラ１３０はアドレスバス４１０上のアドレスとキャッシュ内のアドレスとを比較し、そのアドレスに対応する命令またはデータ値がキャッシュ内に記憶されているかどうかを判断するように、キャッシュ４００に命令する。記憶されている場合、その命令またはデータ値はキャッシュ４００からプロセッサデータ（ＰＤ）バス４１５に出力され、このバスからプロセッサコア１００によって読み出される。そのアドレスに対応する命令またはデータ値がキャッシュ４００内にない場合、キャッシュコントローラ１３０はその命令またはデータ値が外部バス２２０を通してメモリ１４０から検索されるようにする。次に、検索された命令またはデータ値はキャッシュ４００に記憶され、データバス４１５を通してプロセッサコア１００へ戻される。

【００５４】

プロセッサコア１００は、プロセッサデータバス４１５を通して命令またはデータ値を受信する以外に、データバス４１５を通してキャッシュに記憶できるよう、キャッシュ４００へデータ値を出力することもできる。これらデータ値は後の処理操作に必要とされる時にプロセッサコア１００によってその後、検索できる。

【００５５】

埋め込み型ＩＣＥユニット１５０は、図１および４を参照してこれまで説明したのと同様に作動する。従ってこのユニットは、種々のバスライン４１０、４１５、４２０に接続され、プロセッサコア１００によって開始される命令またはデータアクセスに関する必要な情報を受信する。単一ステップ動作またはベクトルキャッチングのために、埋め込み型ＩＣＥユニットはデータフェッチではなく、命令フェッチを考慮するだけである。図４を参照してこれまで説明したように、ステートマシン３００を使用してデバッガーが単一のステップ動作を実行できるようにし、ベクトルキャッチ論理回路３１０を使用して例外ルーチンへのアクセスが生じたことを識別し、プロセッサコアによって実行される処理を中止するように、プロセッサコアにブレークポイント信号を発生し、このポイントにおけるプロセッサコアおよび／またはメモリのステートをデバッガーが分析できるようにする。パス３７０を通してプロセッサコアにブレークポイント信号が発生される時、プロセッサコア１００はパス２５０を通して埋め込み型ＩＣＥユニット１５０にデバッグアクノーレッジ信号を送り、その処理を中止したことを確認する。

【００５６】

好ましい実施例の埋め込み型ＩＣＥユニット１５０は、ハーバードアーキテクチャまたはフォンノイマンアーキテクチャのいずれかを有するデータ処理装置と共に使用する他に、キャッシュを有しないデータ処理装置、例えば図３に示されているようなデータ処理装置と共に使用することも可能である。図３に示されているように、このプロセッサコア１００はプロセッサバスライン４５０、４５５、４６０を介してメモリ１４０にプロセッサコア１００が接続されている。プロセッサコア１００が命令またはデータ値を必要とすると、プロセッサコアはその命令またはデータ値のメモリアドレスをプロセッサアドレス（ＰＡ）バス４５０に出力し、更にプロセッサ制御（ＰＣ）バス４６０にプロセッサ制御信号を発生する。このプロセッサ制御信号はメモリ１４０によって受信され、ＰＡバス４５０に出力されたアドレスに記憶された命令またはデータ値をメモリが検索するように促す。命令フェッチとデータフェッチとを区別する別の制御信号も更に設けられている。検索された命令またはデータ値はメモリ１４０からプロセッサデータ（ＰＤ）バス４５５に出力され、このバスからプロセッサコア１００によって読み出される。プロセッサコア１００はＰＤバス４５５を通して命令またはデータ値を受信する他に、ＰＤバス４５５を通して

