(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5419103号
(P5419103)
(45)発行日　平成26年2月19日(2014.2.19)　(24)登録日　平成25年11月29日(2013.11.29)

(51)Int.Cl.　　F I
G06F 11/28 (2006.01)　　G06F 11/28 320E
G06F 9/38 (2006.01)　　G06F 9/38 380C

請求項の数 5　(全 29 頁)

(21)出願番号　特願2010-526984 (P2010-526984)
(86)(22)出願日　平成20年8月4日(2008.8.4)
(65)公表番号　特表2010-541067 (P2010-541067A)
(43)公表日　平成22年12月24日(2010.12.24)
(86)国際出願番号　PCT/US2008/072109
(87)国際公開番号　WO2009/045628
(87)国際公開日　平成21年4月9日(2009.4.9)
審査請求日　平成23年8月4日(2011.8.4)
(31)優先権主張番号　11/864,292
(32)優先日　平成19年9月28日(2007.9.28)
(33)優先権主張国　米国(US)

(73)特許権者　504199127
フリースケール　セミコンダクター　インコーポレイテッド
アメリカ合衆国　テキサス州　７８７３５　オースティン　ウィリアム　キャノン　ドライブ　ウエスト　６５０１
(74)代理人　100142907
弁理士　本田　淳
(72)発明者　モイヤー、ウィリアム　シー.
アメリカ合衆国　７８６２０　テキサス州　ドリッピング　スプリングス　メドウリッジ　ドライブ　１１１１

審査官　大塚　俊範

最終頁に続く

(54)【発明の名称】デバッグイベントを監視するためのシステム及び方法

(57)【特許請求の範囲】
【請求項１】
複数の命令について、フェッチ、デコード、および実行と、各命令の実行に関連した結果の書込とを連続的に行うことによって、複数の命令を実行するためのパイプライン・プロセッサと、
前記パイプライン・プロセッサに結合されており、前記複数の命令の実行を監視してデバッグイベントが発生する時を判定し、デバッグ例外を発生させて命令処理フローを中断するためのデバッグ回路と、を含むシステムであって、前記デバッグ回路は、前記デバッグイベントの発生から該デバッグイベントの発生に応じたデバッグ例外の発生までに書き戻しによって完了する命令の数のカウントを行うための制御回路をさらに含む、システム。
【請求項２】
前記デバッグ回路の前記制御回路は、
前記デバッグイベントの発生から前記デバッグ例外の発生までに書き戻しによって完了する命令の数のカウントとしてカウント値を計数して与えるためのカウンタをさらに含む請求項１に記載のシステム。
【請求項３】
前記デバッグ回路は、前記デバッグイベントの発生から前記デバッグ例外の発生までに書き戻しによって完了する命令の数のカウントを保持するためのフィールドを有するデバッグ状態レジスタをさらに含む、請求項１に記載のシステム。

【請求項４】
前記デバッグ回路は、前記複数の命令のうちの１つのデコードを行うことと、前記パイプライン・プロセッサによって形成されたアドレスを１つ以上の所定のデバッグアドレスと比較してデータアドレス一致が発生したか否かを判定することと、前記パイプライン・プロセッサによってアクセスの行われたデータ値を１つ以上の所定のデータ値と比較してデータ値一致が発生したか否かを判定することとに応じて、デバッグイベントが発生したことを判定し、デバッグイベントは両比較演算について一致が発生することを必要とする、請求項１に記載のシステム。
【請求項５】
複数の命令について、フェッチ、デコード、および実行と、各命令の実行に関連した結果の書込とを連続的に行うことによって、複数の命令を実行するためのパイプライン処理回路と、
前記パイプライン処理回路に結合されており、１つ以上のデバッグアドレスを命令実行によって形成されるアドレスと選択的に比較することと、１つ以上のデバッグデータ値を命令実行によって形成されるデータと選択的に比較することとによって、前記複数の命令の実行を監視するデバッグ回路と、を含むシステムであって、前記デバッグ回路は、両方の比較が一致を生じるときを判定し、該判定に応答してデバッグイベントの発生を示すことと、前記デバッグ回路は、前記デバッグイベントの発生から該デバッグイベントの発生に応じたデータ値ブレークポイントの発生までに実行される命令の数のカウントを行う、システム。
【発明の詳細な説明】
【技術分野】
【０００１】
本開示は、一般的に、データ処理システムに関し、特に、デバッグイベントを監視するためのシステム及び方法に関する。
【背景技術】
【０００２】
データ値ブレークポイントには、ブレークポイント例外を発生させるか否かを判定するために、データアクセスアドレスと、そのデータアクセスアドレスに関係するデータとの両方の比較が含まれる。しかしながら、通常のパイプライン・プロセッサでは、データは、アクセスアドレスが利用可能になった後、何サイクルも経過するまで利用可能にならないことがある。従って、アクセスデータがまだ利用可能でないときにデータアクセスに続く１つ以上の命令が実行され得るので、命令ストリームのどこでブレークポイントが発生したか正確には不明である。現在利用可能なデータ処理システムには、アクセスデータがまだ利用可能でない間に命令が実行されないように、通常の実行に追加のストールを付加するものがある。しかしながら、このことは追加のオーバーヘッドを必要とし、また、通常の実行タイミングを乱すものであり、デバッグ時、望ましくないことがある。
【図面の簡単な説明】
【０００３】
【図１】本発明の一実施形態に基づくデータ処理システムのブロック図。
【図２】本発明の一実施形態に基づく、図１のデータ処理システムに関係するプロセッサのブロック図。
【図３】図１のデータ処理システムに関係する代表的なデバッグレジスタを示す図。
【図４】本発明の一実施形態に基づく、図３のデバッグレジスタに関係するデバッグ制御レジスタの図。
【図５】本発明の一実施形態に基づく、図４のデバッグ制御レジスタの一部の機能を表形式で示す図。
【図６】本発明の一実施形態に基づく、図４のデバッグ制御レジスタの一部の機能を表形式で示す図。
【図７】本発明の一実施形態に基づく、図３のデバッグレジスタに関係するデバッグ状態

レジスタの図。
【図８】本発明の一実施形態に基づく、図７のデバッグ状態レジスタの一部の機能を表形式で示す図。
【図９】図２のプロセッサの異なる動作例を示すタイミング図。
【図１０】図２のプロセッサの異なる動作例を示すタイミング図。
【図１１】図２のプロセッサの異なる動作例を示すタイミング図。
【図１２】様々な代表的なデータアドレス比較（ＤＡＣ）イベント及びそれらに対応する結果を表形式で示す図。
【図１３】様々な代表的なデータアドレス比較（ＤＡＣ）イベント及びそれらに対応する結果を表形式で示す図。
【図１４】様々な代表的なデータアドレス比較（ＤＡＣ）イベント及びそれらに対応する結果を表形式で示す図。
【図１５】様々な代表的なデータアドレス比較（ＤＡＣ）イベント及びそれらに対応する結果を表形式で示す図。
【図１６】様々な代表的なデータアドレス比較（ＤＡＣ）イベント及びそれらに対応する結果を表形式で示す図。
【図１７】本発明の一実施形態に基づく、図３のデバッグレジスタに関係するデバッグ状態レジスタの図。
【図１８】図２のプロセッサの動作例を示すタイミング図。
【発明を実施するための形態】
【０００４】
本発明は、一例として示すものであり、添付図によって限定されない。図では、同様な参照符号は、同様な要素を示す。図の要素は、簡単明瞭に示されており、必ずしも縮尺通りに描かれていない。
【０００５】
コードのデバッグ時、実際にどの命令がデバッグ例外発生を引き起こしたか知ることは、困難であることが多い。例えば、データ値ブレークポイントの場合、データアクセスアドレスの比較及びアクセスしたデータとのデータ値の比較の双方が実施されるので、比較結果によって通常の命令ストリームの実行を中断するデバッグ例外を発生させることがある。アクセスに対するデータが、そのアクセスの開始から何サイクルも後に返されることがあるが、これは、データアクセス要求及びアクセスアドレスを供給することによって起こる。要求したアクセスに基づくデータアドレス比較と受信データに基づくデータ値比較との間の遅れにより、命令ストリーム中のどの命令が実際に命令ストリームを中断したか判定することは、格段に困難なことがある。一実施形態では、データ値ブレークポイントを引き起こした命令と命令ストリーム中におけるブレークポイント例外が実際に発生するポイントとの間で幾つの命令が実行されたかを示すために、オフセット値が用いられる。このようにして、デバッグ機能の改善が達成される。
【０００６】
本明細書に用いる用語「バス」は、データ、アドレス、制御、又は状態等の、１つ又は複数の様々な種類の情報を伝達するのに用い得る複数の信号又はコンダクタ（導体）を参照するために用いる。本明細書で述べる導体は、単一の導体、複数の導体、単方向の導体、又は双方向の導体であることに関連して例示又は記述し得る。しかしながら、異なる実施形態では、導体の実装も異なり得る。例えば、双方向の導体よりもむしろ別個の単方向導体を用いたり、その逆の場合もあったりする。また、直列に又は時分割多重方式で複数の信号を伝達する単一の導体で複数の導体を置き換えることがある。同様に、複数の信号を搬送する単一の導体は、これらの信号の一部を搬送する様々な異なる導体に分離し得る。従って、信号を伝達するための多くのオプションが存在する。
【０００７】
用語「アサート」又は「設定」及び「ネゲート」（又は「ディアサート」又は「クリア」）は、信号、状態ビット、又は同様の装置をそれぞれその論理的に真又は論理的に偽の

状態にすることを参照する際に用いる。論理的に真の状態が論理レベル１である場合、論理的に偽の状態は、論理レベル０である。また、論理的に真の状態が論理レベル０である場合、論理的に偽の状態は、論理レベル１である。

【０００８】

本明細書に述べた各信号は、正又は負論理として構成し得る。この場合、負論理は、信号名称上のバー又は名称に続くアスタリスク記号（＊）によって示し得る。負論理信号の場合、信号は、アクティブ「ロー」であり、この場合、論理的に真の状態は、論理レベル０に対応する。正論理信号の場合、信号は、アクティブ「ハイ」であり、この場合、論理的に真の状態は、論理レベル１に対応する。本明細書で述べる信号は、いずれも負又は正のいずれかの論理信号として構成し得ることに留意されたい。従って、他の実施形態では、正論理信号として述べた信号は、負論理信号として実現してもよく、また、負論理信号として述べた信号は、正論理信号として実現してよい。

【０００９】

角括弧は、本明細書では、バスの導体又は値のビット位置を示すために用いる。例えば、「バス６０［７：０］」又は「バス６０の導体［７：０］」は、バス６０の８つの下位導体を示し、「アドレスビット［７：０］」又は「ＡＤＤＲＥＳＳ［７：０］」は、アドレス値の８つの下位ビットを示す。数に先行する記号「＄」は、その数が１６進数、即ち、基数１６の形式で表されることを示す。数に先行する記号「％」又は「０ｂ」は、その数が２進数、即ち、基数２の形式で表されることを示す。

【００１０】

図１は、本発明の実施形態と整合性のあるデータ処理システム１０を示す。データ処理システムは、システム・オン・チップであってよい。データ処理システム１０は、単一の集積回路又は複数の集積回路上に実装し得る。データ処理システム１０には、プロセッサ１２、外部デバッグ回路１４、Ｉ／Ｏモジュール１６、及びメモリ１８が含まれ、これらは、バス２０を介して結合し得る。他の実施形態では、メモリ１８は、任意の種類のメモリであってよく、また、プロセッサ１２と同じ集積回路上に配置してもよく、又はプロセッサ１２と異なる集積回路上に配置してもよい。メモリ１８は、任意の種類のメモリであってよく、例えば、読み出し専用メモリ（ＲＯＭ）、ランダムアクセスメモリ（ＲＡＭ）、不揮発性メモリ（例えば、フラッシュメモリ）等であってよい。更に、メモリ１８は、他の周辺装置もしくはスレーブ内又は異なる集積回路上に配置されたメモリ又は他のデータ記憶装置であってよい。

【００１１】

図２は、図１のデータ処理システム１０に関係するプロセッサ１２のブロック図である。プロセッサ１２には、命令パイプ２２、実行ユニット２４、命令フェッチユニット２６、制御回路２８、汎用レジスタ３０、ロード／記憶ユニット３２、バス・インターフェイス・ユニット（ＢＩＵ）３４、及び内部デバッグ回路４０を含み得る。プロセッサ１２は、ＢＩＵ（３４）に結合したバス２０を介して、データ処理システム１０の他の構成要素と通信を行い得る。内部デバッグ回路４０は、図２に示すデバッグポートを介して外部デバッグ処理ユニットに結合してよく、例えば、ＩＥＥＥ　ＩＳＴＯ－５００１準拠のＮｅｘｕｓ（商標）デバッグ処理ユニットに結合してよい。Ｎｅｘｕｓ（商標）は、テキサス州オースティン所在のフリースケール・セミコンダクタ（Ｆｒｅｅｓｃａｌｅ　Ｓｅｍｉｃｏｎｄｕｃｔｏｒ）社の商標である。デバッグポートは、ＪＴＡＧ等のシリアルインターフェイスであってよく、又はパラレルポート、シリアル及びパラレルポートの組合せ、又はイーサネット（登録商標）ポートとして実装してよい。内部デバッグ回路４０には、デバッグレジスタ４２及びデバッグ制御回路４４を含み得る。デバッグ制御回路４４には、１つ又は複数のオフセットカウンタ４１を含んでもよく、これらは、デバッグイベントを引き起こした命令と、そのデバッグイベントが処理される時の命令実行中の一つのポイントとの間で命令実行を完了させる命令（存在する場合）の数を求めるために用い得る。デバッグレジスタ４２には、命令ブレークポイント、データブレークポイント、ウォッチポイント、及びデバッグ処理に関連する他のメッセージ伝達を含む様々なデバッグ関連イ

ベントを制御するためのフィールドにグループ分けされたビットを含み得る。これらのデバッグ処理資源は、プロセッサ１２と外部デバッグ回路１４との間で共有し得る。更に、デバッグ制御回路４４は、導体３５を経由してＢＩＵ（３４）とアドレス及びデータを通信し得る。

【００１２】

次に、図３において、デバッグレジスタ４２内のレジスタは、更に、命令及び／又はデータアクセスブレークポイント及びウォッチポイントイベント、並びに他のデバッグ制御基準を実現するための１つ又は複数のアドレス比較値、アドレス範囲、及びデータマッチ値を記憶するために提供し得る。これらのアドレス及びデータ値は、様々な制御基準値と共に用いられ、プロセッサ１２が、ブレークポイント又はウォッチポイントイベントを発生させる目的で、１つ又は複数の所定の命令アドレス又はデータアドレスにいつアクセスするか判定される。これによって、内部デバッグモードが有効な場合、プロセッサ１２は、デバッグ例外のための例外処理を開始することができるようになる。あるいは、外部デバッグモードが有効な場合、プロセッサ１２は、デバッグ休止モードに入ることができるようになり、この場合、内部デバッグユニット４０のデバッグポートを通して外部デバッグ回路１４によって提供されたコマンドに応答する。一例として、デバッグレジスタ４２には、様々なデバッグ制御レジスタを含んでよく、例えば、デバッグ制御レジスタ５０（ＤＢＣＲ０）及び他のデバッグ制御レジスタ４３（ＤＢＣＲ１、ＤＢＣＲ２、ＤＢＣＲ３、及びＤＢＣＲ４）を含んでよい。デバッグレジスタ４２には、更に、命令アドレス比較レジスタ４５（ＩＡＣ１及びＩＡＣ２）を含み得る。命令アドレス比較レジスタ４５は、アドレス比較の目的のために命令アドレスを記憶し得る。デバッグレジスタ４２には、更に、データアドレス比較レジスタ４７（ＤＡＣ１及びＤＡＣ２）を含み得る。データアドレス比較レジスタ４７は、アドレス比較の目的のためにデータアクセスアドレスを記憶し得る。デバッグレジスタ４２には、更に、デバッグ状態レジスタ４９、デバッグカウンタ５１（ＤＢＣＮＴ１及びＤＢＣＮＴ２）、及びデータ値比較レジスタ５３（ＤＶＣ１及びＤＶＣ２）を含み得る。デバッグレジスタ４２は、ユーザのソフトウェア・プログラミング・モデルの一部であってよい。デバッグカウンタ５１は、１つ又は複数の計数割込可能なイベントが発生すると、カウントダウンするように構成し得る。カウント値が０に達すると、デバッグ計測イベントが通知され、割込可能ならば、デバッグ割込を生成し得る。データ値比較レジスタ５３は、データ比較の目的のためにデータ値を記憶し得る。

