(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4648622号
(P4648622)
(45)発行日　平成23年3月9日(2011.3.9)　(24)登録日　平成22年12月17日(2010.12.17)

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

請求項の数 2 (全 17 頁)

(21)出願番号　特願2003-392669(P2003-392669)
(22)出願日　平成15年11月21日(2003.11.21)
(65)公開番号　特開2004-178591(P2004-178591A)
(43)公開日　平成16年6月24日(2004.6.24)
審査請求日　平成18年11月21日(2006.11.21)
(31)優先権主張番号　301969
(32)優先日　平成14年11月22日(2002.11.22)
(33)優先権主張国　米国(US)
(31)優先権主張番号　302189
(32)優先日　平成14年11月22日(2002.11.22)
(33)優先権主張国　米国(US)

(73)特許権者　501229528
テキサス　インスツルメンツ　インコーポレイテッド
アメリカ合衆国、テキサス、ダラス、チャーチル　ウエイ　７８３９
(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　鉐三
(74)代理人　100118821
弁理士　祖父江　栄一

最終頁に続く

(54)【発明の名称】ダイナミック・トレース用プログレッシブ拡張圧縮マスク

(57)【特許請求の範囲】
【請求項１】
トレース・アドレスの圧縮の方法であって、
以前のトレース・アドレスを格納するステップと、
現トレース・アドレスのそれぞれ複数の等しい長さの区間を、前記格納された以前のトレース・アドレスの対応する複数の等しい長さの区間と比較するステップと、
前記格納された以前のトレース・アドレスのそれぞれの区間と一致しない、前記現トレース・アドレスの下位区間、または前記格納された以前のトレース・アドレスと一致しない前記現トレース・アドレスの、ある区間よりも下位にある区間、のみを送信することによって、前記現トレース・アドレスの指示を送信するステップと、
前記現トレース・アドレスのどの区間も、前記格納された以前のトレース・アドレスの対応する区間に一致しない場合、前記現トレース・アドレスを以前のトレース・アドレスとして格納するステップと、
前記現トレース・アドレスの少なくとも１つの区間が、前記格納された以前のトレース・アドレスの対応する区間に一致する場合、前記格納された以前のトレース・アドレスを再利用するステップと、
を含む、方法。
【請求項２】
トレース圧縮装置であって、
現トレース・アドレスを受け取る入力と、

以前のトレース・アドレスを格納する比較アドレス・レジスタと、
複数の区間比較器であって、各区間比較器が前記比較アドレス・レジスタに接続されて前記以前のトレース・アドレスの１区間を受け取り、更に前記入力に接続されて前記現トレース・アドレスの対応する区間を受け取り、各区間比較器が、前記間隔的に対応する区間の一致または不一致を示す、前記複数の区間比較器と、
前記入力と前記区間比較器の各々とに接続されたトレース・ストリーム制御部と、
を含み、
前記トレース・ストリーム制御部が、
前記格納した以前のトレース・アドレスのそれぞれの区間と一致しない、前記現トレース・アドレスの下位区間、または前記格納した前記以前のトレース・アドレスと一致しない前記現トレース・アドレスの、ある区間よりも下位にある区間、のみを送信し、
前記現トレース・アドレスのどの区間も、前記格納された以前のトレース・アドレスの対応する区間に一致しない場合、前記現トレース・アドレスを以前のトレース・アドレスとして格納するよう、前記比較アドレス・レジスタを制御し、
前記現トレース・アドレスの少なくとも１つの区間が、前記格納された以前のトレース・アドレスの対応する区間に一致する場合、前記格納された以前のトレース・アドレスを再利用するよう、前記比較アドレス・レジスタを制御する、
前記トレース圧縮装置。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、特に高度集積ディジタル信号処理システムのためのエミュレーション・ハードウエアに関し、特にトレース・アドレスの比較方法およびトレース圧縮装置に関する。