メモリ１４０にデータ値を出力し、メモリに記憶できるようにすることも可能である。
【００５７】
埋め込み型ＩＣＥユニット１５０は、命令またはデータアクセスに関する必要な情報を受信するように種々のプロセッサバス４５０、４５５、４６０に接続されており、図１、２および４を参照して先に説明したのと同じように作動する。以上で本発明の好ましい実施例に係わる埋め込み型ＩＣＥユニット１５０の一般的な構造について説明したので、次に図５～７を参照して埋め込み型ＩＣＥユニット１５０内のステートマシン３００の作動についてより詳細に説明する。
【００５８】
図５は、単一ステップ作動モードで作動する際に好ましい実施例のデータ処理装置によって発生される多数の信号の間の関係を示すタイミング図である。各クロックサイクルの間が分離されていることを示すために垂直線５２０が加えられている。これら種々のクロックサイクルの間のステートマシン３００のステート（ＳＭ　ＳＴＡＴＥ）は、図５の底部に表示されており、Ｉはステートマシンがアイドルステートにあり、Ｆ１はステートマシン３００が第１フェッチステートにあり、Ｆ２はステートマシン３００が第２フェッチステートにあることを示す。図６にはこれらステートおよびステート間の関係が示されている。
【００５９】
第１クロックサイクルＦＥ１の間、プロセッサクロックは第１命令をフェッチするようになっている。従って、次のクロックサイクルの間のクロック信号５００の立ち上がりエッジでは、プロセッサコアは命令フェッチを開始するために論理「０」値を有するInMREQ信号５１０を発生し、同時に命令バスにその命令のアドレスを出力するようになっている。ＦＥ１クロックサイクルにおけるクロックの立ち上がりエッジでは、ステートマシン３００はプロセッサによって制御バスに第１InMREQ信号が発生されたことを通知し、従って、アイドルステートを脱出し、Ｆ１ステートに移行する。ＦＥ１クロックサイクルの終了時に第１命令フェッチリクエストに対応する命令データ５３０は、命令データ（ＩＤ）バスに戻される。
【００６０】
次にクロックサイクルでは、プロセッサは（図５における参照符号ＤＥ１によって表示される）第１命令をデコードするようにされ、（図５における参照符号ＦＥ２によって表示される）メモリまたはキャッシュからの第２命令もフェッチする。ＦＥ１クロックサイクルにおけるクロックの立ち上がりエッジとＦＥ２クロックサイクルにおけるクロックの立ち上がりエッジとの間で、制御バス上にInMREQ信号がアサートされ、この命令の対応するアドレスは命令バス上に出力される。更にＦＥ２クロックサイクルにおけるクロックの対応するエッジではステートマシン３００は第２InMREQ信号が発生されたことを認識し、よってステートマシンはＦ１ステートからＦ２ステートへ移行する。このようなＦＥ２クロックサイクルの終了時に第２命令リクエストに対応する命令データ５３５が命令データバスを通してプロセッサコアに戻される。
【００６１】
更に、ＦＥ２クロックサイクルにおけるクロックの立ち上がりエッジにて、ステートマシン３００は第２InMREQ信号が発生されていることを認識し、よって第２命令フェッチが行われたことを認識しているので、ステートマシン３００はＦＥ２クロックサイクルのクロックの立下りエッジで、ブレークポイント信号５４０をアサートするようになっている。ＦＥ２クロックサイクルにおけるクロックの立下りエッジとブレークポイント信号５４０の発生との関係は、図５における矢印５６０によって表示されている。
【００６２】
次のクロックサイクルでは、プロセッサ１００は（図５における参照符号ＥＸ１によって表示される）第１命令を実行し、更に（図５における参照符号ＤＥ２によって表示される）第２命令をデコードするようになっている。更にプロセッサは、メモリまたはキャッシュから第３命令をフェッチし、よって第３InMREQ信号がアサートされる。このクロックサ

イクルにおけるクロックの立ち上がりエッジでは、ステートマシン３００は第３InMREQ信号が発生されたことを認識し、更にプロセッサコアから埋め込み型ＩＣＥユニット１５０へのデバッグアクノーレッジ信号５５０がまだ低レベルであることも認識する。このことは、プロセッサコア１００がまだ命令処理を停止していないことを示している。このような状況では、ステートマシンは図６に示されるようなＦ２ステートのままである。この理由は、プロセッサが一旦停止すると、ステートマシンはアイドルステートに戻るにすぎないからである。

【００６３】

ステートマシン３００は、まだ論理「１」レベルのデバッグアクノーレッジ信号５５０を受信していないので、このクロックサイクルの終了時に次のクロックサイクルにおけるブレークポイント信号５４０を再アサートするようになっている。第１命令は、例えば分岐条件となることがあり、よって第１命令の直後に第２命令を実行すると自動的にみなすことができないので、デバッグアクノーレッジ信号５５０がアサートされるまでフェッチされるその後の各命令に対し、ブレークポイント信号は再アサートされることが好ましい。プロセッサがブレークポイントされた命令を処理するまで、すなわちデバッグアクノーレッジ信号がプロセッサによってアサートされるポイントにて、パイプラインの実行ステージに命令が到達するまで、命令はフェッチされ続ける。

【００６４】

図５を参照すると、次のクロックサイクルにおいて、プロセッサコアは第２命令を実行しており、ブレークポイント信号５４０を受信するので、プロセッサコアは命令の処理を停止し、この次のクロックサイクルにおけるクロックの立ち上がりエッジでデバッグアクノーレッジ信号５５０を発生するようになっている。従って、先のクロックサイクルでフェッチされ、デコードされた第２命令は、このクロックサイクルでは実行されず、従って、このクロックサイクルにおけるクロックの立ち上がりエッジにてステートマシンはプロセッサが停止したことを表示するデバッグアクノーレッジ信号５５０の存在を認識し、よってアイドルステートに戻る。次に、このクロックサイクルの立下りエッジにてデータ処理装置はデバッグステートに入り、このデバッグステートではInMREQ信号は命令フェッチが行われないことを表示するハイレベルに留まる。プロセッサコアはブレークポイント信号に応答しており、よって、もうブレークポイント信号を発生する必要はないので、ブレークポイント信号５４０は低レベルのままであり、デバッグアクノーレッジ信号５５０はプロセッサが停止したことを表示する高レベルのままである。

【００６５】

デバッガーがプロセッサコアおよび／またはメモリのステートの分析を終了すると、クロックはオンに戻るように切り換えられ、デバッグアクノーレッジ信号５５０はクロックの第１の立ち上がりエッジで論理「０」レベルに戻る。更にプロセッサコアは、第２命令が再フェッチされることを求めるためのInMREQ信号５１０を発生する。図５に示されるように、第２命令に対し、単一ステッププロセス全体が繰り返され、この結果、第２命令が実行されると、データ処理装置は再びデバッグステートになる。