【００１３】

内部デバッグモードにおいて、これらのレジスタ資源は、ソフトウェアによって管理され、外部デバッグ回路は、用いる必要が無い。ソフトウェアは、専用レジスタ命令への及び専用レジスタ命令からのムーブ（ｍｏｖｅ）を用いたデータ移動によりレジスタを構成することができるが、これらの専用レジスタ命令は、ソフトウェアベースのデバッグ処理アクティビティを実施するために個々のデバッグレジスタを初期化するためのプログラマ・モデルのソフトウェア命令であり、この場合、割込可能なデバッグイベントによりソフトウェアデバッグ割込が発生する。すると、ソフトウェア割込ハンドラは、データ処理システム１０のソフトウェア・プログラマによって判定される様々な所望のアクティビティを実施し得る。外部デバッグモードでは、外部デバッグ回路１４は、これらの共有デバッグイベント設定レジスタの所有権を割り当てられ、また、設定されたデバッグイベントが発生すると、プロセッサ１２は、休止状態に入り、コマンドが外部デバッグ回路１４によって提供されるのを待つことができる。ソフトウェアは、外部デバッグモードが割込可能である場合、もはや共有デバッグ資源を制御しない。外部デバッグ回路１４は、デバッグレジスタ４２を含む共有デバッグ資源にデバッグポート（図２に示す）を介して直接アクセスすることができるが、デバッグポートは、例えば、ＪＴＡＧ　ＴＡＰポートとして実現し得る。一実施形態では、デバッグレジスタ４２は、様々なＪＴＡＧ命令用の１つ又は複数のフィールド内にレジスタ選択符号が含まれるＪＴＡＧデータレジスタとしてマッピングし得るが、これらＪＴＡＧ命令は、ＪＴＡＧ　ＩＲ及びＤＲ動作を介したデバッガによるレジスタの読込及び書出アクセスを規定する。

【００１４】
一組のレジスタを共有すると、実装すべきプロセッサ１２の資源が低減され、これにより、データ処理システム１０のユーザ向けのプログラミングモデルが簡素化される。内部デバッグユニット４０は、プロセッサ１２内のアクティビティを監視し、また、記憶したデバッグ設定情報に基づく１つ又は複数の所定条件の検出に応じて、１つ又は複数のデータブレークポイントイベント、命令ブレークポイントイベントを発生させることができ、更に、分岐又はトラップ発生イベント、命令完了イベント等の命令実行イベントを発生させることができる。このような動作では、プロセッサ１２は、当業者が認識できるように機能する。
【００１５】
図４は、図１のデータ処理システムに関係するデバッグ制御レジスタ５０の図である。デバッグ制御レジスタ５０は、デバッグレジスタ４２の一部として含むことができ、デバッグレジスタ４２は、更に、内部デバッグユニット４０の一部として含み得る。デバッグ制御レジスタ５０は、デバッグ設定情報を記憶するために用い得る。図４は、特定のビットフィールドを用いる本発明の特定の実施形態を示すが、本発明の他の実施形態では、各フィールドに様々な数のビットを有する異なるビットフィールドを用い得る。図４に示す特定のビットフィールドは、例示のみの目的で示す。一例として、デバッグ制御レジスタ５０には、３２ビットを含み得る。デバッグ制御レジスタ５０には、ＥＤＭ（５２）、ＩＤＭ（５４）、ＩＣＭＰ（５８）、ＢＲＴ（６０）、ＩＡＣ１（６２）、ＩＡＣ２（６４）、ＤＡＣ１（６６）、ＤＡＣ２（６８）、ＤＶＣ１（７０）、及びＤＶＣ２（７２）と表記したビットフィールドを含み得る。これらのビットフィールドは、単なる例であり、デバッグ制御レジスタ５０には、これより少ない又は追加のビットフィールドを含み得る。更に、これらのビットフィールドは、様々に構成し得る。デバッグ制御レジスタ５０には、更に、将来用い得る予約ビットフィールド５６、６１、及び７４を含み得る。様々なビットフィールドの機能は、図５及び図６で後述する。一例として、デバッグ制御レジスタ５０は、書込可能レジスタであってよく、更に読込可能であってもよいが、これは、ユーザのソフトウェア・プログラミング・モデルの一部であってよい。本発明の他の実施形態では、デバッグ制御レジスタ５０は、ユーザのソフトウェア・プログラミング・モデル内の制御レジスタでなくてよいが、その代わり、ユーザのソフトウェア・プログラミング・モデル外に実装してよい。任意の種類の記憶回路をデバッグ制御レジスタ５０の実現のために用い得る。
【００１６】
図５は、図４のデバッグ制御レジスタ５０の一部の機能を表形式で示す。ＥＤＭビット５２は、外部デバッグモードが有効か無効かを示し得る。例えば、ＥＤＭビット５２を１に設定すると、デバッグ制御レジスタ５０等の制御レジスタは、外部デバッグ回路１４の排他的制御下に置かれ、データ処理システム１０のソフトウェアは、これらの制御レジスタに情報を書き込めない。他の選択肢として、ＥＤＭビット５２を１に設定すると、ソフトウェアは、デバッグ制御レジスタの特定の部分に書き込めない。従って、ＥＤＭビット５２は、デバッグ制御及びセットアップ情報を含み得るデバッグ制御レジスタ５０及び他のデバッグ資源に記憶された情報が何らかのリセットイベントにより消去されるのを選択的に阻止するために用いられる。ＩＤＭビット５４は、内部デバッグモードが有効か無効かを示し、従って、デバッグ例外が有効か無効かを示し得る。ＩＣＭＰビット５８は、命令完了デバッグイベントが有効か無効かを示すために用い得る。ＢＲＴビット６０は、分岐発生デバッグイベントが有効か無効かを示すために用い得る。ビット６：７（６１）は、将来の用途のために予約し得る。次に、図６を参照すると、図６は、図４のデバッグ制御レジスタ５０の一部の機能を表形式で示す。ＩＡＣ１ビット６２は、命令アドレス比較１デバッグイベントが有効か無効かを示すために用い得る。ＩＡＣ２ビット６２は、命令アドレス比較２デバッグイベントが有効か無効かを示すために用い得る。ＤＡＣ１ビット６６は、データアドレス比較１デバッグイベントが有効か無効かを示すために用い得る。有効な場合、ＤＡＣ１ビット６６は、更に、どの種類の記憶アクセスに対してデータアド

レス比較１デバッグイベントが有効かを示す（例えば、記憶（ストア）型のデータ記憶アクセス、ロード型のデータ記憶アクセス、又はロード型もしくは記憶型両方のデータ記憶アクセス）。ＤＡＣ２ビット６８は、データアドレス比較２デバッグイベントが有効か無効かを示すために用い得る。有効な場合、ＤＡＣ２ビット６８は、更に、どの種類の記憶アクセスに対してデータアドレス比較１デバッグイベントが有効かを示す（例えば、記憶型のデータ記憶アクセス、ロード型のデータ記憶アクセス、又はロード型もしくは記憶型両方のデータ記憶アクセス）。ＤＶＣ１ビット７０は、データ値比較１修飾子が有効か否かを示すために用い得る。ＤＶＣ２ビット７２は、データ値比較２修飾子が有効か否かを示すために用い得る。ビット１６：３１は、将来の用途のために予約し得る。図５及び６は、デバッグイベントに関係する異なる設定情報を提供するための特定の数のビットフィールドについて述べるが、これらの図に示したものと異なる数のビットフィールドも用いてよい。

【００１７】

図７は、図１のデータ処理システムに関係するデバッグ状態レジスタ４９の図である。デバッグ状態レジスタ４９は、デバッグレジスタ４２の一部として含むことができ、デバッグレジスタ４２は、更に、内部デバッグユニット４０の一部として含み得る。デバッグ状態レジスタ４９は、デバッグイベントに関する状態情報を記憶するために用い得る。図７は、特定のビットフィールドを用いる本発明の特定の実施形態を示すが、本発明の他の実施形態は、各フィールドに様々な数のビットを有する異なるビットフィールドを用い得る。図７に示す特定のビットフィールドは、例示だけの目的で示す。一例として、デバッグ状態レジスタ４９には、３２ビットを含み得る。デバッグ状態レジスタ４９には、ＩＤＥ（７６）、ＩＣＭＰ（７８）、ＢＲＴ（８０）、ＩＡＣ１（８２）、ＩＡＣ２（８４）、ＩＡＣ３（８６）、ＩＡＣ４（８８）、ＤＡＣ１Ｒ（９０）、ＤＡＣ１Ｗ（９２）、ＤＡＣ２Ｒ（９４）、ＤＡＣ２Ｗ（９６）、及びＤＡＣ＿ＯＦＳＴ（９８）と表記したビットフィールドを含み得る。これらのビットフィールドは、単なる例であり、デバッグ状態レジスタ４９には、これより少ない又は追加のビットフィールドを含み得る。更に、これらのビットフィールドは、様々に構成し得る。デバッグ状態レジスタ４９には、更に、将来用い得る予約ビットフィールド１００を含み得る。様々なビットフィールドの機能は、図８で以下に述べる。更に、デバッグ状態レジスタ４９において、ビットの設定は、論理レベル１の書込を意味し、ビットのクリアは、論理レベル０の書込を意味する。一例として、デバッグ状態レジスタ４９は、そのビットがハードウェアを介して設定され、ソフトウェアを介して読込及びクリアされるレジスタであってよく、また、ユーザのソフトウェア・プログラミング・モデルの一部であってよい。本発明の他の実施形態では、デバッグ状態レジスタ４９は、ユーザのソフトウェア・プログラミング・モデル内になくてもよく、その代わり、ユーザのソフトウェア・プログラミング・モデル外に実装してもよい。一実施形態では、デバッグ状態レジスタ４９のデバッグ状態ビットは、内部デバッグモードが有効か又は外部デバッグモードが有効の間だけデバッグイベントによって設定される。また、一実施形態では、デバッグ割込が有効な場合、デバッグ状態レジスタ４９の設定済みビットによって、デバッグ割込を生成し得るが、この場合、デバッグ割込ハンドラが、通常実行に戻る前に、デバッグ状態レジスタ４９のビットをクリアする役割を担っている。更に、任意の種類の記憶回路を用いて、デバッグ状態レジスタ４９を実現し得る。

【００１８】

図８は、図７のデバッグ状態レジスタ４９の機能を表形式で示す。ＩＤＥビット７６は、曖昧なデバッグイベントの発生を示すために用いられ、従って、デバッグ例外が無効であり、且つ、デバッグイベントによってそのそれぞれのデバッグ状態レジスタビットが１に設定される場合、１に設定されることがある。即ち、デバッグイベントは、起こり得るが、デバッグ例外は、パイプラインの現在の状態により割込がまだ起こり得ないことから、無効のままであり得る。ＩＣＭＰビット７８は、命令完了デバッグイベントが発生した場合、１に設定し得る。ＢＲＴビット８０は、分岐発生デバッグイベントが発生した場合、１に設定し得る。ＩＡＣ１ビット８２は、ＩＡＣ１デバッグイベントが発生した場合、

１に設定し得る。ＩＡＣ２ビット８４は、ＩＡＣ２デバッグイベントが発生した場合、１に設定し得る。ＩＡＣ３ビット８６は、ＩＡＣ３デバッグイベントが発生した場合、１に設定し得る。ＩＡＣ４ビット８８は、ＩＡＣ４デバッグイベントが発生した場合、１に設定し得る。ＤＡＣ１Ｒビット９０は、ＤＡＣ１ビット６６が％１０又は％１１に等しい（図６に示すように、ＤＡＣ１デバッグイベントがロード型のデータ記憶アクセスに対して有効であることを示す）間に、読込型のＤＡＣ１デバッグイベントが発生した場合、１に設定し得る。ＤＡＣ１Ｗビット９２は、ＤＡＣ１ビット６６が％０１又は％１１に等しい（図６に示すように、ＤＡＣ１デバッグイベントが記憶型のデータ記憶アクセスに対して有効であることを示す）間に、書込型のＤＡＣ１デバッグイベントが発生した場合、１に設定し得る。ＤＡＣ２Ｒビット９４は、ＤＡＣ２ビット６８が％１０又は％１１に等しい（図６に示すように、ＤＡＣ２デバッグイベントがロード型のデータ記憶アクセスに対して有効であることを示す）間に、読込型のＤＡＣ２デバッグイベントが発生した場合、１に設定し得る。ＤＡＣ２Ｗビット９６は、ＤＡＣ２ビット６８が％０１又は％１１に等しい（図６に示すように、ＤＡＣ２デバッグイベントが記憶型のデータ記憶アクセスに対して有効であることを示す）間に、書込型のＤＡＣ２デバッグイベントが発生した場合、１に設定し得る。ＤＡＣ＿ＯＦＳＴビット９８は、データアドレス比較オフセットを示すために用い得る。一実施形態では、ビット１３乃至３１は、考えられる将来の用途のために予約される。

【００１９】

ＤＡＣ１又はＤＡＣ２用にデータ値比較修飾子が、デバッグ制御レジスタ５０に示されている場合、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベントを示すために、データアクセスアドレスの一致並びにデータ値の一致（そのデータアクセスアドレスに関係するデータ値に対する）が起こらなければならない。即ち、それぞれＤＡＣ１Ｒビット９０、ＤＡＣ１Ｗビット９２、ＤＡＣ２Ｒビット９４、及びＤＡＣ２Ｗビット９６は、データ値も一致するまで、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベントを示すようには設定されない。尚、一致すべき値は、データアドレス比較レジスタ４７及びデータ値比較レジスタ５３等のデバッグレジスタ４２に記憶し得る。例えば、デバッグ制御レジスタ５０のＤＶＣ１ビット７０を設定する場合、ＤＡＣ１デバッグイベントは、データ値比較によって限定される。これは、ＤＶＣ１　ＤＡＣ１デバッグイベントを示すには、アドレスがＤＡＣ１データアドレス比較レジスタに一致しなければならず、また、そのアドレスに対する関連データ値が、ＤＶＣ１データ値比較レジスタに一致しなければならないことを意味する。しかしながら、プロセッサ１２のパイプラインの性質により、データは、アクセスアドレスが利用可能になった後何サイクルも比較に利用できないことがある。更に、データが利用可能になりＤＶＣ　ＤＡＣが示されても、その瞬間に命令ストリームが割込可能でないことがある（即ち、デバッグ例外が無効である可能性がある）。これは、実際のデータ値ブレークポイントが、ＤＶＣ　ＤＡＣが示される瞬間に起こり得ないことを意味する。この場合、ＩＤＥビット７６は、デバッグイベントが示された時、デバッグ例外が起こり得ないという曖昧なデバッグイベントを示ために１に設定し得る。即ち、実際のデータ値ブレークポイントは、デバッグ例外が有効になるまで起こり得ず、これは、図９乃至１１を参照して説明するように、後続の命令ストリームにおける様々な（及び予測不可能な）ポイントで起こり得る。デバッグ例外が起こると、割込処理が始まり、デバッグイベントが処理される。