【背景技術】

【０００２】

高度なウエハ・リソグラフィおよび表面実装パッケージング技術は、電子的設計のシリコン・レベルおよびプリント回路基板レベル双方において、増々複雑な機能を統合しつつある。しかしながら、設計密度が高まり、相互接続ピッチが狭くなったことの結果、検査やエミュレーションのために回路に物理的にアクセスし難くなった。完成した製品を制御および観察しつつ検査およびデバッグすることができるようにするためには、設計中にも検査可能である(designed-in testability)必要がある。あらゆる製造上の欠陥は、製品を出荷する前の最終検査の間に検出できることが好ましい。この基本的な要望に応ずることは、論理設計フェーズにおいて検査を可能にし、自動検査機器が製品を検査できるように考慮しなければ、複雑な設計では達成が困難である。

【０００３】

機能性および製造上の欠陥についての検査に加えて、アプリケーション・ソフトウエアの開発では、システムまたはサブシステムの設計フェーズにおいても同様のレベルのシミュレーション、観察可能性および制御可能性が必要となる。設計のエミュレーション・フェーズは、最終機器またはアプリケーションがシステム・ソフトウエアとリンクされたときに、１つ以上のＩＣ（集積回路）のシステムがこれらの中で正しく機能することを保証しなければならない。自動車産業、電気通信、防衛システム、生活支援システムにおいてＩＣの使用が増大するに連れて、総合的な検査や広範なリアル・タイム・デバッグは、欠くことができない必要事項となっている。

【０００４】

機能的検査では、設計者が検査ベクトルを発生して仕様に一致することを確認するが、この検査方法は今でも広く用いられ続けている。超大型システムでは、この方法は、高いレベルでの障害検出に用いるには不適当であることがわかっている。完全な検査可能性、制御可能性および観察可能性を得るには、自動的に発生する検査パターンが望ましい。これらは、システム・レベルからトランジスタ・レベルまでの検査階層全体に及ぶ主要な目標である。

【０００５】
大規模な設計における別の問題は、検査のために長時間およびかなりの出費が設計にかかることである。再利用可能設計(design-for-reusability)の概念と調和する検査性回路、システムおよび方法を有することができれば望ましいであろう。このようにすれば、初期デバイスに実装される検査性、シミュレーションおよびエミュレーション回路、システムならびに方法を再利用することによって、その後のデバイスおよびシステムの検査性、シミュレーションおよびエミュレーションのための限界設計コストを低く抑えることができる。検査性、シミュレーションおよびエミュレーションに先取的な計画性がなければ、後に検査パターンの作成およびアップグレードを行う際に、設計時間の著しい延長を招くことになる。

【０００６】
再利用可能なモジュールを設計し、その検査パターンを完全に作成し評価するために多大な投資を行っても、後にモジュールを使用する際に、これを特定用途ロジック内に埋め込んでしまうことがある。こうなると、モジュールにアクセスするのは困難または不可能となる。したがって、この落とし穴を回避することが望ましい。

【０００７】
ＩＣ設計の進展に伴って、内部可視性および制御性の低下、障害対応可能範囲(fault coverage)の減少、状態を変化させる機能の減少、検査の展開および検証に関する問題の増大、設計シミュレーションの複雑化、ＣＡＤ（コンピュータ支援設計）ツールの絶え間ないコスト上昇を招いている。基板の設計では、レジスタの可視性および制御性が低下し、設計検証におけるデバッグおよびシミュレーションが複雑化し、多くの回路を１つのパッケージに封入したために物理的なアクセスが不可能になったことによって従来のエミュレーションができなくなり、基板上の配線の複雑さが高まり、設計ツール、異種素子同士の封入(mixed-mode packaging)、および生産性のための設計コストが上昇するという副作用がある。アプリケーションの設計では、状態の可視性が低下し、高速エミュレーションが難しくなり、時間シミュレーションの調整(scaled time simulation)が必要となり、デバッグ処理の複雑さが高まり、エミュレータのコストが上昇するといった副作用があげられる。生産における副作用には、可視性および制御性の低下、検査ベクトルおよびモデルの複雑化、検査の複雑化、異種素子同士の封入増加、自動検査機器のコスト上昇し、許容度の厳格化等があげられる。