【００６６】

図６は、図５を参照してこれまで説明したステートマシンのステートの概要を示すものである。従って、ステートマシン３００は第１命令フェッチが行われるまでアイドルステート６００のままであり、第１命令フェッチが生じた時点でステートはＦ１ステート６１０に変わる。次に第２命令フェッチが生じると、ステートマシンはＦ１ステートからＦ２ステート６２０に移行する。ステートマシン３００はプロセッサが停止したことを表示するデバッグアクノーレッジ信号５５０が受信されるまで、Ｆ２ステート６２０のままであり、デバッグアクノーレッジ信号が受信された時点でステートマシンはＦ２ステート６２０からアイドルステート６００に戻る。

【００６７】

図７にはステートマシン３００の機能を行うための好ましい実施例で使用される回路が示されている。論理回路７００は命令フェッチが発生されると論理「０」の値となるInMREQ

信号、プロセッサが停止すると論理「１」の値となるデバッグアクノーレッジ（DBGACK）信号、および単一ステップを行うかどうかを示すようセットされる、制御レジスタ２１０内のビット値であるCTL 信号を受信するようになっており、このCTL 信号は単一ステップ方法を実行すべき時は論理「１」の値を有する。論理回路７００はステートマシンのステートを示す２ビットの情報を受信するようにもなっており、これら２ビットは「StepStat e 」と称される。StepState ［１：０］の値はステートマシン３００の次のステートを表示するための論理回路７００によって出力される「NextState 」［１：０］信号に応じて決まり、論理回路７００への種々の入力信号と本発明の好ましい実施例により論理回路７００から出力されるNextState ［１：０］の値との間の関係は次の表に示される。

【００６８】

【表２】

StepState[1:0]	InMREQ	DBGACK	CTL	NextState[1:0]
xx	x	x	0	00
xx	x	1	x	00
00	0	0	1	01
01	0	0	1	10
10	0	0	1	10

【００６９】

上記表では、値００はステートマシン３００のアイドルステートに対応し、値「０１」はステートマシン３００のＦ１ステートに対応し、値「１０」はステートマシン３００のＦ２ステートに対応する。

データ処理回路に、まず電力が供給されると、パス７１５を通してＤタイプレジスタ７１０、更にＤタイプラッチ７２０にもリセット信号が送られる。このリセット信号はＤタイプレジスタ７１０がStepState としての値「００」を出力するようにさせ、Ｄタイプラッチ７２０が「StepBkpt」信号を「０」にリセットするようにさせる。

【００７０】

図５を参照すると、ＦＥ１クロックサイクルの間、InMREQ信号は低レベルになり、DBGACK信号は低レベルとなり、CTL 信号は高レベルとなり、単一ステップ動作を実行すべきと見なされる。ステップステートが「００」となると、上記表からNextState がＦ１ステートに対応する「０１」となることが示される。

【００７１】

Ｄタイプレジスタ７１０は、このNextState 値を受信し、パス７３０を通してクロック信号も受信する。ＦＥ１クロックサイクルの間、クロックの立ち上がりエッジでＤタイプレジスタ７１０はNextState 値、すなわちステップステートとしての「０１」を出力し、このステップステートは論理回路７００へ入力される。更にパス７６０を通してAND ゲート７８０にStepState のビット「１」が送られる。このAND ゲート７８０も（インバータ７７０を介し）反転されたDBGACK信号を受信するようになっている。ステップステート信号の第１ビットはこのケースが０であるので、AND ゲート７８０によって出力される信号は「０」となる。

【００７２】

ＤタイプラッチＤ７２０はAND ゲート７８０からの出力信号、更にクロック信号７３０を受信し、ＦＥ１クロックサイクルにおけるクロックの立下りエッジにて信号StepBkptとしてAND ゲート７８０から受信した信号を出力する。AND ゲート７８０からの信号はこのケー