【００２０】

従って、ＤＡＣ＿ＯＦＳＴビット９８を用いると、ＤＶＣ　ＤＡＣを示させた命令と、データ値ブレークポイントが起こり且つデバッグ例外が発生するポイントと、の間に実行される命令の数を示し得る。ＤＡＣ＿ＯＦＳＴビット９８は、データアドレス比較デバッグ例外を起こしたロード又は記憶（ストア）命令のアドレスからのセーブしたＤＳＲＲ０値の「オフセット－１」を記憶するために用い得る。尚、ＤＳＲＲ０は、デバッグセーブ復旧レジスタ０に対応し、セーブしたＤＳＲＲ０値は、ブレークポイント割込用の復帰ポインタに対応する。従って、データ値ブレークポイントが発生した場合、セーブしたＤＳ

ＲＲ０アドレス値は、割込処理前に実行された最後の命令に続く命令のアドレス値に対応する。このようにして、ＤＶＣ１　ＤＡＣ１を引き起こしたアドレスと、命令ストリームが中断されデータ値ブレークポイントが生じるポイントとの間で幾つの命令が実行されたかを求めることができる。これによって、ユーザは、どの命令が実際にデータ値ブレークポイントを引き起こしたか判定し得る。一実施形態では、ＤＡＣ＿ＯＦＳＴビット９８は、通常％００に設定され、ＤＶＣ　ＤＡＣは、このフィールドを、「オフセット－１」を表す％０１、％１０、又は％１１に設定する。図９乃至１１に関して、後で例を示す。しかしながら、アクセス許可エラー又は他のアクセス関連エラーにより、同時変換索引バッファミス（ＤＴＬＢエラーとも称する）又はデータ記憶割込（例えば、ＤＳＩ）エラーが起こった場合、ＤＡＣ＿ＯＦＳＴビット９８は、％００に設定され、ＩＤＥビット７６は、１に設定し得ることに留意されたい。これらの場合、これらのエラーの内の１つが起こることにより、そのアドレスに関係するデータは、決して利用可能になり得ないことに留意されたい。

【００２１】

様々な状況及び条件が、データアドレス比較を実施する時とデータ値ブレークポイントが実際に発生する時との間に実行される命令の数に影響を及ぼし得る。図９乃至１１が示すタイミング図は、どのように一連の４つのロード命令（Ｉ０、Ｉ１、Ｉ２、及びＩ３、によって表され、この場合、Ｉ０がＤＶＣ　ＤＡＣデバッグイベントを発生させる）が異なるＤＡＣオフセット値をもたらすかを表す。これらのＤＡＣオフセット値は、ＤＡＣオフセットカウンタ４１によって追跡し得るが、Ｉ０と、Ｉ０のＤＶＣ　ＤＡＣが示された後、データ値ブレークポイントのデバッグ例外が起こる時点と、の間で実行される命令の数を表す又は符号化する。図９乃至１１の場合、Ｉ０が、ＤＶＣ１　ＤＡＣ１デバッグイベントを発生させると仮定する。従って、ＤＡＣ１ビット６６は、％１０又は％１１であり、ＤＡＣ１デバッグイベントが読込型のデータ記憶アクセスに対して有効であり、ＤＶＣ１ビット７０は、ＤＡＣ１デバッグイベントがデータ値比較によって限定されることを示すように設定される。Ｉ０のＤＶＣ１　ＤＡＣ１デバッグイベントに対応する状態ビットは、ＤＡＣ１Ｒビット９０であり、これは、同じアクセスに関するアドレス値比較及びデータ値比較の各々が一致するまで設定されない。尚、本例では、データアドレス比較の場合、データアドレス比較レジスタ４７のＤＡＣ１値は、Ｉ０によって演算されたデータアドレスとの比較に用い得る。また、データ値比較の場合、データ値比較レジスタ５３のＤＶＣ１値は、Ｉ０に関係する読込結果のデータとの比較に用い得ることに留意されたい。他の実施形態では、Ｉ０は、ＤＡＣ２ビット６８、ＤＶＣ２ビット７２、ＤＡＣ２Ｒビット９０等を用いて、ＤＡＣ２　ＤＶＣ２に対応してもよい。

【００２２】

図９乃至１１の各々は、六段パイプラインについて例示しており、これらの段には、フェッチ、デコード、有効アドレス（ＥＡ）／Ｅ０、メモリ１（ｍｅｍ１）／Ｅ１、メモリ２（ｍｅｍ２）／Ｅ２、及び書き戻しを含むことに留意されたい。これは、プロセッサ１２の命令パイプ２２のパイプライン例として提供されるが、他の実施形態には、例えば、異なる数の段を有する異なるパイプラインを含んでもよいことに留意されたい。図９乃至１１で参照される六段パイプライン等の命令パイプラインの動作は、当分野で公知であり、従って、本明細書では詳述しない。更に、図９乃至１１の各々には、プロセッサ１２のプロセッサ・クロックに対応し得るクロック信号が含まれる。プロセッサ・クロックは、システムクロック又はプロセッサ１２の一部だけに供給されるクロックであってよく、当分野で公知なように生成し得る。

【００２３】

図９において、Ｉ０は、クロックサイクル１においてフェッチ段に入り、クロックサイクル２においてデコード段に入り、クロックサイクル３においてＥＡ／Ｅ０段に入り、ここでＩ０の有効アドレスが計算され、そして、クロックサイクル４においてｍｅｍ１／Ｅ１段に入る。従って、有効アドレスは、クロックサイクル３の終わりに準備が整っており、導体３５のアドレス部でデバッグ制御回路４４に提供され、Ｉ０の有効アドレスとデー

タアドレス比較レジスタ４７に記憶されたＤＡＣ１値との間でアドレス比較を実施できるようにする（尚、この時点で、ＢＩＵ（３４）によってバス２０上にもアドレスを配置し得る）。このＤＡＣ１アドレス比較は、ＤＡＣ１アドレス比較信号の正のパルスによって示されるように、クロックサイクル４において実施するが、ＤＡＣ１アドレス比較信号は、命令Ｉ０に対してＤＡＣ１アドレス比較の一致が起こっていることを示すためにデバッグ制御回路４４内でアサートされる制御信号であってよい。しかしながら、ＤＡＣ１は、ＤＶＣ１修飾子を必要とすることから（ＤＶＣ１ビット７０の設定により）、ＤＡＣ１Ｒデバッグイベントは、アドレス比較が一致してもまだ示されない。命令処理は、Ｉ０に関係する（Ｉ０ロード命令に応じて検索される）データがまだデータ値比較のために利用可能でないため、継続する。次に、Ｉ０は、ｍｅｍ２／Ｅ２段に進み、この間、データがメモリ（例えば、図１のメモリ１８等）から読み込まれる。この読み込んだデータは、この段の終わりで利用可能となり得る（また、例えば、バス２０を介してＢＩＵ（３４）によって受信される）。そして、導体３５のデータ部でデバッグ制御回路４４に提供され、Ｉ０に関連する読み込んだデータ値とデータ値比較レジスタ５３に記憶されたＤＶＣ１値との間でデータ値比較を実施できるようにする。従って、データ値比較は、ＤＶＣ１データ比較信号の正のパルスによって示されるように、次のクロックサイクルであるクロックサイクル６において実施するが、ＤＶＣ１データ比較信号は、命令Ｉ０に関してＤＶＣ１データ比較の一致が起こっていることを示すためにデバッグ制御回路４４内でアサートされる制御信号であってよい。データ値比較レジスタ５３に記憶したＤＶＣ１値に一致すると、ＤＶＣ１　ＤＡＣ１デバッグイベントが示され、従って、ハードウェアが、ＤＡＣ１Ｒビット９０を設定する。

【００２４】

サイクル６内においてＩ０上でＤＶＣ１　ＤＡＣ１を検出すると、ＤＡＣオフセットカウンタが０にクリアされ、また、後続の各完了した命令の計数が開始できるようになることに留意されたい。しかしながら、パイプラインは、Ｉ１及びＩ２がメモリ段（ｍｅｍ１及びｍｅｍ２）にあるため、クロックサイクル６ではまだ中断できない。従って、Ｉ１及びＩ２は、メモリアクセスが開始されると、中断できず、完了しなければならないことから、待ち状態のデバッグ例外を更に処理する前に完了しなければならない。後続のメモリアクセスは、しかしながら、ＤＶＣ１　ＤＡＣ１例外が現在待ち状態であることから、起動されない。Ｉ１及びＩ２の各メモリアクセスは、書き戻し段に入った後、完了と見なされる。Ｉ２（命令Ｉ１及びＩ２の後側）は、クロックサイクル８の書き戻し段にあり、その後、待ち状態のデバッグ例外を発生させることができ、割込処理を開始できることに留意されたい。即ち、ＤＶＣ１　ＤＡＣ１デバッグイベントの割込処理は、クロックサイクル８の後（例えば、クロックサイクル９）まで開始されない。ＤＶＣ１　ＤＡＣ１を検出する時点で、Ｉ３は、まだパイプラインのＥＡ／Ｅ０段にあり、従って、キルし得ることに留意されたい。即ち、命令ストリームは、Ｉ３において中断される（即ち、デバッグ例外は、Ｉ２の実行後、Ｉ３の実行前に起こる）。通常の実行は、従って、割込処理の完了後に、Ｉ３で再開し得る。従って、ＤＳＲＲ０レジスタは、Ｉ３のアドレスを記憶し得るが、その理由は、これが、データブレークポイントデバッグ割込の復帰ポインタに対応するためである。デバッグ割込ハンドラのソフトウェア・ルーチンの完了に続いて、通常の実行は、割込命令からの復帰を実行することによって再開されるが、通常の（非割込）命令実行を継続するために、復帰すべき命令（この場合、Ｉ３）へのポインタとしてＤＳＲＲ０にセーブした値が用いられる。

【００２５】

ＤＶＣ１　ＤＡＣ１デバッグイベントを検出してＤＡＣオフセットカウンタをクリアした後、後続の各命令が完了した状態で、ＤＡＣオフセットカウンタは、デバッグ例外が発生する（この時点で割込処理が起こる）まで、１ずつ増加することに留意されたい。従って、ＤＡＣオフセットカウンタは、後続の命令Ｉ１及びＩ２が、それぞれクロックサイクル７及び８で完了することから、クロックサイクル８において２まで増加し、従って、デバッグ状態レジスタ４９のＤＡＣ＿ＯＦＳＴビット９８は、％１０に設定される。即ち、

ＤＶＣ１　ＤＡＣ１を引き起こした命令（本例では、Ｉ０）の後で且つデバッグイベント割込処理に先立って実行される命令の数は、２であり、これは、ＤＡＣ＿ＯＦＳＴとして記憶される。この場合、ユーザは、データ値ブレークポイントが発生すると、そのデータ値ブレークポイントを実際に引き起こしたのは、前の命令（本例では、Ｉ２）ではなく、Ｉ２から更に２命令前であることを知ることができる（ここで、この値「２」は、ＤＡＣ＿ＯＦＳＴに対応する）。Ｉ２から更に２命令前の命令は、Ｉ０であり、本例においてデータ値ブレークポイントを引き起こしたのは、これである。ＤＡＣ＿ＯＦＳＴビット９８に関して上述したように、ＤＶＣ　ＤＡＣが起こると、これは、ＤＡＣデバッグ例外を起こした命令（これは、Ｉ０である）のアドレスからのセーブしたＤＳＲＲ０値の「オフセット－１」を示す。Ｉ０からのセーブしたＤＳＲＲ０値（本例では、Ｉ３のアドレスに対応する）のオフセットは、３であり、従って、ＤＡＣ＿ＯＦＳＴビット９８は、２を示すが、これは、「３－１」である。他の実施形態では、ＤＡＣ＿ＯＦＳＴを計算するために異なる境界を用いることが可能であり、あるいは、異なる計数方法を用いて、ＤＡＣ＿ＯＦＳＴが本例のＩ０を示すようにしてもよいことに留意されたい。

【００２６】

図１０において、Ｉ０は、クロックサイクル１においてフェッチ段に入り、クロックサイクル２においてデコード段に入り、クロックサイクル３においてＥＡ／Ｅ０段に入り、ここでＩ０の有効アドレスを計算し、そして、クロックサイクル４においてｍｅｍ１／Ｅ１段に入る。有効アドレスは、導体３５のアドレス部でデバッグ制御回路４４に提供され、Ｉ０の有効アドレスとデータアドレス比較レジスタ４７に記憶されたＤＡＣ１値との間でアドレス比較を実施できるようにする。Ｉ０に対するこのＤＡＣ１アドレス比較は、クロックサイクル４において実施され、ＤＡＣ１アドレス比較信号の正のパルスによって示されるように、アドレス比較の一致が起こる。しかしながら、ＤＡＣ１は、ＤＶＣ１修飾子を必要とすることから（ＤＶＣ１ビット７０の設定により）、ＤＡＣ１Ｒデバッグイベントは、アドレス比較が一致してもまだ示されない。命令処理は、Ｉ０に関係する（Ｉ０ロード命令に応じて検索される）データが、まだデータ値比較に利用可能でないため、継続する。次に、Ｉ０は、ｍｅｍ２／Ｅ２段に進み、この間、データをメモリ（例えば、図１のメモリ１８等）から読み込む。しかしながら、クロックサイクル５では、Ｉ０用の読込データは、メモリ待ち状態によってストールされる（Ｉ０のストールは、パイプラインのストールになり、この場合、Ｉ１乃至Ｉ３は、各々、クロックサイクル５でストールすることに留意されたい）。従って、Ｉ０用の読込データは、クロックサイクル５の終わりに提供される（図９のように）のではなく、クロックサイクル６の終わりまで、導体３５のデータ部でデバッグ制御回路４４に提供されない。次に、データ値比較は、次のクロックサイクルであるクロックサイクル７において実施され、ＤＶＣ１データ比較信号の正のパルスによって示されるように、データの一致が起こる。データ値比較レジスタ５３に記憶されたＤＶＣ１値に一致すると、ＤＶＣ１　ＤＡＣ１デバッグイベントが示され、従って、ハードウェアは、ＤＡＣ１Ｒビット９０を設定する。

【００２７】

サイクル７内においてＩ０上でＤＶＣ１　ＤＡＣ１を検出すると、ＤＡＣオフセットカウンタが０にクリアされ、また、後続の各完了した命令の計数が開始できるようになることに留意されたい。しかしながら、パイプラインは、Ｉ１がメモリ段（ｍｅｍ２）にあるため、クロックサイクル７ではまだ中断できない。従って、Ｉ１は、メモリアクセスが開始されると、中断することはできず、完了しなければならないことから、待ち状態のデバッグ例外の更なる処理に先立って、これを完了しなければならない。後続のメモリアクセスは、しかしながら、ＤＶＣ１　ＤＡＣ１例外が現在待ち状態であることから、起動されない。しかしながら、図１０の例では、命令Ｉ２は、アクセス許可が破られ且つメモリがＩ２の読込データを何も返さないことを示すＤＳＩエラーを返す。従って、Ｉ２もメモリ段（ｍｅｍ１）にあるが、ＤＳＩエラーによりキルし得る。Ｉ３は、ＥＡ／Ｅ０段にあり、通常パイプライン段ｍｅｍ１で起こるメモリアクセスをまだ開始していないことから、同様にキルし得る。命令Ｉ１は、クロックサイクル８において書き戻し段にあり、この後

、待ち状態のデバッグ例外を更に処理することができ、これによりデバッグ例外を発生させて割込処理を開始し得る。即ち、ＤＶＣ１　ＤＡＣ１デバッグイベントの割込処理は、クロックサイクル８の後（例えば、クロックサイクル９）まで開始されない。従って、命令ストリームは、Ｉ２において中断される（即ち、デバッグ例外は、Ｉ１の実行後、Ｉ２の実行に先立って起こる）。そして、通常の実行は、デバッグ割込用の割込処理の完了後に、Ｉ２から再開し得る。従って、ＤＳＲＲ０レジスタは、Ｉ２のアドレスを記憶し得るが、その理由は、これが、データブレークポイントデバッグ割込の復帰ポインタに対応するためである。デバッグ割込ハンドラのソフトウェア・ルーチンの完了に続いて、通常の実行は、割込命令からの復帰を実行することによって再開されるが、通常の（非割込）命令実行を継続するために、復帰すべき命令（この場合、Ｉ２）へのポインタとしてＤＳＲＲ０にセーブした値が用いられる。