【０００８】
走査に基づくエミュレーションおよびマルチプロセッシング・デバッグを利用したエミュレーション技術が導入されたのは、１０年以上も前のことである。１９８８年、設計サイクル時間に対する圧力、およびオンチップ・エミュレーションに新たに利用可能な空間が動機となって、従来の回路内エミュレーションから走査に基づくエミュレーションに推移した。設計サイクル時間に対する圧力は、３つの要因によって生じた。オンチップ・メモリの使用増大等による集積度の上昇によって、より多くの設計時間が必要となった。クロック・レートの上昇とは、エミュレーション支援ロジックのために電気的な進入が増大したことを意味する。封入処理の一層の精巧化のために、エミュレータの接続性の問題が生じた。今日、これらと同じ要因が、新たな展開を伴って、今日の複雑で、クロック・レートが一層高く、集積度が高い設計に必要なシステム・デバッグ機構を、走査に基づくエミュレータが達成できるための課題となっている。結果的に、システムは、小型化、高速化、および低コスト化する。これらの性能は向上し、フットプリントは増々高密度化している。これら実践的なシステムの傾向の各々が、システム・アクティビティ(system activity)の観察、即ち、迅速なシステム開発を可能にするための秘訣に悪影響を及ぼす。この影響を「可視性の消失(vanishing visibility)」と呼ぶ。

【０００９】
図１は、経時的な可視性および制御、ならびにシステムの高集積化の傾向を示す。アプリケーション開発者は、図１に示す最適可視性レベルを好む。この最適可視性レベルでは、関係するシステム・アクティビティ全ての可視性および制御が得られる。集積度の着実

な進歩およびクロック・レートの上昇のために、時の経過と共に実際の可視性および制御の可用性が徐々に低下しつつある。これらの威力によって、可視性および制御のギャップ、最適な可視性および制御レベルと実際に使用可能なレベルとの間の差が生じる。時が経つに連れて、このギャップは広がっていく。アプリケーション開発メーカはギャップ生長レートを最小にすべく努力している。開発ツール・ソフトウエアおよび共に用いるハードウエア構成要素は、より少ないリソースで、しかも多種多様な方法で、より多くのことを行わなければならない。この使いやすさという課題に対する取り組みは、これらの影響力によって一層強化されている。

【００１０】

今日の高集積度システム・オン・チップ（ＳＯＣ：System-On-a-Chip）技術では、可視性および制御のギャップは、時間の経過と共に劇的に拡大しつつある。ロジック・アナライザや組み込み若しくは区分(partitioned)プロトタイプ・システムのような従来からのデバッグ選択肢は、今日のシステムの集積レベルや、増々高まりつつあるクロック・レートに歩調を合わせることはできない。集積度の上昇に伴って、多数のサブシステム構成要素を接続するシステム・バスがチップ上に移動するため、従来のロジック・アナライザはこれらのバスにアクセスできなくなる。バスの可視性の制約または事実上の喪失により、ロジック・アナライザのようなツールを用いても、開発中のシステムを制御するために必要なシステムのアクティビティを視認したり、トリガ機構を設けることができなくなった。この可視性の喪失に伴って、制御が不能となる。何故なら、アクセスできないものを制御することは困難であるからである。

【００１１】

この傾向に対処するために、システム設計者はこれらのバスを表出させるように努めた。このため、システム構成要素は、表出したバスを用いて試作システムの構成が可能となるように組み立てられた。この手法も、システム・クロック・レートの絶え間ない上昇歩調によって阻害される。中央演算装置（ＣＰＵ）のクロック・レートが上昇すると、チップ間のインターフェースの速度はそれについて行けなくなる。チップ間通信レートの遅れを補償するためにインターフェース待機状態が追加されることにより、区分システムの性能は、その集積されている相手と歩調を合わすことができないことを、開発者は突き止めた。ある時点では、この性能低下は許容できないレベルにまで達し、区分プロトタイプ・システムはもはや実用的なデバッグ選択肢ではなくなった。現在では、生産機器は、アプリケーションの開発のためのプラットフォームとして機能しなければならない。