スでは０であるので、StepBkpt信号も０となる。
【００７３】
次のクロックサイクルではInMREQおよびDBGACKの双方は低レベルであり、CTL は高レベルのままであり、StepState は「０１」である。上記表に示されているように、論理回路７００により出力される次のステートは「１０」となる。従って、この次のクロックサイクルにおけるクロックの立ち上がりエッジでは、ステート「１０」がＤタイプレジスタ７１０によってStepState として出力され、このステートはステートマシン３００のＦ２ステートに対応する。
DBGACK信号は「０」であるので、AND ゲート７８０はインバータ７７０からの論理「１」の値を受信する。更に、このステップステート値の第１ビットは「１」であるので、ANDゲートの第２入力端で論理「１」信号も受信し、よってＤタイプラッチ７２０に論理回路「１」信号を出力する。従って、Ｄタイプラッチ７２０によりクロックサイクルの立下りエッジでStepBkpt信号として論理「１」信号が出力される。図４に示されるように、このStepBkpt信号は、パス３３０を通してＯＲゲート３２０に出力される信号であり、この結果、プロセッサコアにブレークポイント信号が発生される。
【００７４】
第３クロックサイクルではInMREQおよびDBGACKは再び低レベルとなり、CTL は高レベルとなり、ステップステート信号は［１０］となる。表の最終行に示されるように、論理回路７００によって出力される次のステートは値「１０」のままであり、よってこのクロックサイクルの立下りエッジではStepBkpt信号は再び値「１」となる。
このようなステートの関係はプロセッサコアはプロセッサが停止したことを表示する高レベルのDBGACK信号をプロセッサコアが発生するときまで続く。上記表における第２の欄に示されているように、StepState 、InMREQ信号またはCTL 信号の値に係わらず、高レベルのDBGACK信号が存在することにより、論理回路７００によって出力される次のステートは「００」とされる。更に、反転したDBGACK信号はAND ゲート７８０にも加えられるので、これによりAND ゲートはStepState の値［１］に拘わらず、「０」の値を出力する。したがって、クロックサイクルの降下エッジではStepBkpt信号は「０」の値に戻る。
【００７５】
以上でステートマシン３００の動作について説明したので、次に図８～１０を参照して好ましい実施例に係わる埋め込み型ＩＣＥユニット１５０内に設けられたベクトルキャッチ論理回路３１０の動作について詳細に説明する。
図８は、例外、例えばデータアボート（data aborts ）が生じる時の例外ルーチンへのアクセスを識別するために、ベクトルキャッチ論理回路３１０を使用する際の好ましい実施例のデータ処理装置によって発生される種々の信号を示すタイミング図である。図８に示された種々の信号の相互作用については、図９に更に示されており、この図９はベクトルキャッチ論理回路３１０の主な要素を示すブロック図である。
【００７６】
種々のクロックサイクルの間の分離を示すために、図８内の垂直線８８０が設けられている。例外が生じた場合、プロセッサコア１００がこの例外を取り扱うために使用される例外ルーチンの第１命令を検索するよう、命令フェッチリクエストを発生する。従って、プロセッサコアはInMREQ信号を発生する他に、命令アドレスバスにアドレス８１０も発生する。このアドレスは、例外ルーチンにおける第１命令のアドレスに対応する。同時にプロセッサコア１００により、評価信号も出力される。この評価信号はInMREQ信号に加えて、制御バスに出力されることが好ましく、この信号は命令フェッチが例外ルーチンの命令に関連することを表示する。好ましい実施例では、この評価信号は「VectAdd 」信号と称され、図８ではライン８２０によって示されている。
【００７７】
図９に示されるように、ベクトルキャッチ論理回路３１０内のデコード論理回路９００に命令アドレス情報およびVectAdd 信号が入力される。好ましい実施例では、デコード論理回路９００には命令アドレスのうちのビット４～２しか入力されない。好ましい実施例で

は、どの命令ルーチンにアクセス中かを判断するのに、これらビットしか必要でないからである。しかしながら当業者であれば、必要に応じてデコード論理回路９００に命令アドレスの任意の数のビットを与えることができることが理解できよう。

【００７８】

デコード論理回路９００はベクトルキャッチレジスタ９３０にもアクセスし、これらレジスタは図４を参照してこれまで述べた埋め込み型ＩＣＥユニット１５０内の制御レジスタ２１０のエクストラフィールドとなっている。好ましい実施例では、８つの例外ルーチンが設けられており、よってベクトルキャッチレジスタ９３０には８つのフィールドが設けられている。更に各フィールドは、デバッガーが対応する例外ルーチンへのアクセスを識別したいかどうかを表示するよう設定可能な１ビットを含むことが好ましい。

【００７９】

例外ルーチンへのアクセスが行われていることを表示するよう、VectAdd 信号が高レベルであると仮定すると、デコード論理回路９００はアドレスのうちのビット４～２、およびどの例外ルーチンがアクセス中であるか、更にベクトルキャッチレジスタ９３０内の対応するビットがセットされているかどうかを判断するためのベクトルキャッチレジスタ９３０からの対応するビットを使用するようになっている。後に図１０を参照し、上記機能を実行するための、本発明の好ましい実施例におけるデコード論理回路９００内に含まれる実際の論理回路について詳細に説明する。

【００８０】

デバッガーが決定された例外ルーチンへのアクセスを識別したいことを、ベクトルキャッチレジスタ９３０内の対応するビットが表示する場合、デコード論理回路９００によりパス９４０を通してＤタイプラッチ９１０に「VDecode 」信号が出力される。このVDecode 信号は図８においてライン８３０によって示されており、VectAdd 信号と同様な形態であるが、デコード論理回路９００が必要な処理を実行するのに必要な時間により、時間がずれていることが判る。

【００８１】

Ｄタイプラッチ９１０はパス９７０を通してクロック信号を受信し、よってクロックの立下りエッジでパス９５０を通して「VTest 」信号としてVDecode 信号を出力するようになっている。このVTest 信号は、図８ではライン８４０によって示されており、フェッチサイクルの開始を示すクロックの立下りエッジの後で、このVTest 信号が出力されることが理解できよう。