【００２８】

ＤＶＣ１　ＤＡＣ１を検出してＤＡＣオフセットカウンタをクリアした後、後続の各命令が完了した状態で、ＤＡＣオフセットカウンタは、デバッグ例外が発生する（この時点で割込処理が起こる）まで、１ずつ増加することに留意されたい。従って、本例では、ＤＡＣオフセットカウンタは、１だけ増加し、従って、デバッグ状態レジスタ４９のＤＡＣ＿ＯＦＳＴビット９８が％０１に設定される。即ち、ＤＶＣ１　ＤＡＣ１を引き起こした命令（本例では、Ｉ０）の後で且つデバッグイベント割込処理前に実行される命令の数は、１であり、これは、ＤＡＣ＿ＯＦＳＴとして記憶される。この場合、ユーザは、データ値ブレークポイントが発生すると、そのデータ値ブレークポイントを実際に引き起こしたのは、前の命令（本例では、Ｉ１）ではなく、Ｉ２から更に１命令前であることを知ることができる。Ｉ１から更に１命令前の命令は、Ｉ０であり、本例においてデータ値ブレークポイントを引き起こしたのは、これである。ＤＡＣ＿ＯＦＳＴビット９８に関して上述したように、ＤＶＣ　ＤＡＣが起こると、これは、ＤＡＣデバッグ例外を起こした命令（これは、Ｉ０である）のアドレスからのセーブしたＤＳＲＲ０値の「オフセット－１」を示す。Ｉ０からのセーブしたＤＳＲＲ０値（本例では、Ｉ２のアドレスに対応する）のオフセットは、２であり、従って、ＤＡＣ＿ＯＦＳＴビット９８は、１を示すが、これは、「２－１」である。他の実施形態では、ＤＡＣ＿ＯＦＳＴを計算するために異なる境界を用いることが可能であり、あるいは、異なる計数方法を用いて、ＤＡＣ＿ＯＦＳＴが本例のＩ０を示すようにしてもよいことに留意されたい。

【００２９】

図１１において、Ｉ０は、クロックサイクル１においてフェッチ段に入り、クロックサイクル２においてデコード段に入り、クロックサイクル３においてＥＡ／Ｅ０段に入り、ここでＩ０の有効アドレスを計算し、また、クロックサイクル４においてｍｅｍ１／Ｅ１段に入る。有効アドレスは、導体３５のアドレス部でデバッグ制御回路４４に提供され、Ｉ０の有効アドレスとデータアドレス比較レジスタ４７に記憶されたＤＡＣ１値との間でアドレス比較を実施できるようにする。このＤＡＣ１アドレス比較は、クロックサイクル４においてＩ０に対して実施され、ＤＡＣ１アドレス比較信号の正のパルスによって示されるように、アドレスの一致が起こる。しかしながら、ＤＡＣ１がＤＶＣ１修飾子を必要とすることから（ＤＶＣ１ビット７０の設定により）、ＤＡＣ１Ｒデバッグイベントは、アドレス比較で一致してもまだ示されない。命令処理は、Ｉ０に関係する（Ｉ０ロード命令に応じて検索される）データが、まだデータ値比較のために利用可能でないため、継続する。次に、Ｉ０は、ｍｅｍ２／Ｅ２段に進み、この間、データをメモリ（例えば、図１のメモリ１８等）から読み込む。Ｉ０用の読込データは、クロックサイクル５の終わりに準備が整っており、導体３５のデータ部でデバッグ制御回路４４に提供される。そして、Ｉ０のデータ値比較は、次のクロックサイクルであるクロックサイクル６において実施され、ＤＶＣ１データ比較信号の正のパルスによって示されるように、データの一致が起こる。データ値比較レジスタ５３に記憶したＤＶＣ１値に一致すると、ＤＶＣ１　ＤＡＣ１デバッグイベントが示され、従って、ハードウェアは、ＤＡＣ１Ｒビット９０を設定する。

【００３０】
サイクル６内においてＩ０上でＤＶＣ１　ＤＡＣ１を検出すると、ＤＡＣオフセットカウンタが０にクリアされ、また、後続の各完了した命令の計数が開始できるようになる。図１１の例では、Ｉ１が、メモリ段ｍｅｍ１において、変換索引バッファ（ＴＬＢ）のアドレス変換ミスによりエラーを引き起こし、従って、Ｉ１は、実行を継続しないことに留意されたい（尚、ＴＬＢは、図示しないが、当分野で公知なように、メモリ管理ユニット（ＭＭＵ）と共にプロセッサ１２に配置することができ、この場合、ＴＬＢ及びＭＭＵは、双方共、当分野で公知なように動作し得る）。従って、クロックサイクル６では、命令Ｉ１乃至Ｉ３の全てをキルし得る。従って、デバッグ例外は、クロックサイクル７の間に直ちに有効にすることができ、これにより、デバッグ例外を発生させることができ、割込処理を開始できる。即ち、命令Ｉ０に対するＤＶＣ１　ＤＡＣ１デバッグイベントの割込処理は、クロックサイクル６の後（例えば、クロックサイクル７において）開始し得る。従って、命令ストリームは、Ｉ１において中断される（即ち、デバッグ例外は、Ｉ０の実行後、Ｉ１の実行の前に起こる）。次に、実行は、割込処理の完了後、Ｉ１から再開し得る。従って、ＤＳＲＲ０レジスタは、Ｉ１のアドレスがデータブレークポイントデバッグ割込の復帰ポインタに対応することから、Ｉ１のアドレスを記憶し得る。本例では、ＤＡＣオフセットカウンタは、０のままである。従って、デバッグ状態レジスタ４９のＤＡＣ＿ＯＦＳＴビット９８は、％００のままである。即ち、ＤＶＣ１　ＤＡＣ１を引き起こした命令（本例では、Ｉ０）は、中断された命令（Ｉ１）の前の命令である。この時、通常の実行は、デバッグ割込用の割込処理が完了した後、Ｉ１から再開し得る。従って、ＤＳＲＲ０レジスタは、Ｉ１のアドレスがデータブレークポイントデバッグ割込の復帰ポインタに対応することから、Ｉ１のアドレスを記憶し得る。デバッグ割込ハンドラのソフトウェア・ルーチンの完了に続いて、通常の実行は、割込命令からの復帰を実行することによって再開されるが、通常の（非割込処理）命令実行を継続するために、復帰すべき命令（この場合、Ｉ１）へのポインタとしてＤＳＲＲ０にセーブした値が用いられる。

【００３１】
従って、ＤＶＣ　ＤＡＣを基準にしたデータ値ブレークポイントのタイミングは、様々な要因に依存して変動することがあり、それらの多くは、予測不可能であり、事前には分からないことに留意されたい。しかしながら、ＤＡＣ＿ＯＦＳＴを利用して、デバッグ例外が実際に発生すると、ユーザは、どの命令が実際にデータ値ブレークポイントを引き起こしたか（即ち、実際にデバッグ例外を発生させたか）判定し得る。

【００３２】
状況によっては、例えば、ユーザが、割込マスク制御でデバッグ割込を明示的にマスクして無効にした場合、ＩＤＥビット７６は、ユーザに対して、デバッグイベント（ＤＶＣ　ＤＡＣデバッグイベント等）が起こったことを通知するが、デバッグ例外がユーザによって一括して無効にされたことから、それが曖昧であることも通知する。即ち、たとえデバッグイベントが起こっても、デバッグ例外は、ユーザによって現在マスクされている。従って、デバッグ割込のマスクが解除された後、そのデバッグ例外が実際に起こると、ＤＡＣ＿ＯＦＳＴ値を用いて、どの命令がデバッグ例外を発生させたか正確に判定できない。これは、ユーザがデバッグ割込のマスクを再び解除する前に、多くの命令を実行し得るためである。この場合、ＩＤＥビット７６は、実行される命令の実際の数を反映しないことから、ＤＡＣ＿ＯＦＳＴの妥当性を限定するためにソフトウェアによって用いてよい。

【００３３】
図１２乃至１６は、ＤＡＣ及びＤＶＣ　ＤＡＣデバッグイベントの発生、並びに一実施形態における、例えば、ＤＡＣ＿ＯＦＳＴビット９８に対するデバッグ状態レジスタ４９の更新結果を示す様々な例を表形式で示す。図１２乃至１６の表では、一連の３つの命令Ｉ０、Ｉ１、及びＩ２を各例に用いる。第１命令Ｉ０は、ロード／記憶クラスの命令であり、第２命令Ｉ１は、他に規定されない限り、ロード／記憶クラス命令であり、また、第３命令Ｉ２は、他に規定されない限り、ロード／記憶命令である。図１２には、行２０１乃至２０８が含まれ、図１３には、行２０９乃至２１２が含まれ、図１４には、行２１３

乃至２１５が含まれ、図１５には、行２１６乃至２１８が含まれ、また、図１６には、行２１９乃至２２０が含まれることに留意されたい。

【００３４】

例を示す表の行２０１では、Ｉ０がＤＴＬＢエラーになり、ＤＡＣデバッグイベントが起こらないと仮定する。この場合、ＤＴＬＢ例外が起こり、デバッグ状態レジスタは、更新されない。行２０２が表す例では、Ｉ０が、データ記憶割込（ＤＳＩ）エラーになり、また、ＤＡＣデバッグイベントが起こらない。この場合、ＤＳＩ例外が生じ、デバッグ状態レジスタは、更新されない。行２０３が表す例では、Ｉ０が、ＤＴＬＢエラーになるが、ＤＡＣｘデバッグイベントが示される（例えば、ＤＡＣ１又はＤＡＣ２デバッグイベント）。この場合、ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、また、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＩＤＥビット７６が設定され、また、ＤＡＣ＿ＯＦＳＴビット９８が％００に設定される。ＤＳＲＲ０レジスタは、Ｉ０を指す（即ち、Ｉ０のアドレスを記憶する）。行２０４が表す例では、Ｉ０が、ＤＳＩエラーになるが、ＤＡＣｘデバッグイベントが示される（例えば、ＤＡＣ１又はＤＡＣ２デバッグイベント）。この場合、ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、また、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＩＤＥビット７６が設定され、また、ＤＡＣ＿ＯＦＳＴビット９８が、％００に設定される。ＤＳＲＲ０レジスタは、Ｉ０を指す（即ち、Ｉ０のアドレスを記憶する）。行２０５が表す例では、Ｉ０がＤＡＣｘデバッグイベント（例えば、ＤＡＣ１又はＤＡＣ２デバッグイベント）を発生させる。この場合、ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、また、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＤＡＣ＿ＯＦＳＴビット９８が、％００に設定される。ＤＳＲＲ０レジスタは、Ｉ１を指す（即ち、Ｉ１のアドレスを記憶する）。行２０３及び２０４に示された条件の場合、ＩＤＥビット７６が設定されるが、行２０５の場合、ＩＤＥビット７６は設定されないことに留意されたい。このことを用いて、行２０３及び２０４の％００のＤＡＣ＿ＯＦＳＴ設定値が、命令Ｉ０が割込を発生させたことを示していることが示され、また、ＤＡＣ１又はＤＡＣ２イベントによりＩ０上でデバッグ割込が起こる時、ＤＳＲＲ０のセーブしたプログラムカウンタ値は、Ｉ０を指しており、通常の位置（Ｉ１）ではないことが示される。これが起こる理由は、Ｉ０上には、同時にＤＴＬＢ又はＤＳＩ例外もあり、従って、Ｉ０は、実行が完了していないことから、デバッグ割込の後、再実行するべきためである。再実行時、ＤＡＣ１、ＤＡＣ２イベントは、ユーザによって無効にされることがあり、その場合、通常のＤＴＬＢ又はＤＳＩ例外が発生し、適切に処理される。しかしながら、行２０５の場合、命令Ｉ０は、実行を完了しており、従って、セーブしたＤＳＲＲ０値がＩ１を指し、ＤＡＣ＿ＯＦＳＴ値が％００であり、また、ＩＤＥがクリアされ、ＤＡＣ＿ＯＦＳＴ値が％００の場合のＤＳＲＲ０の通常の境界条件を示す。

【００３５】

行２０６乃至２２０が表す様々な例では、Ｉ０がＤＶＣｘ　ＤＡＣｘデバッグイベント（例えば、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベントのいずれか）を発生させる。行２０６が表す例では、Ｉ１は、例外を起こさず、任意の命令であってよく、また、Ｉ３は、例外を起こさず、ロード／記憶型の命令ではない。この場合、ＤＶＣｘ　ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、また、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＤＡＣ＿ＯＦＳＴビット９８が％０１に設定される。ＤＳＲＲ０レジスタは、Ｉ２の後の命令を指す。ＤＡＣ＿ＯＦＳＴビット９８を調べることによって、Ｉ０がＤＶＣｘ　ＤＡＣｘイベントを引き起こしたことを判定できる。

【００３６】

行２０７は、図９と同様な例を表し、この例では、Ｉ１が例外を起こさず、また、Ｉ３が例外を起こさないが、Ｉ３は、ロード／記憶クラス命令である。この場合、ＤＶＣｘ

ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＤＡＣ＿ＯＦＳＴビット９８が％１０に設定される。ＤＳＲＲ０レジスタは、Ｉ２の後の命令を指す。本例では、命令Ｉ２は、メモリアクセスを開始したことから、完了できるようにすべきである。行２０８は、図１１と同様な例を表し、この例では、Ｉ１は、ＤＴＬＢエラーを起こし、ＤＡＣを引き起こさない。この場合、ＤＶＣｘ　ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＤＡＣ＿ＯＦＳＴビット９８が％００に設定される。ＤＳＲＲ０レジスタは、Ｉ１が、ＤＴＬＢエラーを引き起こし、実行を完了しなかったことから、Ｉ１を指す。ＤＡＣ＿ＯＦＳＴビット９８を調べることによって、Ｉ０がＤＶＣｘ　ＤＡＣｘイベントを引き起こしたことを判定できる。

【００３７】

行２０９が表す例では、Ｉ１がＤＳＩエラーを起こし、ＤＡＣを引き起こさない。この場合、ＤＶＣｘ　ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＤＡＣ＿ＯＦＳＴビット９８が％００に設定される。ＤＳＲＲ０レジスタは、Ｉ１が、ＤＳＩ例外を引き起こし、実行を完了しなかったことから、Ｉ１を指す。行２１０が表す例では、Ｉ１がＤＴＬＢエラー及びＤＡＣｙ（例えば、ＤＡＣ１又はＤＡＣ２デバッグイベント）を発生させる。この場合、ＤＶＣｘ　ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＤＡＣ＿ＯＦＳＴビット９８が％００に設定される。ＤＳＲＲ０レジスタは、Ｉ１を指す。この場合、Ｉ１のＤＡＣｙイベントは、Ｉ１のＤＴＬＢエラーによりいずれにせよ再実行すべきであることから、報告されない。行２１１が表す例では、Ｉ１がＤＳＩエラー及びＤＡＣｙ（例えば、ＤＡＣ１又はＤＡＣ２デバッグイベント）を発生させる。この場合、ＤＶＣｘ　ＤＡＣｘデバッグイベントに起因するデバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、ＤＡＣ＿ＯＦＳＴビット９８が％００に設定される。ＤＳＲＲ０レジスタは、Ｉ１を指す。この場合、Ｉ１のＤＡＣｙイベントは、Ｉ１のＤＴＬＢエラーによりいずれにせよ再実行すべきであることから、報告されない。行２０８乃至２１１の例の場合、Ｉ１例外は、マスクされるが、このことは、実施例に依存するものであり、他のプロセッサでは異なり得ることに留意されたい。