【００１２】

また、ＣＰＵクロック・レートの上昇により、他の単純な可視性機構の可用性も制限される。ＣＰＵクロック・レートは最大Ｉ／Ｏ状態レートを上回ることができるので、ネーティブな形式で(native form)で情報をエクスポートする可視性ポートはもはやＣＰＵについて行くことはできない。また、オンチップ・サブシステムも、ＣＰＵクロック・レートよりは遅いクロック・レートで動作する。この手法は、システムの設計を簡略化し、電力消費を削減するためには用いることができる。これらの開発は、単純な可視性ポートは、もはや、ＣＰＵアクティビティの正しい状態若しくは明確な表示(clear view)を送り出すためには、あてにできないことを意味する。可視性および制御性が低下するに連れて、アプリケーションを開発するために用いられる開発ツールの生産性が低下する。また、これらのツールは、可視性および制御性を維持するために必要なツールの複雑化のために、増々使用が困難になると思われる。システム・オン・チップによって生じた可視性、制御、および使いやすさの問題のために、製品開発サイクルが長引くことが多くなっている。

【００１３】

集積化の傾向が開発者には厳しいデバッグ環境を強いるとしても、これらは、問題をデバッグする新たな手法が出現するという希望も与える。高密度化やクロック・レートの上昇は、開発サイクル時間に対する重圧となるが、これらを解決する機会も生み出す。オンチップ・デバッグ機能は、これまで以上に手頃な価格となった。超大型メモリ構造が増々高速高性能チップ上で拡大するに連れて、ＣＰＵおよびメモリ・サブシステムに伴うラン

ダム・ロジックに関係するシステム・コストは、システム・コスト全体の割合としては低下しつつある。数千個のゲートによるコスト上昇分は、これまでになく低くなっている。このサイズの回路は、場合によっては、今日のチップ設計の曲がり角に追いやられる場合もある。今日の高密度パッケージにおけるピン当たりのコスト上昇分も低下している。このため、より多くのピンをデバッグのために割り当てることが容易となった。手頃な価格のゲートおよびピンを共に用いることによって、システム・オン・チップによってもたらされた課題に取り組むために必要な、新たなオンチップ・エミュレーション機構（ファシリティ）の配備が可能となる。

【００１４】

また、生産機器がアプリケーション・デバッグ・プラットフォームとしても機能する場合、これらは市場の目標に合わせて時間を捻出するのに十分なデバッグ機能（ケイパビリティ）を備えていなければならない。デバッグの必要性は個々のアプリケーションで異なるので、オンチップ・デバッグ機構を調節し、市場およびコストの要望に対して時間のバランスが取れるようにすることが望ましい。これらオンチップ機能はチップの循環コストに影響を及ぼすので、いずれの解決策にとっても規模変更可能性若しくはスケーラビリティ(scalability)が最も重要である。「必要なものにだけ支払う」ことは、オンチップ・ツールの配備にとっても処理原則であるはずである。この新たな枠組みにおいて、システム構築者は、チップ・コストの制約や製品開発チームのデバッグについての要望のバランスを取りながら、オンチップ・デバッグ機構を、それ以外の機能性と共に指定することができる。

【００１５】

図２は、４つのエミュレータ構成要素を含むエミュレータ・システム１００を示す。これら４つの構成要素は、デバッガ・アプリケーション・プログラム１１０、ホスト・コンピュータ１２０、エミュレーション・コントローラ１３０、およびオンチップ・デバッグ機構（ファシリティ）１４０である。図２は、これらの構成要素の接続を示す。ホスト・コンピュータ１２０は、ホスト１２０外部のエミュレーション・コントローラ１３０に接続されている。また、エミュレーション・コントローラ１３０はターゲット・システム１４０にも接続されている。ユーザは、デバッガ・アプリケーション・プログラム１１０を通じて、ターゲット・システム１４０上でターゲット・アプリケーションを制御することが好ましい。

【００１６】

ホスト・コンピュータ１２０は通常パーソナル・コンピュータである。ホスト・コンピュータ１２０は、エミュレータ・コントローラ１３０を通じてデバッグ機能にアクセスする。デバッガ・アプリケーション・プログラム１１０は、ホスト・コンピュータ１２０を通じて、ユーザに易しい形態でデバッグ機能を提供する。デバッグ・リソースは、必要に応じて、デバッグ・アプリケーション・プログラム１１０によって割り当てられ、このためのユーザの負担を軽減する。ソース・レベルのデバッグは、デバッグ・リソースを利用し、それらの複雑性をユーザには隠している。デバッガ・アプリケーション・プログラム１１０は、オンチップ・トレースおよびトリガ機構と共に、対象のチップ・アクティビティを選択し、記録し、表示する手段を設ける。トレース表示は、トレース・ログを生成したソース・コードと自動的に相関付けられる。エミュレータは、デバッグ制御およびトレース記録機能双方を備えている。