【００８２】

VTest 信号は、パス９７０を通してクロック信号も受信するようになっている別のＤタイプラッチ９２０によって受信される。従って、フェッチサイクルの終了時におけるクロックの立下りエッジでＤタイプラッチ９２０はVBkpt 信号としてのVTest 信号をパス９６０を通して出力するようになっている。このVBkpt 信号は、図４を参照して先に示したように、パス３４０を通してＯＲゲート３２０へベクトルキャッチ論理回路３１０によって出力される信号に対応する。従って、図４に示された埋め込み型ＩＣＥユニット１５０内のORゲート３２０はベクトルキャッチ論理回路３１０からのVBkpt 信号を受信した際に、ブレークポイント信号を出力するようになっている。図８では、VBkpt 信号はライン８５０によって示され、ブレークポイント信号はライン８６０によって示されている。信号がベクトルキャッチ論理回路３１０からORゲート３２０へ通過し、ORゲート３２０によって処理されるのにかかる時間のためにブレークポイント信号８６０は一部がVBkpt 信号８５０の後で発生されることが、図８から理解できよう。

【００８３】

最後に、図８を参照すると、命令アドレスバス上に発生されるアドレス８１０に対応する命令データ８７０は、フェッチサイクルの終了に向かって命令バスを通してプロセッサコアに戻される。

図８および図９の上記説明から、例外ルーチン命令がフェッチされるフェッチサイクル直後のクロックサイクルにおけるブレークポイント信号のアサートが、上記方法によって保

証されることが理解できよう。従って、その命令が実行される前にプロセッサコアは強制的に処理を停止させられる。性質上、例外ルーチン命令は遅延を伴うことなく実行されるので、図８に示されるようにブレークポイント信号を１回アサートすれば充分である。

【００８４】

図１０は、図９に示されたデコード論理回路９００をより詳細に示す。図１０に示されるように、このデコード論理回路９００はパス９８０を通して命令アドレスのビット４～２を受信し、パス９９０を通してVectAdd 信号を受信し、パス９８５を通してベクトルキャッチレジスタからの８ビットを受信する。命令アドレスのうちのビット４～２はインバータ９９２によって反転され、反転されたアドレスＮＡ［２：０］を形成し、この信号は再び反転され、バッファ化されたアドレス指定されたビットＢＡ［２：０］を形成する。

【００８５】

好ましい実施例では命令アドレスのうちのビット４～２がすべて「０」であり、従ってＮＡ信号のうちのビット２～０がすべてが「１」であれば、第１例外ルーチンにアクセスされる。更にベクトルキャッチレジスタ９３０のビット０は、この例外ルーチンに対応し、よってベクトルキャッチレジスタのビット０はＮＡ信号のうちのビット２～０と共にANDゲート９９６へ送られる。更にAND ゲート９９６はパス９９０を通してVectAdd 信号を受信し、この信号は例外ルーチンへの命令フェッチが行われる時はいつも高レベルである。従って、図１０を参照するとＮＡ信号のビット２～０が高レベルである場合、ベクトルキャッチレジスタのビット０が高レベルであり（デバッガーが第１例外ルーチンへのアクセスの識別を望むことを表示する場合）、およびVectAdd 信号が高レベルであり、例外ルーチンへのアクセスが行われることを表示する場合、AND ゲート９９６はレベル「１」の信号を出力するだけであることが理解できよう。

【００８６】

ベクトルキャッチレジスタ９３０の他の７つのビットの各々に対し、同様なAND ゲート９９８、９９９が設けられており、これらAND ゲート９９８、９９９の特定の例外ルーチンに応じてＮＡ信号またはＢＡ信号のいずれかからの選択的ビットを受信するようになっている。したがって、一例として命令アドレスのうちのビット４および３が０であり、命令アドレスのうちのビット２が１である場合、第２例外ルーチンにアクセスされる。この場合、ＮＡ信号のうちのビット２および１は高レベルとなり、ＢＡ信号のうちのビット０は高レベルとなるので、これらビットはベクトルキャッチレジスタのうちのビット１の他にAND ゲート９９８へ送られる。したがって、第２例外ルーチンにアクセスされ、ベクトルキャッチレジスタのうちのビット１が高レベルにセットされ、デバッガーがその第２例外ルーチンへのアクセスを識別するのに関心のあることを表示する場合、AND ゲート９９８は、例外ルーチンへのアクセスが行われているのを表示するのにVectAdd 信号が高レベルである限り、ORゲート１０００に高レベル信号を発生する。

【００８７】

最終例として、命令アドレスのうちのビット４～２がすべて高レベルであり、よってＢＡ信号のうちのビット２～０が高レベルである場合、第８例外ルーチンにアクセスされる。従って、AND ゲート９９９はＢＡ信号のうちのビット２～０およびベクトルキャッチレジスタ９３０のうちのビット７を受信するようになっている。従って、ベクトルキャッチのうちのそのビットがセットされており、ＢＡ信号のうちの対応するビット２～０が高レベルであれば、AND ゲート９９９はORゲート１０００へ論理「１」信号を出力し、VectAdd 信号は高レベルとなり、例外ルーチンへのアクセスが行われていることを示す。

ORゲート１０００がAND ゲート９９６、９９８、９９９のうちのいずれかから論理「１」信号を受信する場合、ORゲートは論理「１」の値としてVDecode 信号を出力する。