【００３８】

行２１２が表す例では、Ｉ１がＤＡＣｙ（例えば、ＤＡＣ１又はＤＡＣ２デバッグイベント）を発生させる。この場合、デバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、対応するＤＡＣｙビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、また、ＤＡＣ＿ＯＦＳＴビット９８が％０１に設定される。ＤＳＲＲ０レジスタは、Ｉ２を指す。行２１３が表す例では、Ｉ１は、ＤＶＣｙ　ＤＡＣｙ（例えば、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベント）を引き起こし、また、通常のロード／記憶命令であり、Ｉ２は、ロード／記憶命令ではない。この場合、デバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、対応するＤＡＣｙビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、また、ＤＡＣ＿ＯＦＳＴビット９８が％０１に設定される。ＤＳＲＲ０レジスタは、Ｉ２を指す。この場合、ｘがｙに等しければ、デバッグ状態レジスタ及びＤＳＲＲ０に結果的に生じる状態は、「ＤＡＣｙなし」の場合（即ち、Ｉ２がＤＡＣｙを引き起こさない場合）と区別できなくなり得ることにも留意されたい。

【００３９】

行２１４が表す例では、Ｉ１は、ＤＶＣｙ　ＤＡＣｙ（例えば、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベント）を引き起こし、また、通常のロード／記憶命令であり、Ｉ２は、例外を引き起こさないロード／記憶命令である。この場合、デバッグ例外は、Ｉ２がメモリアクセスを開始したことから、その完了後に発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、対応するＤＡＣｙビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、また、ＤＡＣ＿ＯＦＳＴビット９８が％１０に設定される。ＤＳＲＲ０レジスタは、Ｉ２の後の命令を指す。この場合、ｘがｙに等しければ、デバッグ状態レジスタ及びＤＳＲＲ０に結果的に生じる状態は、「ＤＡＣｙなし」の場合（即ち、Ｉ２がＤＡＣｙを引き起こさない場合）と区別できなくなり得ることにも留意されたい。

【００４０】

行２１５が表す例では、Ｉ１は、ＤＶＣｙ　ＤＡＣｙ（例えば、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベント）を引き起こし、また、通常のロード／記憶命令であり、Ｉ２は、ＤＳＩエラーを発生させる。行２１６が表す例では、Ｉ１は、ＤＶＣｙ　ＤＡＣｙ（例えば、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベント）を引き起こし、また、通常のロード／記憶命令であり、Ｉ２は、ＤＴＬＢエラーを発生させる。これらの場合のいずれにおいても、デバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、対応するＤＡＣｙビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、また、ＤＡＣ＿ＯＦＳＴビット９８が％０１に設定される。ＤＳＲＲ０レジスタは、Ｉ２を指す。この場合、ｘがｙに等しければ、デバッグ状態レジスタ及びＤＳＲＲ０に結果として生じる状態は、「ＤＡＣｙなし」の場合（即ち、Ｉ２がＤＡＣｙを引き起こさない場合）と区別できなくなり得ることにも留意されたい。更に、これらの場合、命令Ｉ２は、マスクされるが、この振舞いは、実施例に依存するものであり、他のプロセッサでは異なり得ることに留意されたい。

【００４１】

行２１７が表す例では、Ｉ１は、ＤＶＣｙ　ＤＡＣｙ（例えば、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベント）を引き起こし、また、通常のロード／記憶命令であり、Ｉ２は、ＤＡＣｙ又はＤＶＣｙ　ＤＡＣｙを発生させる通常のロード／記憶命令又は複数ワードのロード／記憶命令である。この場合、デバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、対応するＤＡＣｙビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、また、ＤＡＣ＿ＯＦＳＴビット９８が％１０に設定される。ＤＳＲＲ０レジスタは、Ｉ２の後の命令を指す。この場合、ｘがｙに等しければ、デバッグ状態レジスタ及びＤＳＲＲ０に結果として生じる状態は、「ＤＡＣｙなし」の場合（即ち、Ｉ１又はＩ２がＤＡＣｙを引き起こさない場合）と区別できなくなり得ることにも留意されたい。

【００４２】

行２１８が表す例では、Ｉ１は、ＤＶＣｙ　ＤＡＣｙ（例えば、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２デバッグイベント）を引き起こし、また、複数ワードのロード／記憶命令であり、Ｉ２は、任意の命令である。この場合、デバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、対応するＤＡＣｙビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、また、ＤＡＣ＿ＯＦＳＴビット９８が％０１に設定される。ＤＳＲＲ０レジスタは、Ｉ２を指す。この場合、Ｉ１は、複数のレジスタに読込又は書込を行うことから、Ｉ２のメモリへのアクセスを防止するのに充分な時間があり、従って、Ｉ０にＤＶＣｘ　ＤＡＣｘの条件が現れると、それをキルし得る。この場合、ｘがｙに等しければ、デバッグ状態レジスタ及びＤＳＲＲ０に結果として生じる状態は、「ＤＡＣｙなし」の場合（即ち、Ｉ２がＤＡＣｙを引き起こさない場合）と区別

できなくなり得ることにも留意されたい。

【００４３】

行２１９が表す例では、Ｉ１は、任意の命令であり、また、例外を起こさず、Ｉ２は、ＤＳＩエラーを発生させる通常のロード／記憶又は複数ワードのロード／記憶命令であって、ＤＡＣを発生させる場合も発生させない場合もある命令である。この場合、デバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、対応するＤＡＣｙビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、また、ＤＡＣ＿ＯＦＳＴビット９８が％０１に設定される。ＤＳＲＲ０レジスタは、Ｉ２を指す。この場合、命令Ｉ２は、マスクされるが、この振舞いは、実施例に依存するものであり、他のプロセッサでは異なり得ることに留意されたい。

【００４４】

行２２０が表す例では、Ｉ１は、任意の命令であり、また、例外を起こさず、Ｉ２は、ＤＡＣｙ又はＤＶＣｙ　ＤＡＣｙを発生させる通常のロード／記憶又は複数ワードのロード／記憶命令である。この場合、デバッグ例外が発生し、そして、対応するＤＡＣｘビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、対応するＤＡＣｙビットフィールドが設定され（例えば、ＤＡＣ１Ｒ、ＤＡＣ１Ｗ、ＤＡＣ２Ｒ、ＤＡＣ２Ｗ）、また、ＤＡＣ＿ＯＦＳＴビット９８が％１０に設定される。ＤＳＲＲ０レジスタは、Ｉ２の後の命令を指す。この場合、ｘがｙに等しければ、デバッグ状態レジスタ及びＤＳＲＲ０に結果として生じる状態は、「ＤＡＣｙなし」の場合（即ち、Ｉ２がＤＡＣｙを引き起こさない場合）と区別できなくなり得ることにも留意されたい。

【００４５】

図１７は、本発明の他の実施形態に基づく、図１のデータ処理システムに関係するデバッグ状態レジスタ４９の図である。図１７の例では、デバッグ状態レジスタ４９には、ＩＤＥ（７６）、ＩＣＭＰ（７８）、ＢＲＴ（８０）、ＩＡＣ１（８２）、ＩＡＣ２（８４）、ＩＡＣ３（８６）、ＩＡＣ４（８８）、ＤＡＣ１Ｒ（９０）、ＤＡＣ１Ｗ（９２）、ＤＡＣ２Ｒ（９４）、ＤＡＣ２Ｗ（９６）、ＤＡＣ＿ＯＦＳＴＡ（１０２）、ＯＣＣＡ（１０４）、ＤＡＣ＿ＯＦＳＴＢ（１０６）、及びＯＣＣＢ（１０８）と表記したビットフィールドが含まれる。図１７の例には、更に、将来の用途のために予約し得る予約ビットフィールド１１０を含み得る。尚、デバッグ状態レジスタ４９、並びにビットフィールドＩＤＥ（７６）、ＩＣＭＰ（７８）、ＢＲＴ（８０）、ＩＡＣ１（８２）、ＩＡＣ２（８４）、ＩＡＣ３（８６）、ＩＡＣ４（８８）、ＤＡＣ１Ｒ（９０）、ＤＡＣ１Ｗ（９２）、ＤＡＣ２Ｒ（９４）、及びＤＡＣ２Ｗ（９６）に関する上記説明（例えば、図７及び８の説明等）は、図１７に同様に当てはまり、従って、ここでは繰り返さない。しかしながら、図７の例と異なり、図１７の例には、ＤＡＣ＿ＯＦＳＴ（９８）ではなく複数のＤＡＣオフセットフィールド（例えば、ＤＡＣ＿ＯＦＳＴＡ（１０２）及びＤＡＣ＿ＯＦＳＴＢ（１０６））が含まれる。複数のＤＡＣオフセットフィールドを用いると、複数のオフセットを追跡することができ、この場合、各ＤＡＣオフセットフィールドは、ＤＡＣ＿ＯＦＳＴ（９８）に関して上述したように動作し得る。しかしながら、複数のＤＡＣオフセットフィールドが存在することにより、各ＤＡＣオフセットフィールド（例えば、ＤＡＣ＿ＯＦＳＴＡ（１０２）及びＤＡＣ＿ＯＦＳＴＢ（１０６））は、対応する発生フィールドを有する（例えば、それぞれＯＣＣＡ（１０４）及びＯＣＣＢ（１０８））。

【００４６】

従って、ＤＡＣ＿ＯＦＳＴＡビット１０２を用いると、第１ＤＶＣ　ＤＡＣ（ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２のいずれか）を示させた命令と、データ値ブレークポイントが起こり、ＤＶＣ　ＤＡＣ用のデバッグ例外が発生するポイントとの間に実行した命令の数を示し得る。ＤＡＣ＿ＯＦＳＴＡビット１０２を用いると、ＤＡＣ＿ＯＦＳＴ（９８）と同様に、第１データアドレス比較デバッグ例外を起こしたロード又は記憶命令のアドレスからのセーブしたＤＳＲＲ０値の「オフセット－１」を記憶し得る。上述したよ

うに、ＤＳＲＲ０は、デバッグセーブ復旧レジスタ０に対応し、セーブしたＤＳＲＲ０値は、デバッグ割込用の復帰ポインタに対応する。このようにして、第１ＤＶＣ　ＤＡＣを引き起こしたアドレスと、命令ストリームが中断されてデータ値ブレークポイントが発生したポイント（この場合、そのデータ値ブレークポイントが、第１ＤＶＣ　ＤＡＣ又は他のＤＶＣ　ＤＡＣの結果として発生し得る）との間の期間内に幾つの命令が実行されたかを判定し得る。これによって、ユーザは、どの命令が第１ＤＶＣ　ＤＡＣを引き起こしたかを判定し得る。一実施形態では、ＤＡＣ＿ＯＦＳＴＡビット１０２は、通常、％００に設定され、ＤＶＣ　ＤＡＣが、このフィールドを％００、％０１、％１０、又は％１１に設定するが、これは、「オフセット－１」を表す。ＤＡＣ＿ＯＦＳＴＡビット１０２は、既定値として通常％００に設定されることから、ＯＣＣＡ（１０４）は、第１ＤＶＣ　ＤＡＣの発生時に設定され、ＤＡＣ＿ＯＦＳＴＡが、ＤＶＣ　ＤＡＣオフセットを示す値を保持していることを示し得る。ＤＡＣ＿ＯＦＳＴＢビット１０６を用いると、従って、第２ＤＶＣ　ＤＡＣを示させた命令と、データ値ブレークポイントが起こり、ＤＶＣ　ＤＡＣ用のデバッグ例外が発生するポイントとの間に実行される命令の数を示し得る。ＤＡＣ＿ＯＦＳＴＢビット１０６を用いると、第２データアドレス比較デバッグ例外を起こしたロード又は記憶命令のアドレスからのセーブしたＤＳＲＲ０値の「オフセット－１」を記憶し得る。このようにして、第２ＤＶＣ　ＤＡＣを引き起こしたアドレスと、命令ストリームが中断されてデータ値ブレークポイントが発生したポイント（ここでもまた、そのデータ値ブレークポイントが、第１ＤＶＣ　ＤＡＣ又は第２ＤＶＣ　ＤＡＣの結果として発生し得る）との間の期間内に幾つの命令が実行されたかを判定し得る。一実施形態では、ＤＡＣ＿ＯＦＳＴＢビット１０６は、通常、％００に設定され、ＤＶＣ　ＤＡＣが、このフィールドを％００、％０１、％１０、又は％１１に設定するが、これは、「オフセット－１」を表す。ＤＡＣ＿ＯＦＳＴＢビット１０６は、通常、既定値として％００に設定されることから、ＯＣＣＢ（１０８）は、第２ＤＶＣ　ＤＡＣの発生時に設定し得る（更に、上述したように、ＤＶＣ　ＤＡＣに対して同時にＤＴＬＢエラー又はＤＳＩエラーが起こる場合、ＤＡＣ＿ＯＦＳＴビット９８に関して上述したように、対応するＤＡＣオフセットビットは、％００に設定され、ＩＤＥビット７６は、１に設定され得ることに留意されたい）。

【００４７】

従って、複数の命令が、ＤＶＣ　ＤＡＣをもたらすことがあり、また、命令ストリームが中断されてデータ値ブレークポイントが発生するポイントにおいて、オフセットフィールドＤＡＣ＿ＯＦＳＴＡ（１０２）及びＤＡＣ＿ＯＦＳＴＢ（１０４）の各々は、対応する発生ビットが設定されていると仮定すると、命令ストリームのどの命令（１つ又は複数）が、データ値ブレークポイントをもたらし得るＤＶＣ　ＤＡＣを引き起こしたかを示せることに留意されたい。ＯＣＣＡビット１０４及びＯＣＣＢビット１０８の双方が設定され、２つのＤＶＣ　ＤＡＣが発生したことを示す場合、ユーザは、どの命令が実際に各ＤＶＣ　ＤＡＣを引き起こしたかを示すために、対応するオフセット値を用い得るが、これら２つの命令の内のどちらが（即ち、２つのＤＶＣ　ＤＡＣの内のどちらが）命令ストリームを中断したデータ値ブレークポイントを実際にもたらしたかについてユーザが実際に判定することが必要なこともある。

【００４８】

図１８は、図１７について述べたように、複数のオフセットフィールドの使用方法を示すタイミング図を表す。図１８の例の場合、Ｉ０乃至Ｉ４の各々は、ロード命令であり、Ｉ０は、第１ＤＡＣ　ＤＶＣデバッグイベント（ＤＶＣｘ　ＤＡＣｘ、ここで、ｘは、１又は２のいずれかであり得る）を発生させ、Ｉ２は、第２ＤＡＣ　ＤＶＣデバッグイベント（ＤＶＣｙ　ＤＡＣｙ、ここで、ｙは、１又は２のいずれかであり得る）を発生させるものと仮定する。図１８は、更に、六段パイプラインについて描かれているが、他の実施形態には、例えば、異なる数の段を有する異なるパイプラインを含んでもよい。図１８で参照する六段パイプライン等の命令パイプラインの動作は、当分野で公知であり、従って、本明細書では詳述しない。更に、図１８には、プロセッサ１２のプロセッサ・クロック

に対応し得るクロック信号が含まれる。プロセッサ・クロックは、システムクロック又はプロセッサ１２の一部だけに供給されるクロックであってよく、当分野で公知なように生成し得る。