【００１７】

デバッグ機構は、ＪＴＡＧまたは同様のシリアル・デバッグ・インターフェースを介した標準的なエミュレータ・デバッグ・アクセスを用いてプログラムすることが好ましい。ピンは貴重なので、本発明の好適な実施形態では、デバッグ・ピンの集合を、トレース、トリガ、およびその他のデバッグ機能で共有し、シリコン・コストにおける増加分を少なく済ませるようにした。固定したピン・フォーマットにも対応可能である。ピン共有選択肢を採用する(deploy)場合、デバッグ・ピンの利用は、各デバッグ・セッションの開始時に決定され、その後にターゲット・システム１４０にアプリケーション・プログラムを実

行するように指令する。これによって、トレース・エクスポート帯域幅(trace export bandwidth)を最大化する。トレース帯域幅を最大化するには、最大数のピンをトレースに割り当てる。

【００１８】

システム内部のデバッグ機能および構築（ビルディング）ブロックは、可変とすることもできる。したがって、デバッガ・アプリケーション・プログラム１００は、実行時（ランタイム）にコンフィギュレーションを確定する。この手法では、ハードウエア・ブロックが、コンフィギュレーションおよびレジスタ編成に関する一連の制約を満たす必要がある。他の構成要素は、システム・メモリ・マップにおいてブロックやその他のペリフェラル(peripheral)を突き止め位置づけるように設計されたハードウエア検索機能を設ける。デバッガ・アプリケーション・プログラム１１０は、検索機構を用いてリソースを突き止める。モジュールが位置するアドレスおよびタイプＩＤが、発見された各ブロックを一意に特定する。一旦ＩＤが発見されたなら、設計データベースを用いれば、正確なコンフィギュレーションならびに全てのシステム入力および出力を確認することができる。

【００１９】

ホスト・コンピュータ１２０は、一般に、少なくとも６４Ｍバイトのメモリを含み、Ｗｉｎｄｏｗｓ（登録商標）９５、ＳＲ－２、Ｗｉｎｄｏｗｓ（登録商標）　ＮＴ、またはそれ以降のバージョンのＷｉｎｄｏｗｓ（登録商標）を走らせることができる。ホスト・コンピュータ１２０は、エミュレータが必要とする通信インターフェースの１つに対応していなければならない。これらは、イーサネット（登録商標）１０Ｔおよび１００Ｔ、ＴＣＰ／ＩＰプロトコル、ユニバーサル・シリアル・バス（ＵＳＢ）、ファイアワイヤＩＥＥＥ１３９４、ならびにＳＰＰ、ＥＰＰおよびＥＣＰのようなパラレル・ポートを含むことができる。

【００２０】

ホスト・コンピュータ１２０は、リアル・タイムのデータ交換帯域幅を決定するにあたって主要な役割を果たす。まず、ホスト－エミュレータ間通信は、最大維持リアル・タイム・データ交換帯域幅を規定する際に主要な役割を果たす。何故なら、エミュレータ・コントローラ１３０はその受信リアル・タイム・データ交換バッファが満杯になったら直ちにこれらを空にしなければならないからである。第２に、リアル・タイム・データの発信または受信を行うホスト・コンピュータ１２０は、受信したリアル・タイム・データ交換データの準備および送信または処理および格納を維持するための十分な処理能力またはディスク帯域幅を有していなければならない。技術的現状では、ファイアワイヤ(Fire wire)通信チャネル（ＩＥＥＥ１３９４）を備えたパーソナル・コンピュータが、最大のリアル・タイム・データ交換帯域幅を得るには好ましい。この帯域幅は、他の通信選択肢よりも性能を１０倍高めることができる。

【００２１】

エミュレーション・コントローラ１３０は、ホスト・コンピュータ１２０とターゲット・システム１４０との間にブリッジを設ける。エミュレーション・コントローラ１３０は、ホスト・コンピュータ１２０上で実行するデバッガ・アプリケーション・プログラム１１０とターゲット・システム１４０上で実行するターゲット・アプリケーションとの間で受け渡されるあらゆるデバッグ情報を処理する。現時点において好適な最小のエミュレータ・コンフィギュレーションは、リアル・タイム・エミュレーション、リアル・タイム・データ交換、トレース、および高度分析といった機能の全てに対応する。