【００８８】

本発明の好ましい実施例に係わるデータ処理装置のうちの上記記載から、単一ステッピング機能および例外ルーチンへのアクセスの識別をそれぞれ行う専用ステートマシン３００およびベクトルキャッチ論理回路３１０を設けたことにより、他の目的に自由に使用できる埋め込み型ＩＣＥユニット内に任意の一般的ハードウェアブレークポイントユニットを

残しながら、これら一般的なデバッギング条件を取り扱うための、特に効率的な技術が提供される。

【００８９】

以上で本明細書に特定の実施例を説明したが、本発明はこの実施例に限定されるものでなく、発明の範囲内で多くの変形および追加を行うことができることが理解できよう。例えば本発明の範囲から逸脱することなく、従属請求項の特徴事項と独立請求項の特徴事項との種々の組み合わせが可能であることが理解できよう。

【図面の簡単な説明】

【図１】データ処理装置がハーバードアーキテクチャを有する、本発明の第１実施例に係わるデータ処理装置を示す。

【図２】データ処理装置がフォンノイマンアーキテクチャを有する、本発明の第２実施例に係わるデータ処理装置を示す。

【図３】データ処理装置がキャッシュを使用せず、代わりにプロセッサコアがメモリに直接アクセスするようになっている、本発明の第３実施例に係わるデータ処理装置。

【図４】本発明の好ましい実施例に係わる埋め込み型ＩＣＥユニットの主要部品を示す。

【図５】図４のステートマシンに対応する種々の信号の間の関係を示すタイミング図である。

【図６】図４に示されたステートマシンの種々のステートの間の関係を示すステート図である。

【図７】図４に示されたステートマシンの部品を示すブロック図である。

【図８】図４に示されたベクトルキャッチ論理回路に対応する種々の信号の間の関係を示すタイミング図である。

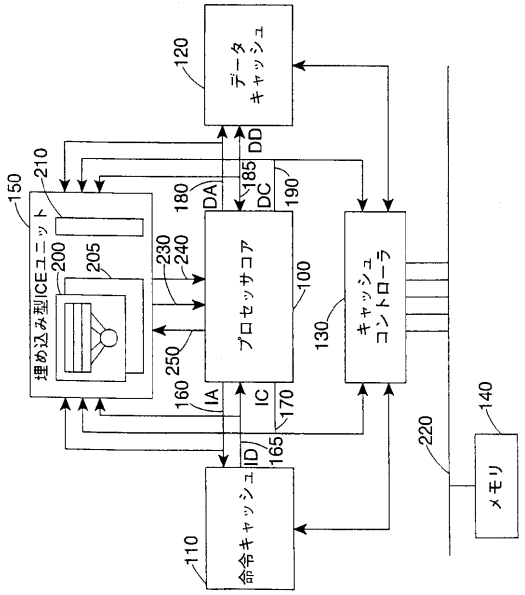
【図９】図４に示されたベクトルキャッチ論理回路の部品を示すブロック図である。

【図１０】図９のデコード論理回路内に含まれる論理回路を示す図である。

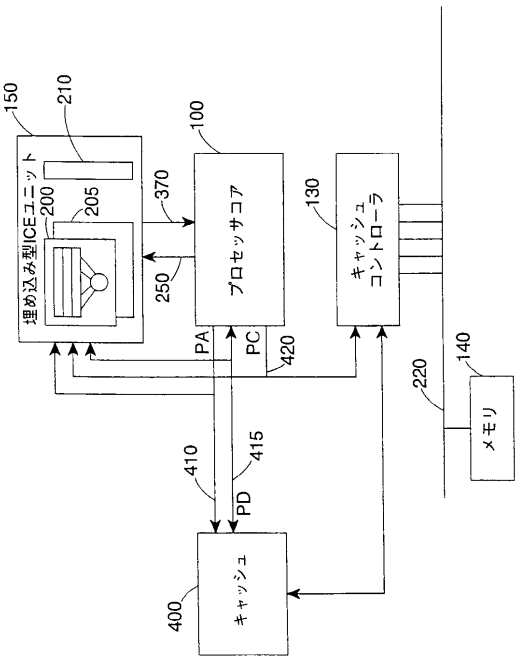
【符号の説明】

１００　プロセッサコア
１１０　命令キャッシュ
１２０　データキャッシュ
１４０　メモリ
１５０　埋め込み型ＩＣＥユニット

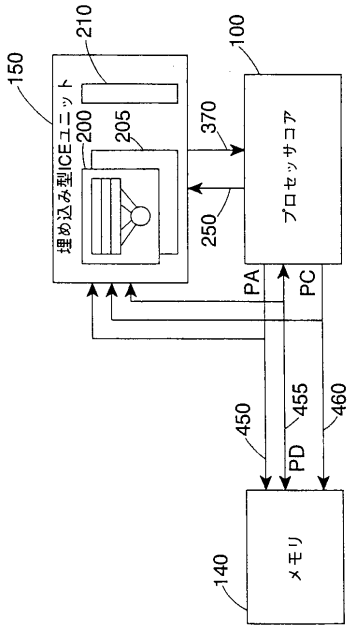
【図１】


データキャッシュ
120
150
210
埋め込み型ICEユニット
200
205
180
DA
185
DD
190
DC
230
240
プロセッサコア
100
250
130
キャッシュコントローラ
160
IA
IC
170
ID
165
命令キャッシュ
110
220
140
メモリ