【００４９】

図１８において、Ｉ０は、クロックサイクル１においてフェッチ段に入り、クロックサイクル２においてデコード段に入り、クロックサイクル３においてＥＡ／Ｅ０段に入り、ここでＩ０の有効アドレスを計算し、そして、クロックサイクル４においてｍｅｍ１／Ｅ１段に入る。従って、有効アドレスは、クロックサイクル３の終わりに準備が整っており、導体３５のアドレス部でデバッグ制御回路４４に提供され、Ｉ０の有効アドレスとデータアドレス比較レジスタ４７に記憶したＤＡＣ値（ＤＡＣ１及びＤＡＣ２）との間でアドレス比較を実施できるようにする（更に、この時点で、ＢＩＵ（３４）によってバス２０上にもアドレスを配置し得ることに留意されたい）。このＤＡＣｘアドレス比較は、ＤＡＣアドレス比較信号の正のパルスによって示されるように、クロックサイクル４において命令Ｉ０に対して実施され、一致が起こるが、ＤＡＣアドレス比較信号は、ＤＡＣｘアドレス比較が実施され、一致が起こっていることを示すためにデバッグ制御回路４４内でアサートされる制御信号であってよい。しかしながら、ＤＡＣｘは、ＤＶＣｘ修飾子を必要とすることから（ＤＶＣｘビットの設定、例えば、ＤＶＣ１ビット７０又はＤＶＣ２ビット７２の設定により）、デバッグイベントは、ＤＡＣｘアドレス比較で一致しても、まだ示されない。命令処理は、Ｉ０に関係する（Ｉ０ロード命令に応じて検索される）データが、まだデータ値比較のために利用可能でないため、継続する。次に、Ｉ０は、ｍｅｍ２／Ｅ２段に進み、この間、データをメモリ（例えば、図１のメモリ１８等）から読み込む。この読み込んだデータは、この段の終わりで利用可能となり得る（また、例えば、バス２０を介してＢＩＵ（３４）によって受信される）。そして、導体３５のデータ部でデバッグ制御回路４４に提供され、Ｉ０に関連する読み込んだデータ値と、データ値比較レジスタ５３に記憶したＤＶＣｘ値との間でデータ値比較を実施できるようにする。従って、データ値比較は、ＤＶＣｘデータ比較信号の正のパルスによって示されるように、次のクロックサイクルであるクロックサイクル６において実施され、一致が起こるが、ＤＶＣｘデータ比較信号は、ＤＶＣデータ比較が起こっていることを示すためにデバッグ制御回路４４内でアサートされる制御信号であってよい。データ値比較レジスタ５３に記憶したＤＶＣｘ値に一致すると、ＤＶＣｘ　ＤＡＣｘデバッグイベントが示され、従って、ハードウェアは、対応する状態ビットを設定する（例えば、ＤＡＣ１Ｒビット９０又はＤＡＣ２Ｒビット９４。Ｉ０がＤＡＣ１　ＤＶＣ１を引き起こしたか又はＤＡＣ２　ＤＶＣ２を引き起こしたかに依存する）。

【００５０】

サイクル６内においてＩ０上でＤＶＣｘ　ＤＡＣｘを検出すると、ＤＡＣオフセットＡカウンタを０にクリアし、また、後続の各完了した命令の計数を開始できるようにすることに留意されたい（ＤＡＣオフセットＡカウンタは、カウンタ４１に含み得る）。しかしながら、パイプラインは、Ｉ１及びＩ２がメモリ段（ｍｅｍ１及びｍｅｍ２）にあるため、クロックサイクル６ではまだ中断できない。従って、Ｉ１及びＩ２は、メモリアクセスが開始すると、中断することはできず、完了しなければならないことから、待ち状態のデバッグ例外の更なる処理に先立って、完了しなければならない。後続のメモリアクセスは、しかしながら、ＤＶＣｘ　ＤＡＣｘ例外が現在待ち状態であることから、起動されない。Ｉ１及びＩ２の各メモリアクセスは、書き戻し段に入った後、完了と見なされる。Ｉ２（命令Ｉ１及びＩ２の後半）は、クロックサイクル８において書き戻し段にあり、その後、デバッグ例外が発生し、また、割込処理を開始できるように、待ち状態のデバッグ例外を処理し得ることに留意されたい。即ち、ＤＶＣ１　ＤＡＣ１デバッグイベントの割込処理は、クロックサイクル８の後（例えば、クロックサイクル９）まで、開始できない。ＤＶＣｘ　ＤＡＣｘが検出される時点で、Ｉ３は、まだパイプラインのＥＡ／Ｅ０段にあり、従って、キルし得ることに留意されたい。即ち、命令ストリームは、Ｉ３において中断される。実行は、従って、割込処理の完了後に、Ｉ３から再開し得る。従って、ＤＳＲＲ

０レジスタは、Ｉ３のアドレスがデータブレークポイントデバッグ割込の復帰ポインタに対応することから、Ｉ３のアドレスを記憶し得る。デバッグ割込ハンドラのソフトウェア・ルーチンの完了に続いて、通常の実行は、割込命令からの復帰を実行することによって再開されるが、通常の（非割込）命令実行を継続するために、復帰すべき命令（この場合、Ｉ３）へのポインタとしてＤＳＲＲ０にセーブした値が用いられる。

【００５１】

ＤＶＣｘ　ＤＡＣｘデバッグイベントを検出してＤＡＣオフセットＡカウンタをクリアした後、後続の各命令が完了した状態で、ＤＡＣオフセットＡカウンタは、デバッグ例外が発生する（この時点で割込処理が起こる）まで、１ずつ増加することに留意されたい。従って、ＤＡＣオフセットＡカウンタは、クロックサイクル８において２まで増加し、従って、デバッグ状態レジスタ４９のＤＡＣ＿ＯＦＳＴＡビット１０２が％１０に設定され、更に、ＯＣＣＡビット１０４が設定される。即ち、ＤＶＣｘ　ＤＡＣｘを引き起こした命令（本例では、Ｉ０）の後で且つ割込処理の前に実行される命令の数は、２であり、これは、ＤＡＣ＿ＯＦＳＴＡとして記憶される。

【００５２】

Ｉ２は、クロックサイクル５においてＥＡ／Ｅ０段に入ることに留意されたい。従って、有効アドレスは、クロックサイクル５の終わりに準備が整っており、導体３５のアドレス部でデバッグ制御回路４４に提供され、Ｉ２の有効アドレスとデータアドレス比較レジスタ４７に記憶したＤＡＣ値（ＤＡＣ１及びＤＡＣ２）との間でアドレス比較を実施できるようにする（更に、この時点で、ＢＩＵ（３４）によってバス２０上にもアドレスを配置し得ることに留意されたい）。このＤＡＣｙアドレス比較は、ＤＡＣｙアドレス比較信号の正のパルスによって示されるように、クロックサイクル６において実施され、一致が起こるが、ＤＡＣｙアドレス比較信号は、ＤＡＣアドレス比較が実施されて一致が起こっていることを示すためにデバッグ制御回路４４内でアサートされる制御信号であってよい。しかしながら、ＤＡＣｙは、ＤＶＣｙ修飾子を必要とすることから（ＤＶＣｙビット、例えば、ＤＶＣ１ビット７０又はＤＶＣ２ビット７２の設定により）、デバッグイベントは、ＤＡＣｙアドレス比較で一致しても、まだ示されない。命令処理は、Ｉ２に関係する（Ｉ２ロード命令に応じて検索される）データが、まだデータ値比較のために利用可能でないため、継続する。次に、Ｉ２は、ｍｅｍ２／Ｅ２段に進み、この間、データをメモリ（例えば、図１のメモリ１８等）から読み込む。この読み込んだデータは、この段の終わりで利用可能となり得る（また、例えば、バス２０を介してＢＩＵ（３４）によって受信し得る）。そして、導体３５のデータ部でデバッグ制御回路４４に提供され、Ｉ２に関連する読み込んだデータ値と、データ値比較レジスタ５３に記憶したＤＶＣｙ値との間でデータ値比較を実施できるようにする。従って、データ値比較は、ＤＶＣｙデータ比較信号の正のパルスによって示されるように、次のクロックサイクルであるクロックサイクル８において実施され、一致が起こるが、ＤＶＣｙデータ比較信号は、ＤＶＣデータ比較が実施されて一致が起こっていることを示すためにデバッグ制御回路４４内でアサートされる制御信号であってよい。データ値比較レジスタ５３に記憶したＤＶＣｙ値に一致すると、ＤＶＣｙ　ＤＡＣｙデバッグイベントが示され、従って、ハードウェアは、対応する状態ビット（例えば、ＤＡＣ１Ｒビット９０又はＤＡＣ２Ｒビット９４。Ｉ２がＤＡＣ１　ＤＶＣ１を引き起こしたか又はＤＡＣ２　ＤＶＣ２を引き起こしたかに依存する）。

【００５３】

サイクル８内においてＩ２上でＤＶＣｙ　ＤＡＣｙを検出すると、ＤＡＣオフセットＢカウンタを０にクリアし、また、後続の各完了した命令の計数を開始できるようにすることに留意されたい（ＤＡＣオフセットＢカウンタは、カウンタ４１に含み得る）。パイプラインは、クロックサイクル８において、まだ中断されていない。その理由は、Ｉ０によって引き起こされたＤＶＣｘ　ＤＡＣｘに応じてデータ値ブレークポイントが発生する前に、Ｉ１及びＩ２の双方が完了しなければならないためである。従って、Ｉ２に対応するデータも返され、ＤＶＣｙ　ＤＡＣｙがデバッグ例外の発生前に起こり得る。上述したように、命令ストリームは、サイクル８の後、Ｉ３において中断され、これは、ＤＡＣオフ

セットＢカウンタが、増加せず、デバッグ例外発生時に０のままであることを意味する。従って、デバッグ状態レジスタ４９のＤＡＣ＿ＯＦＳＴＢビット１０６が％００に設定され、更に、ＯＣＣＢビット１０８が設定される（これにより、ユーザは、％００が、実際のオフセット値を示すものであって、第２ＤＶＣ　ＤＡＣが起こらなかった場合に返される単なる既定値ではないことを知ることができる）。従って、ＤＶＣｙ　ＤＡＣｙを引き起こした命令（本例では、Ｉ２）の後で且つ割込処理前に実行される命令の数は、０である。このようにして、ＤＡＣ＿ＯＦＳＴＢビット１０６及びＯＣＣＢビット１０８は、第１デバッグイベント（例えば、ＤＶＣｘ　ＤＡＣｘ）と、その第１デバッグイベントのデバッグ例外が起こるポイントとの間で起こるデバッグイベント（例えば、ＤＶＣｙ　ＤＡＣｙ）に関する情報を提供し得ることに留意されたい。

【００５４】

従って、図１８の例では、各々異なる命令によって引き起こされた２つのＤＶＣ　ＤＡＣが、サイクル８の後、デバッグ例外の発生の前に起こることに留意されたい。ＤＡＣ＿ＯＦＳＴＡビット１０２及びＤＡＣ＿ＯＦＳＴＢビット１０４を、対応するＯＣＣＡビット１０４及びＯＣＣＢビット１０８と共に用いて、Ｉ０（命令ストリームが中断したポイントに対応する前の命令Ｉ２から更に２命令前）及びＩ２（命令ストリームが中断したポイントに対応する前の命令Ｉ２から更に０命令前）の各々がＤＶＣ　ＤＡＣを引き起こしたことが分かる。そして、ユーザは、どちらの命令が実際にデータ値ブレークポイントをもたらしたかをデコードし得る（本例では、Ｉ０であり、これが第１ＤＶＣ　ＤＡＣ、ＤＶＣｘ　ＤＡＣｘを引き起こした）。更に、Ｉ０がＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２の１つを引き起こし、Ｉ２がＤＶＣ１　ＤＡＣ１及びＤＶＣ２　ＤＡＣ２の他方を発生させるように、ｘ及びｙが異なってよいことに留意されたい。他の選択肢として、Ｉ０及びＩ２の双方が、ＤＶＣ１　ＤＡＣ１又はＤＶＣ２　ＤＡＣ２のいずれかを発生させるように、ｘがｙに等しくてもよい。更に、一実施形態では、ＤＡＣ　Ａオフセットカウンタ及びＤＡＣ　Ｂオフセットカウンタは、オフセットカウンタ（１つ又は複数）４１内に配置し得ることに留意されたい。

【００５５】

従って、デバッグイベントの発生とデバッグ例外の発生との間（即ち、デバッグイベントの発生とそのデバッグイベントの割込処理の開始との間）で実行する命令の数の判定を支援するために、どのように１つ又は複数のオフセット値を用いるかについて認識することができる。これによって、デバッグ処理を改善でき、ユーザは、これらの１つ又は複数のオフセット値を用いて、実際にどの命令がデバッグ例外を発生させたか良く理解できるようになる。上記の説明は、データ値ブレークポイントに帰着するＤＶＣ　ＤＡＣデバッグイベントに関して行ったが、これらのオフセット値（ＤＡＣ＿ＯＦＳＴ、ＤＡＣ＿ＯＦＳＴＡ、及びＤＡＣ＿ＯＦＳＴＢ等）は、他の種類のデバッグイベントに用いてよい。即ち、ＤＡＣ＿ＯＦＳＴ、ＤＡＣ＿ＯＦＳＴＡ、及びＤＡＣ＿ＯＦＳＴＢ等のオフセットフィールドは、任意の種類のデバッグイベントの発生と、対応するデバッグ例外の発生との間（即ち、任意の種類のデバッグイベント発生と、特定のデバッグイベントの割込処理開始との間）に実行する命令の数を示すために用い得る。

【００５６】

更に、デバッグイベントを引き起こした命令の位置ではなく、そのイベントが起こってから実行される命令の数を知ることは、有用であり得ることに留意されたい。その理由は、待ち状態のデバッグイベントの場合、命令ストリームを中断できるパイプライン境界に達する前に、（同じ位置の）同じ命令を何回も実行可能であり、従って、位置情報が、それ自体では、実際の一連のイベントを判定するには不充分な場合があり得ることを理解できるためである。

【００５７】

一実施形態では、システムは、複数の命令について、フェッチ、デコード、および実行と、各命令の実行に関連した結果の書込とを連続的に行うことによって、複数の命令を実行するためのパイプライン・プロセッサと、パイプライン・プロセッサに結合されており

、複数の命令の実行を監視してデバッグイベントが発生する時を判定し、デバッグ例外を発生させて命令処理フローを中断するためのデバッグ回路と、を含む。デバッグ回路は、デバッグイベントを引き起こした命令と命令実行においてデバッグ例外が発生するポイントとの間で命令実行を完了させる命令（存在する場合）の数を表す値を示すための制御回路をさらに含む。

【００５８】

更なる一実施形態では、デバッグ回路の制御回路は、デバッグイベントを引き起こした命令に続き命令実行を完了させる命令の数を表す値としてカウント値を計数して与えるためのカウンタをさらに含む。更に他の実施形態では、カウント値が０より大きいことはデバッグイベントの処理が曖昧であることを示す。

【００５９】

更なる他の実施形態では、デバッグ回路は、デバッグイベントを引き起こした命令に続き命令実行を完了させる１つ以上の追加の命令もデバッグイベントの発生を引き起こしたことを示す。

【００６０】

更なる他の実施形態では、パイプライン・プロセッサはデバッグイベントを各々引き起こす２つ以上の命令を実行し、制御回路は複数のカウンタをさらに含み、各カウンタは、デバッグイベントを引き起こす２つ以上の命令のそれぞれの命令に続き命令実行を完了させる追加の命令の数によって決定される、それぞれのカウント値を与える。

【００６１】

更なる他の実施形態では、デバッグ回路は、デバッグイベントを引き起こした命令に続き命令実行においてデバッグ例外が発生する所定のポイントまでに命令実行を完了させる追加の命令（存在する場合）の数を表す値を保持するためのフィールドを有するデバッグ状態レジスタをさらに含む。

【００６２】

更なる他の実施形態では、デバッグ回路は、複数の命令のうちの１つのデコードを行い、パイプライン・プロセッサによって形成されたアドレスを１つ以上の所定のデバッグアドレスと比較してデータアドレス一致が発生したか否かを判定し、パイプライン・プロセッサによってアクセスの行われたデータ値を１つ以上の所定のデータ値と比較してデータ値一致が発生したか否かを判定することとに応じて、デバッグイベントが発生したことを判定し、デバッグイベントは両比較演算について一致が発生することを必要とする。