【００２２】

エミュレーション・コントローラ１３０は、好ましくは、実行制御、メモリ、レジスタ・アクセスのようなリアル・タイム・エミュレーション機能に、３、４、または５ビットの走査型インターフェース(scan based interface)を通じてアクセスする。リアル・タイム・データ交換機能は、走査によって、または走査以外の直接的なターゲット－エミュレータ間接続を用いる３つの上位帯域幅リアル・タイム・データ交換フォーマットを用いることによってアクセスすることができる。入力および出力トリガによって、他のシステム

構成要素がチップにデバッグ・イベントを知らせることや、その逆が可能となる。ビットＩ／Ｏによって、エミュレータはシステム入力および出力を刺激すること、または監視することが可能となる。ビットＩ／Ｏは、工場検査およびその他の時間的に厳しくない(non-time-critical)低帯域幅のエミュレータ／ターゲット動作に対応するために用いることができる。拡張動作モードは、デバイス検査およびエミュレーション動作モードを指定するために用いられる。エミュレータ・コントローラ１３０は、通信部およびエミュレーション部に区分される。通信部は、ホスト通信リンクを支援し、一方エミュレーション部はターゲットにインターフェースして、ターゲット・デバッグ機能およびデバイス・デバッグ・ポートを管理する。エミュレーション・コントローラ１３０は、先にこの中で概説した業界標準の通信リンクの１つを用いて、ホスト・コンピュータ１２０と交信する。ホスト－エミュレータ間接続は、汎用の配線技術を用いて行われる。ホスト－エミュレータ間の分離は、用いるインターフェースに適用される規格によって管理される。

【００２３】

エミュレーション・コントローラ１３０は、１本または複数のターゲット・ケーブルを通じてターゲット・システム１４０と交信する。デバッグ、トレース、トリガ、およびリアル・タイム・データ交換機能は、ターゲット・ケーブルを共有し、場合によっては、同じデバイス・ピンを共有する。ターゲット・システム１４０が、１本のケーブルには収容しきれないトレース幅を配備する場合、１本よりも多いターゲット・ケーブルが必要となる場合もある。トレース、リアル・タイム・データ交換、およびデバッグ通信は全て、このリンクを通じて行われる。エミュレータ・コントローラ１３０は、少なくとも２フィート（約６０ｃｍ）のターゲット－エミュレータ間分離を設けることが好ましい。このエミュレーション技術は、５０ＭＨＺまでの検査クロック・レート、および２００から３００ＭＨＺまたはこれ以上のトレース・クロック・レートが可能である。エミュレータの設計がターゲット・システム１４０の制約を緩和する技法を用いても、これらのレートでエミュレータ・コントローラ１３０およびターゲット・システム１４０間で通知を行うには、設計に努力が必要である。このエミュレーション技術は、チップのデバッグ・ピンの配置、基板のレイアウトに制約を賦課する場合があり、正確なピンのタイミングを必要とする。タイミングの制約を満たすのを補助するために、オンチップ・ピン・マクロを設ける。

【００２４】

オンチップ・デバッグ機能は、開発者に、豊富な開発機能を２段階の拡張可能な手法で提供する。第１段階では、ＣＰＵのメガ・モジュールに組み込んだリアル・タイム・エミュレーション機能を利用した機能性を備える。このリアル・タイム・エミュレーション機能は、一定の機能性を有し、永続的にＣＰＵの一部であるが、殆どの場合、コアの外側に、高性能リアル・タイム・データ交換、高度分析およびトレース機能が追加される。これらの機能を個々に選択し、チップに加入する。エミュレーション・ペリフェラルをシステム設計に加入することにより、第２段階の機能性を形成する。コスト効率的なエミュレーション・ペリフェラルのライブラリは、システムを作成するための構築ブロックを含み、高度分析、高性能リアル・タイム・データ交換、およびトレース機能の構造化を可能にする。好適な実施形態では、５つの標準デバッグ・コンフィギュレーションが提供されるが、カスタム・コンフィギュレーションにも対応する。具体的なコンフィギュレーションについては、この中で後に扱う。