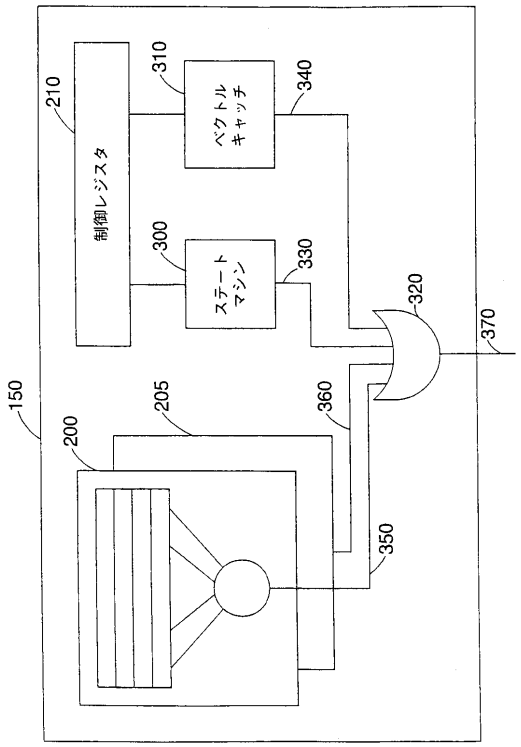
【図２】


150
210
埋め込み型ICEユニット
200
205
100
370
プロセッサコア
250
130
キャッシュコントローラ
PA
PC
420
220
140
メモリ
410
415
PD
400
キャッシュ

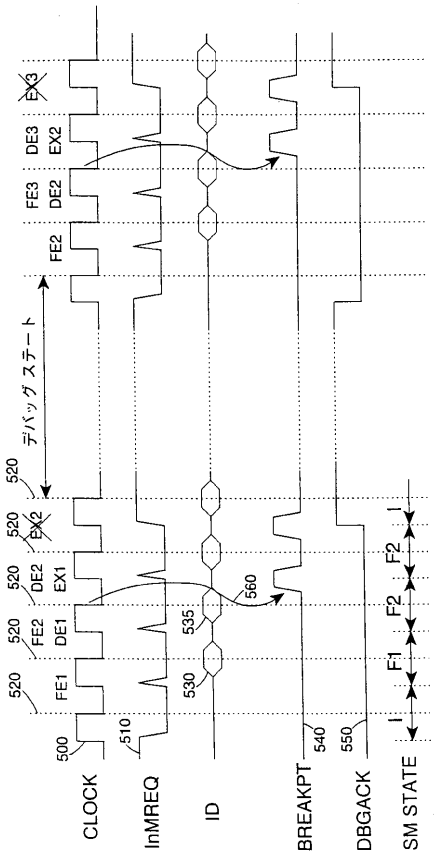
【図３】


150
210
埋め込み型ICEユニット
200
205
100
370
プロセッサコア
250
PA
PC
450
455
460
PD
140
メモリ

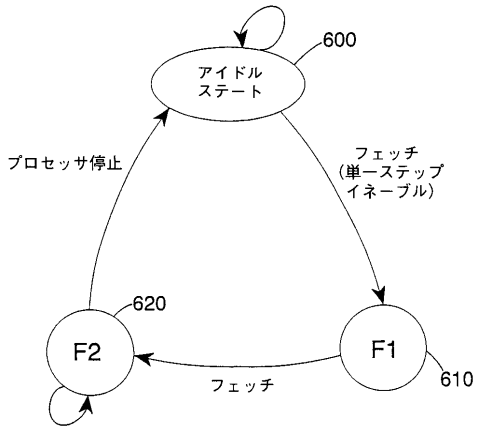
【図４】


210
制御レジスタ
310
ベクトルキャッチ
340
300
ステートマシン
330
320
370
150
205
360
200
350

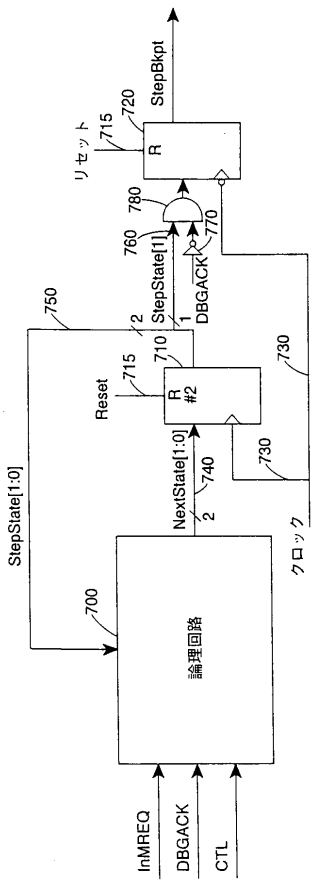
【図５】


CLOCK
InMREQ
ID
BREAKPT
DBGACK
SM STATE
500
510
520
530
535
540
550
560
FE1
DE1
FE2
EX1
DE2
EX2
DE3
EX3
デバッグ ステート
F1
F2
I