【００６３】

他の実施形態では、システムは、複数の命令について、フェッチ、デコード、および実行と、各命令の実行に関連した結果の書込とを連続的に行うことによって、複数の命令を実行するためのパイプライン処理回路と、パイプライン処理回路に結合されており、１つ以上のデバッグアドレスを命令実行によって形成されるアドレスと選択的に比較することと、１つ以上のデバッグデータ値を命令実行によって形成されるデータと選択的に比較することとによって、複数の命令の実行を監視するデバッグ回路と、を含む。デバッグ回路は、両方の比較が一致を生じるときを判定し、該判定に応答してデバッグイベントを示し、デバッグイベントを引き起こした命令の後に命令実行においてデータ値ブレークポイントが発生する所定のポイントまでに実行される命令の数を示す。

【００６４】

他の実施形態の更なる実施形態では、データ値ブレークポイントは曖昧であり、試験回路は、曖昧なデータ値ブレークポイントが発生したか否かを示すビットフィールドを有するレジスタをさらに含む。

【００６５】

他の実施形態の更なる他の実施形態では、試験回路は、デバッグイベントが発生する時とデータ値ブレークポイントが発生する時との間に発生するイベントに関する状態情報を示すための状態レジスタをさらに含む。更に他の実施形態では、デバッグイベントが発生する時とデータ値ブレークポイントが発生する時との間に発生するイベントのうちの１つ

は、第２のデバッグイベントをさらに含む。

【００６６】

他の実施形態の更なる他の実施形態では、デバッグ回路は、デバッグイベントを引き起こした命令に続き命令実行を完了させる命令の数のカウント値を計数して与えるためのカウンタをさらに含む。

【００６７】

他の実施形態の更なる他の実施形態では、デバッグ回路は、デバッグイベントを引き起こした命令に続き、命令実行においてデータ値ブレークポイントが発生する所定のポイントまでに命令実行を完了させる追加の命令（存在する場合）の数を示す値を保持するためのフィールドを有する状態デバッグレジスタをさらに含む。

【００６８】

更に他の実施形態では、方法は、複数の命令について、フェッチ、デコード、および実行と、各命令の実行に関連した結果の書込とを連続的に行うことによって、複数の命令を実行する工程と、複数の命令の実行を監視してデバッグイベントが発生する時を判定する工程と、デバッグ例外を発生させて命令処理フローを中断する工程と、デバッグイベントを引き起こした命令と命令実行においてデバッグ例外が発生するポイントとの間で命令実行を完了させる命令（存在する場合）の数を示す工程と、を含む。

【００６９】

更に他の実施形態の更なる実施形態では、方法は、デバッグイベントを引き起こした命令に続き命令実行を完了させる命令の数のカウント値を計数して与える工程をさらに含む。更なる実施形態では、方法は、デバッグイベントの処理が曖昧であることを示す０より大きいカウント値を形成する工程をさらに含む。

【００７０】

他の実施形態の更なる他の実施形態では、方法は、デバッグイベントを引き起こした命令に続き命令実行を完了させる１つ以上の追加の命令もデバッグイベントの発生を引き起こしたことを判定する工程をさらに含む。

【００７１】

他の実施形態の更なる他の実施形態では、方法は、２つ以上の命令が各々デバッグイベントを引き起こすことを判定する工程と、２つ以上の命令の各々に続き命令実行を完了させる追加の命令の数のそれぞれのカウント値を異なるカウンタに記憶する工程と、をさらに含む。

【００７２】

他の実施形態の更なる他の実施形態では、方法は、デバッグ回路にデバッグ状態レジスタを与える工程と、状態レジスタは、デバッグイベントを引き起こした命令に続き命令実行においてデバッグ例外が発生する所定のポイントまでに命令実行を完了させる追加の命令（存在する場合）の数を示すための値を保持するためのフィールドを有することと、をさらに含む。

【００７３】

他の実施形態の更なる他の実施形態では、方法は、複数の命令のうちの１つのデコードを行うことに応答して、デバッグイベントが発生したと判定する工程と、パイプライン・プロセッサによって形成されるアドレスを１つ以上の所定のデバッグアドレスと比較してデータアドレス一致が発生したか否かを判定する工程と、パイプライン・プロセッサによって形成されるデータ値を１つ以上の所定のデータ値と比較してデータ値一致が発生したか否かを判定する工程と、デバッグイベントは両比較演算について一致が発生することを必要とすることと、をさらに含む。

【００７４】

本発明を実装する装置は、ほとんど当業者には既知の電子部品や回路から構成されるため、回路の詳細については、本発明の根底にある概念の理解や認識のために、また、本発明の教示内容が不明瞭になったり注意が逸れたりしないように、上述した如く必要と思われる以上の説明はしない。

【００７５】
本明細書に用いる用語「プログラム」は、コンピュータシステム上での実行用に設計された一連の命令と定義する。プログラム、即ち、コンピュータプログラムには、サブルーチン、関数、手順、オブジェクト・メソッド、オブジェクト実装、実行可能なアプリケーション、アプレット、サーブレット、ソースコード、オブジェクトコード、共有ライブラリ／動的読込ライブラリ、及び／又はコンピュータシステム上での実行用に設計された他の一連の命令を含み得る。
【００７６】
上述した実施形態の一部は、適用可能であれば、様々な異なる情報処理システムを用いて実現し得る。例えば、図１及びそれについての説明では、代表的な情報処理構成について述べるが、この代表的な構成は、本発明の様々な態様の説明において有用な基準を提供するためだけに示している。勿論、構成の説明は、説明のために簡素化しており、また、この構成は、本発明に基づいて用い得る多くの異なる種類の適切な構成の内の１つに過ぎない。当業者は、論理ブロック間の境界は、説明のためだけであり、他の実施形態では、論理ブロック又は回路素子を併合したり、様々な論理ブロック又は回路素子に機能を分解して代替させたりしてよいことを認識されるであろう。
【００７７】
従って、本明細書に示した構成は代表例に過ぎず、実際には、同じ機能を実現する任意の他の構成を実装し得ることを理解されたい。抽象的ではあるが明確に、同じ機能を達成する構成要素の任意の構成は、所望の機能を達成するように有効に「関連付けられる」。従って、本明細書において特定の機能を達成するために組み合わせた任意の２つの構成要素は、構成又は中間構成要素にかかわらず、所望の機能を達成するように互いに「関連付けられている」と理解し得る。同様に、そのように関連付けた任意の２つの構成要素は、所望の機能を達成するために互いに「動作可能に接続された」又は「動作可能に結合された」とも見なし得る。
【００７８】
更に、例えば、一実施形態では、システム１０の図示した要素は、単一の集積回路上又は同一装置内に配置された回路である。他の選択肢として、システム１０には、相互接続された任意の数の別個の集積回路又は別個の装置を含み得る。更に、例えば、システム１０又はその一部は、物理的な回路又は物理的な回路に転換可能な論理表現のソフト又はコード表現であってよい。このように、システム１０は、任意の適切な種類のハードウェア記述言語で具現化し得る。
【００７９】
更に、当業者は、上述した動作の機能間の境界は、単なる例示であることを認識されるであろう。複数動作の機能は、単一動作に集約してもよく、及び／又は単一動作の機能は、追加の動作に分散してもよい。更に、他の実施形態には、特定の動作の複数のインスタンスを含んでもよく、動作順序は、様々な他の実施形態では、変更してもよい。
【００８０】
本明細書に述べたソフトウェアの全て又は一部は、データ処理システム１０の受信される要素であってよく、例えば、メモリ１８等のコンピュータ判読可能媒体又は他のコンピュータシステム上の他の媒体から受信し得る。そのようなコンピュータ判読可能媒体は、データ処理システム１０等の情報処理システムに、恒久的に、着脱可能に、又は遠隔的に結合してよい。コンピュータ判読可能媒体には、例えば、これらに限定するものではないが、ほんの一例を挙げると、ディスク及びテープ記憶媒体を含む磁気記憶媒体と、コンパクトディスク媒体（例えば、ＣＤ－ＲＯＭ、ＣＤ－Ｒ等）及びデジタル映像ディスク記憶媒体等の光学記憶媒体と、ＦＬＡＳＨメモリ、ＥＥＰＲＯＭ、ＥＰＲＯＭ、ＲＯＭ等の半導体ベースのメモリユニットを含む不揮発性メモリ記憶媒体と、強磁性デジタルメモリと、ＭＲＡＭと、レジスタ、バッファ又はキャッシュ、主記憶装置、ＲＡＭ等を含む揮発性記憶媒体と、コンピュータネットワーク、二地点間電気通信設備、及び搬送波伝送媒体を含むデータ伝送媒体と、の内から任意の数のものを含んでよい。

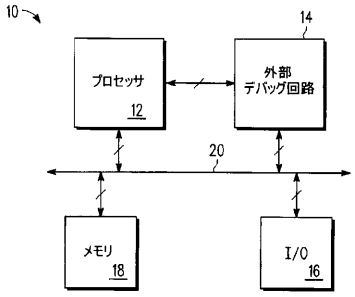
【００８１】
本発明について、本明細書では特定の実施形態を参照して述べたが、様々な修正や変更は、以下の請求項に記載した本発明の範囲から逸脱することなく行い得る。従って、明細書及び図は、限定的であるというよりもむしろ例示であると解釈すべきであり、また、そのような全ての修正は、本発明の範囲内に含まれるものとする。本明細書において具体的な実施形態に関して述べたあらゆる恩恵、利点、又は問題に対する解決策も、全ての請求項の不可欠、必須、又は本質的な特徴もしくは要素であると解釈しようとするものではない。

【００８２】
本明細書に用いた用語「結合された」は、直接結合又は機械的結合に限定するものではない。

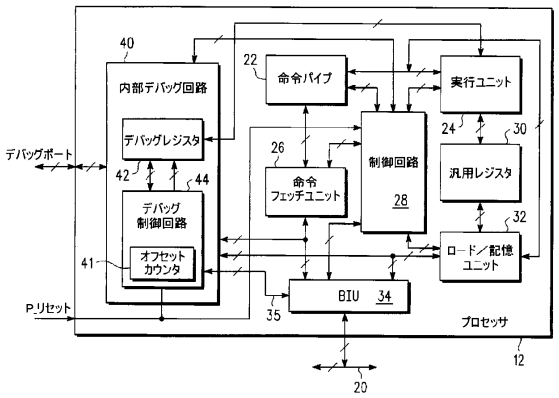
更に、本明細書に用いた用語「ａ又はａｎ（不定冠詞）」は、１つ又は２つ以上と定義する。更に、請求項において「少なくとも１つの」及び「１つ又は複数の」等の導入節を用いることは、不定冠詞「ａ」又は「ａｎ」による他の請求項要素の導入が、同じ請求項に導入節「１つ又は複数の」又は「少なくとも１つの」及び「ａ」又は「ａｎ」等の不定冠詞が含まれる場合でも、そのように導入した請求項要素を含む任意の特定の請求項を、そのような要素を１つだけ含む発明に限定することを意味すると解釈すべきではない。同じことは、定冠詞の使用にも当てはまる。

【００８３】
特に明記しない限り、「第１」及び「第２」等の用語は、そのような用語が述べる要素間を任意に区別するために用いる。従って、これらの用語は、必ずしもそのような要素の時間的な又は他の優先順位付けを示そうとするものではない。

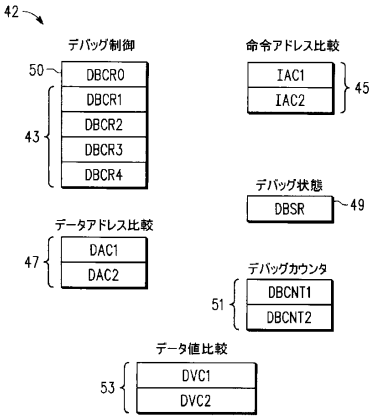
【図１】



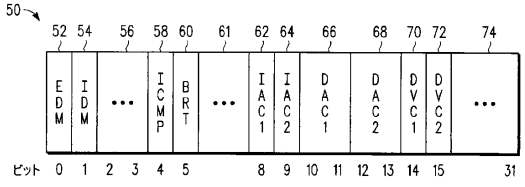
【図２】



【図３】



【図４】



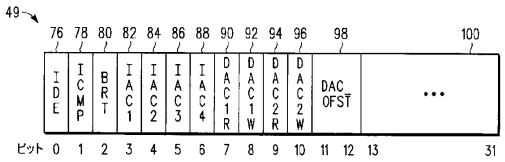
【図５】

ビット	名前	説明	
0	EDM	外部デバッグモード。このビットはソフトウェアには読取のみ可能。 0-：外部デバッグモード無効。内部デバッグイベントは外部デバッグイベントへマッピングされない。 1-：外部デバッグモード有効。イベントはCPUにコードの中断を媒介させない。ソフトウェアにはデバッグレジスタに対する書込は許可されない。{DBCRx, DBSR, DBCNT, IAC1-2, DAC1-2, DVC1-2}.	52
1	IDM	内部デバッグモード。 0-：デバッグ例外は無効。EDMがセットされない限りデバッグイベントはDBSRに影響を与えない。 1-：デバッグ例外は有効。有効化されたデバッグイベントはDBSRを更新する。また、CPUにコードの中断を媒介させることもできる。ソフトウェアにはデバッグレジスタに対する書込が許可される。	54
2:3	-	予約済み	56
4	ICMP	命令完了デバッグイベント有効。 0-：ICMPデバッグイベントは無効。 1-：ICMPデバッグイベントは有効。	58
5	BRT	分岐取得デバッグイベント有効。 0-：BRTデバッグイベントは無効。 1-：BRTデバッグイベントは有効。	60
6:7	-	予約済み	61

【図６】

ビット	名前	説明	
8	IAC1	命令アドレス比較1デバッグイベント可能 0-：IAC1デバッグイベントは無効。 1-：IAC1デバッグイベントは有効。	62
9	IAC2	命令アドレス比較2デバッグイベント可能 0-：IAC2デバッグイベントは無効。 1-：IAC2デバッグイベントは有効。	64
10:11	DAC1	データアドレス比較1デバッグイベント可能 00-：DAC1デバッグイベントは無効。 01-：記憶型データストレージアクセスに対してのみDAC1デバッグイベントは有効。 10-：ロード型データストレージアクセスに対してのみDAC1デバッグイベントは有効。 11-：記憶型またはロード型データストレージアクセスに対してのみDAC1デバッグイベントは有効。	66
12:13	DAC2	データアドレス比較2デバッグイベント可能 00-：DAC2デバッグイベントは無効。 01-：記憶型データストレージアクセスに対してのみDAC2デバッグイベントは有効。 10-：ロード型データストレージアクセスに対してのみDAC2デバッグイベントは有効。 11-：記憶型またはロード型データストレージアクセスに対してのみDAC2デバッグイベントは有効。	68
14	DVC1	データ値比較1修飾子 0-：DAC1デバッグイベントはデータ値比較による影響を受けない 1-：DAC1デバッグイベントはデータ値比較によって修飾される	70
15	DVC2	データ値比較2修飾子 0-：DAC2デバッグイベントはデータ値比較による影響を受けない 1-：DAC2デバッグイベントはデータ値比較によって修飾される	72
16-31	-	予約済み	73