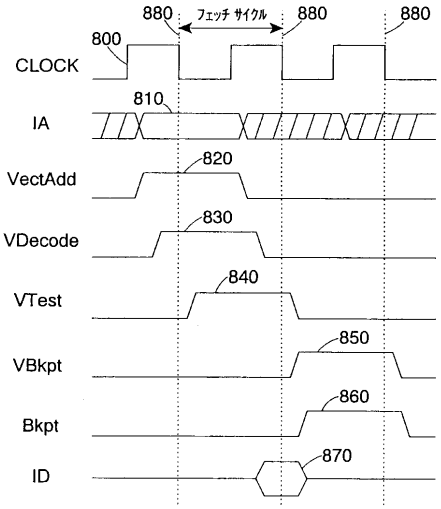
【図６】


600
アイドル
ステート
プロセッサ停止
フェッチ
(単一ステップ
イネーブル)
620
F2
F1
610
フェッチ

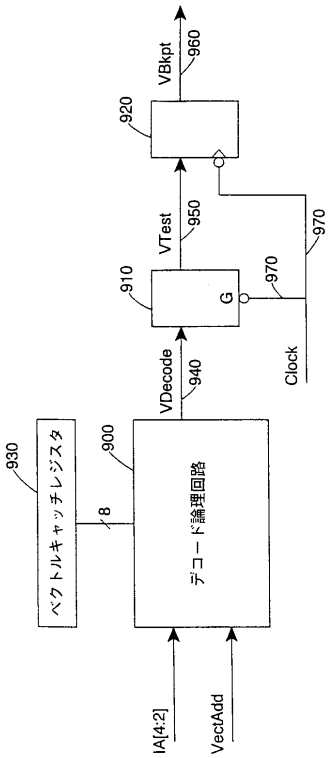
【図７】


StepBkpt
720
R
リセット
715
780
760
770
StepState[1]
DBGACK
750
2
1
710
730
Reset
R
#2
NextState[1:0]
740
StepState[1:0]
クロック
700
論理回路
InMREQ
DBGACK
CTL

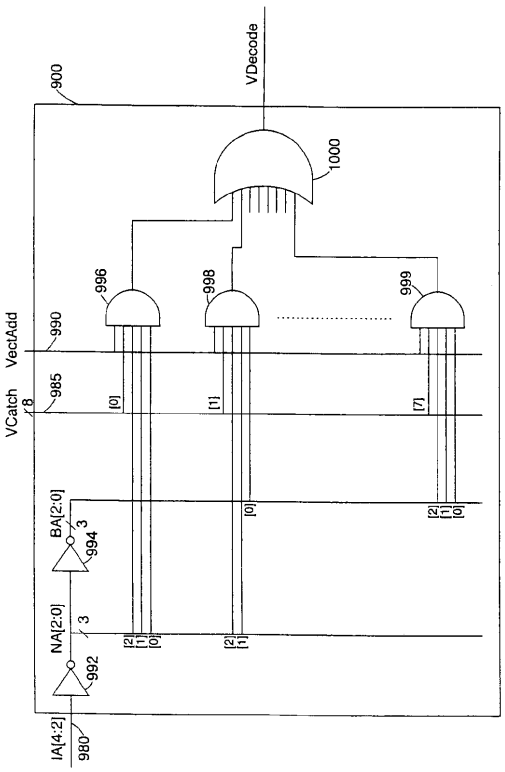
【図８】


880
フェッチ サイクル
880
880
800
CLOCK
810
IA
820
VectAdd
830
VDecode
840
VTest
850
VBkpt
860
Bkpt
870
ID

【図９】


VBkpt
960
920
VTest
950
910
G
970
970
Clock
VDecode
940
930
ベクトルキャッチレジスタ
8
900
デコード論理回路
IA[4:2]
VectAdd

【図１０】


900
VDecode
1000
996
998
999
VectAdd
990
VCatch
8
985
[0]
[1]
[7]
BA[2:0]
3
994
[0]
[2]
[1]
[0]
NA[2:0]
3
992
[2]
[1]
[0]
[2]
[1]
IA[4:2]
980

フロントページの続き

(72)発明者　サイモン　アンソニー　セガーズ
イギリス国　ケンブリッジシャー，ケンブリッジ，フルバーン，ペティッツ　クロース　７
(72)発明者　ピーター　ローガン　ハロッド
イギリス国　ケンブリッジシャー，ケンブリッジ，ルマー　ドライブ　１４
(72)発明者　アンドリュー　ジョン　マーリット
イギリス国　ケンブリッジシャー，ケンブリッジ，オーク　トリー　アベニュー　３３

審査官　多胡　滋

(56)参考文献　特開平０２－１５０９３３（ＪＰ，Ａ）
特開平０３－２２８１４５（ＪＰ，Ａ）
特開平０３－２０４０４３（ＪＰ，Ａ）
特開平０４－０５４６３６（ＪＰ，Ａ）
特開平０３－０７１３４５（ＪＰ，Ａ）
特開平０８－１８５３３６（ＪＰ，Ａ）
特開平０８－１６１１９０（ＪＰ，Ａ）
特開平０４－３５０７３５（ＪＰ，Ａ）
特開平０３－２８２６３５（ＪＰ，Ａ）
特開平０５－３２４３２２（ＪＰ，Ａ）
特開平０６－０６７９２９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 11/28