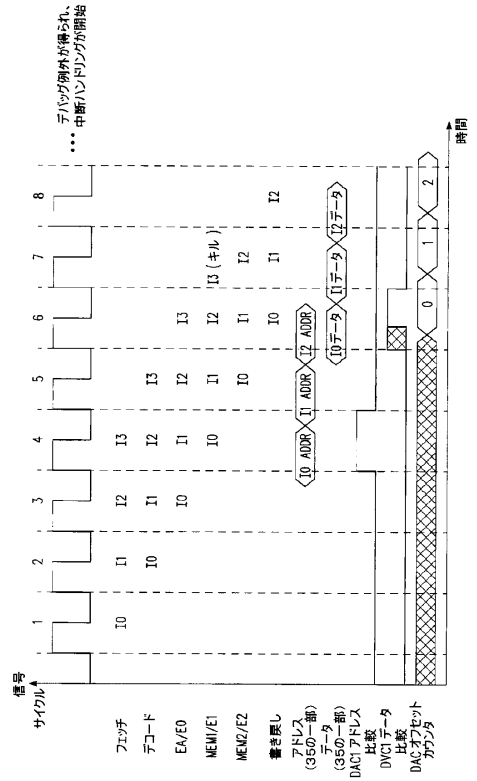
【図７】



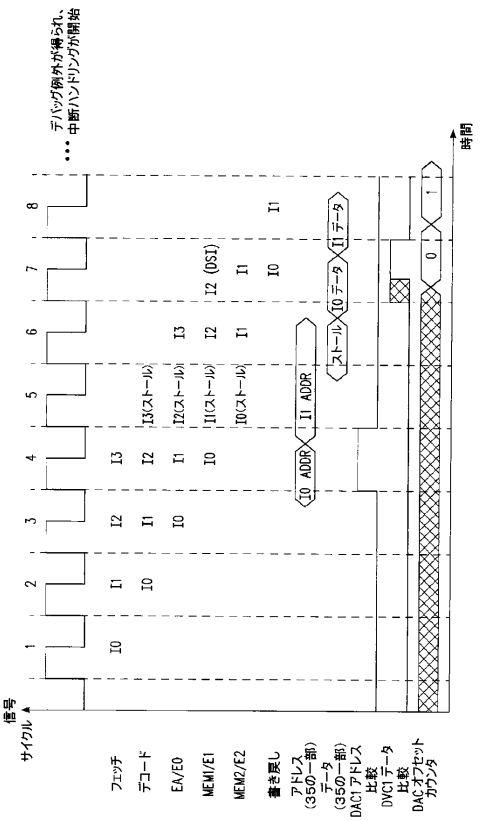
【図８】

ビット	名前	説明	
0	IDE	曖昧デバッグイベント デバッグ例外が無効の場合は1にセットされ、デバッグイベントによってそのデバッグ状態レジスタビットは1にセットされる。	76
1	ICMP	命令完了デバッグイベント 命令完了デバッグイベントが発生した場合は1にセットされる。	78
2	BRT	分岐取得デバッグイベント 分岐取得デバッグが発生した場合は1にセットされる。	80
3	IAC1	命令アドレス比較1デバッグイベント IAC1デバッグイベントが発生した場合は1にセットされる。	82
4	IAC2	命令アドレス比較2デバッグイベント IAC2デバッグイベントが発生した場合は1にセットされる。	84
5	IAC3	命令アドレス比較3デバッグイベント IAC3デバッグイベントが発生した場合は1にセットされる。	86
6	IAC4	命令アドレス比較4デバッグイベント IAC4デバッグイベントが発生した場合は1にセットされる。	88
7	DAC1R	データアドレス比較1読取デバッグイベント DBCR0[DAC1]=0b10 または DBCR0[DAC1]=0b11 において、読取型のDAC1デバッグイベントが発生した場合は1にセットされる。	90
8	DAC1W	データアドレス比較1書込デバッグイベント DBCR0[DAC1]=0b01 または DBCR0[DAC1]=0b11 において、書込型のDAC1デバッグイベントが発生した場合は1にセットされる。	92
9	DAC2R	データアドレス比較2読取デバッグイベント DBCR0[DAC2]=0b10 または DBCR0[DAC2]=0b11 において、読取型のDAC2デバッグイベントが発生した場合は1にセットされる。	94
10	DAC2W	データアドレス比較2書込デバッグイベント DBCR0[DAC2]=0b01 または DBCR0[DAC2]=0b11 において、書込型のDACwデバッグイベントが発生した場合は1にセットされる。	96
11:12	DAC_OFST	データアドレス比較オフセット 同時にDTLBまたはDSIエラーが発生しない限り、DACデバッグ例外を得るロードまたは記憶命令のアドレスからの保存されたDSRRO値のオフセット－1を示し、発生した場合、このフィールドは0b00にセットされ、DBSR[IDE]は1にセットされる。正常な場合、0b00にセットされている。DVC DACでは、このフィールドは、0b01, 0b10, または0b11にセットされる。	98
13-31	-	予約済み	100

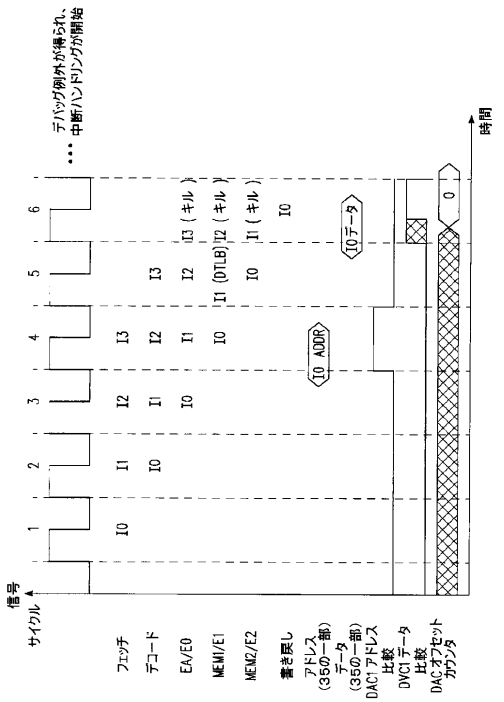
【図９】



【図１０】



【図１１】



【図１２】

	第1ロード／記憶クラス命令 (I0)	第2命令（他に指定のない限り、ロード／記憶クラス） (I1)	第3命令（他に指定のない限り、ロード／記憶クラス） (I2)	結果
201	DTLBエラー、DACなし	–	–	DTLB例外を取り、DBSR更新なし。
202	DSI、DACなし	–	–	DSI例外を取り、DBSR更新なし。
203	DTLBエラー、DACxあり	–	–	デバッグ例外を取り、DBSRはDACxおよびIDEの設定を更新し、DAC_OFSTは0b00にセットされる。DSRROポインタは第1ロード／記憶クラス命令を指す。
204	DSI、DACxあり	–	–	デバッグ例外を取り、DBSRはDACxおよびIDEの設定を更新し、DAC_OFSTは0b00にセットされる。DSRROポインタは第1ロード／記憶クラス命令を指す。
205	DACx	–	–	デバッグ例外を取り、DBSRはDACxの設定を更新し、DAC_OFSTは0b00にセットされる。DSRROポインタは第2ロード／記憶クラス命令を指す。
206	DVCx DACx	例外なし、任意の命令	例外なし、非LD／ST命令	デバッグ例外を取り、DBSRはDACxの設定を更新し、DAC_OFSTは0b01にセットされる。DSRROポインタは第3ロード／記憶クラス命令を指す。
207	DVCx DACx（図9）	例外なし	例外なし、LD／ST命令	デバッグ例外を取り、DBSRはDACxの設定を更新し、DAC_OFSTは0b10にセットされる。DSRROポインタは第3命令の後の命令を指す。
208	DVCx DACx（図11）	DTLBエラー、DACなし	–	デバッグ例外を取り、DBSRはDACxの設定を更新し、DAC_OFSTは0b00にセットされる。DSRROポインタは第2ロード／記憶クラス命令を指す。 注：この場合、第2LD／ST例外はマスクされる。この挙動は実装に応じる。また、他のプロセッサに対して異なってもよい。

【図１３】

	第1ロード／記憶クラス命令 (I0)	第2命令（他に指定のない限り、ロード／記憶クラス） (I1)	第3命令（他に指定のない限り、ロード／記憶クラス） (I2)	結果
209	DVCx DACx	DSI、DACなし	–	デバッグ例外を取り、DBSRはDACxの設定を更新し、DAC_OFSTは0b00にセットされる。DSRROポインタは第2ロード／記憶クラス命令を指す。 注：この場合、第2LD／ST例外はマスクされる。この挙動は実装に応じる。また、他のプロセッサに対して異なってもよい。
210	DVCx DACx	DTLBエラー、DACyあり	–	デバッグ例外を取り、DBSRはDACxの設定を更新し、DAC_OFSTは0b00にセットされる。DSRROポインタは第2ロード／記憶クラス命令を指す。 注：この場合、第2LD／ST例外はマスクされる。この挙動は実装に応じる。また、他のプロセッサに対して異なってもよい。
211	DVCx DACx	DSI、DACyあり	–	デバッグ例外を取り、DBSRはDACxの設定を更新し、DAC_OFSTは0b00にセットされる。DSRROポインタは第2ロード／記憶クラス命令を指す。 注：この場合、第2LD／ST例外はマスクされる。この挙動は実装に応じる。また、他のプロセッサに対して異なってもよい。
212	DVCx DACx	DACy	–	デバッグ例外を取り、DBSRはDACx、DACyの更新し、DAC_OFSTは0b01にセットされる。DSRROポインタは第3命令を指す。 注：この場合、X=Yであるとき、得られるDBSRおよびDSRROの状態は「DACyなし」の場合と区別できないことがある。

【図１４】

	第1ロード／記憶クラス命令 (I0)	第2命令（他に指定のない限り、ロード／記憶クラス） (I1)	第3命令（他に指定のない限り、ロード／記憶クラス） (I2)	結果
213	DVCx DACx	DVCy DACy, 正常なLD／ST	非LD／ST命令	デバッグ例外を取り、DBSRはDACx, DACyの設定を更新し、DAC_OFSTは0b01にセットされる。DSRROポインタは第3命令を指す。 注：この場合、X＝Yであるとき、得られるDBSRおよびDSRROの状態は「DACyなし」の場合と区別できないことがある。
214	DVCx DACx	DVCy DACy, 正常なLD／ST	LD／ST命令、例外なし	デバッグ例外を取り、DBSRはDACx, DACyの設定を更新し、DAC_OFSTは0b10にセットされる。DSRROポインタは第3ロード／記憶クラス命令の後の命令を指す。 注：この場合、X＝Yであるとき、得られるDBSRおよびDSRROの状態は「DACyなし」の場合と区別できないことがある。
215	DVCx DACx	DVCy DACy, 正常なLD／ST	DSI, DAC ありまたはなし	デバッグ例外を取り、DBSRはDACx, DACyの設定を更新し、DAC_OFSTは0b01にセットされる。DSRROポインタは第3命令を指す。 注：この場合、X＝Yであるとき、得られるDBSRおよびDSRROの状態は「DACyなし」の場合と区別できないことがある。 注：この場合、第3LD／ST例外はマスクされる。この挙動は実装に応じる。また、他のプロセッサに対して異なってもよい。

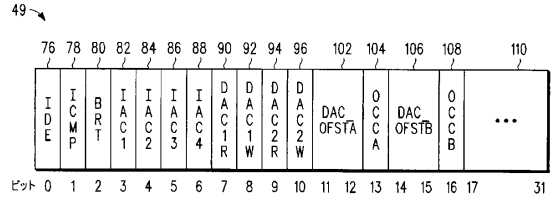
【図１５】

	第1ロード／記憶クラス命令 (I0)	第2命令（他に指定のない限り、ロード／記憶クラス） (I1)	第3命令（他に指定のない限り、ロード／記憶クラス） (I2)	結果
216	DVCx DACx	DVCy DACy, 正常なLD／ST	DTLB, DAC ありまたはなし	デバッグ例外を取り、DBSRはDACx, DACyの設定を更新し、DAC_OFSTは0b01にセットされる。DSRROポインタは第3命令を指す。 注：この場合、X＝Yであるとき、得られるDBSRおよびDSRROの状態は「DACyなし」の場合と区別できないことがある。 注：この場合、第3LD／ST例外はマスクされる。この挙動は実装に応じる。また、他のプロセッサに対して異なってもよい。
217	DVCx DACx	DVCy DACy, 正常なLD／ST	DACy, または DVCy DACy, 正常なLD／ST または多ワード LD／ST	デバッグ例外を取り、DBSRはDACx, DACyの設定を更新し、DAC_OFSTは0b10にセットされる。DSRROポインタは第3ロード／記憶クラス命令の後の命令を指す。 注：この場合、X＝Yであるとき、得られるDBSRおよびDSRROの状態は「DACyなし」の場合と区別できないことがある。
218	DVCx DACx	DVCy DACy, LD／ST, 多（LMW, STMW）	LD／STを含む任意の命令	デバッグ例外を取り、DBSRはDACx, DACyの設定を更新し、DAC_OFSTは0b01にセットされる。DSRROポインタは第3命令を指す。 注：この場合、X＝Yであるとき、得られるDBSRおよびDSRROの状態は「DACyなし」の場合と区別できないことがある。

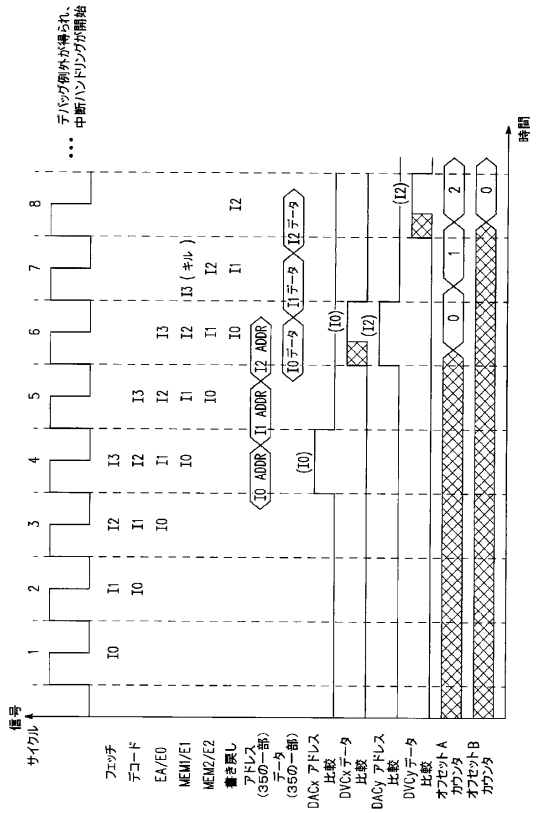
【図１６】

	第1ロード／記憶クラス命令 (I0)	第2命令（他に指定のない限り、ロード／記憶クラス） (I1)	第3命令（他に指定のない限り、ロード／記憶クラス） (I2)	結果
219	DVCx DACx （図10）	任意の命令（例外なし）	DSI, DAC ありまたはなし, 正常なLD／ST または多ワード LD／ST	デバッグ例外を取り、DBSRはDACx, DACyの設定を更新し、DAC_OFSTは0b01にセットされる。DSRROポインタは第3命令を指す。 注：この場合、第3LD／ST例外はマスクされる。この挙動は実装に応じる。また、他のプロセッサに対して異なってもよい。
220	DVCx DACx	任意の命令（例外なし）	DACy, または DVCy DACy, 正常なLD／ST または多ワード LD／ST	デバッグ例外を取り、DBSRはDACx, DACyの設定を更新し、DAC_OFSTは0b10にセットされる。DSRROポインタは第3クラス命令の後の命令を指す。 注：この場合、X＝Yであるとき、得られるDBSRおよびDSRROの状態は「DACyなし」の場合と区別できないことがある。

【図１７】



【図１８】



フロントページの続き

(56)参考文献 特開平０８－１８５３３６（ＪＰ，Ａ）
特開平０６－１３９１０７（ＪＰ，Ａ）
特開平１０－２２２３９３（ＪＰ，Ａ）
特開平１０－３３３９３９（ＪＰ，Ａ）
特開昭６２－２３９２３８（ＪＰ，Ａ）
特開２００２－１６３１２６（ＪＰ，Ａ）
米国特許第０６５９１３７８（ＵＳ，Ｂ１）
米国特許第０５２５７３５８（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １１／２８－１１／３４
Ｇ０６Ｆ ９／３８