【発明の開示】

【発明が解決しようとする課題】

【００２５】

本発明が解決する問題は、アドレス基準のトレースに関する。マイクロプロセッサおよびディジタル信号プロセッサが有するアーキテクチャは、通例では、高いクロック・レートで実行し、大量のトレース情報を生成する。かかるトレース情報を、分析精度に悪影響を及ぼすことなく、トレース取得およびエンコード・ハードウエアによって処理することは難しい。チップ境界における外部専用トレース・エクスポート・ピンに対する帯域幅の要求が大きいために、トレース・ハードウエアは、中央演算装置を減速(stall)させて、

トレース情報全ての収集を確保せざるを得ない場合もある。このため、同じトレース・ハードウエアを用いた中央演算処理の性能分析（プロファイリング）が侵入性となり、したがって非常に精度が低下する。

【００２６】

これら基準アドレスの収集に対処するには、柔軟性のある圧縮方式が必要となる。本発明は、アプリケーションが用いる基底アドレスに漸進（プログレッシブ）圧縮方式を設けると共に、このアドレスからオフセットのみをエクスポートする方法を設ける。この方式は、基準アドレスの時間的および空間的所在(locality)を利用して、アドレスの範囲が共通基底（ベース）アドレスに該当する状況において、ピンを通じてエクスポートするデータ量を削減する。

【課題を解決するための手段】

【００２７】

トレース・ストリームは、ネーティブ・プログラム・カウンタまたはデータ基準アドレスを、同期マーカの一部として、あるいはレジスタ分岐または例外不連続をエンコードするときに収集する。個々のシステム・コンフィギュレーションは、異なる物理位置毎にコードをマップすることができるが、所与のシステム／アプリケーション・コンフィギュレーション内部におけるコードまたはデータの所在はかなり固定的であることが多い。

【００２８】

ダイナミック・トレース・ストリームの帯域幅は、ピンの割当によって制限され、レジスタ分岐、例外、同期マーカ（これらは全てネーティブ・プログラム・カウンタ・タグを用いる）の存在によって大きく制限される。データ記録(data logging)またはアドレス所在分析(address locality profiling)の一部としてデータ・アドレスを収集するダイナミック・トレース・ストリームは、システム性能に不利になるような侵入を犯すことなく、これらのアドレスを頻繁にエクスポートしなければならない。本発明は、以前のアドレスまたは中央演算装置が指定した比較アドレスとのバイト毎の比較により、エクスポートされるプログラム・カウンタまたはデータ基準アドレスのデータ圧縮を可能にする。異なるバイトのみをトレース・ストリームに含ませる。

本発明のこれらおよびその他の態様を図面に示す。

【発明を実施するための最良の形態】

【００２９】

本発明は、ダイナミック・トレースのための漸増圧縮マーク(progressive extended compression mask)である。本発明は、レジスタ分岐のエンコード、データ・アドレス基準、例外、または同期マーカのエクスポートに対処する際の帯域幅要件を減少させる。３２ビットのプログラム・カウンタまたはデータ・アドレスに対処するための、かかるマークの実施態様の一例では、最後の有効プログラム・カウンタ・アドレスを保持する２４ビット・レジスタを含む。２４ビット・マスクを用いて、トレース取得ハードウエア内部のパイプライン・フラットナ(pipeline flattener)の出力から出現する際のネーティブ・プログラム・カウンタまたはデータ・アドレスの上位範囲を比較する。マスクは、バイト毎に比較され、トレース・ストリームにエクスポートする必要がある基準アドレスの量を判断する。

【００３０】

図３は、ターゲット・システム１４０を用いたオンチップ・デバッグ・アーキテクチャの一例を示す。このアーキテクチャは、数個のモジュール・クラスを用いてデバッグ機能を構成する。これらのクラスの１つに、バス・イベント検出器２１０、補助イベント検出器２１１、およびカウンタ／ステートマシン２１３を含むイベント検出器がある。第２のクラスのモジュールは、トリガ・ビルダ(trigger builder)２２０を含むトリガ発生器である。第３のクラスのモジュールは、トレース収集２３０およびフォーマット化を含むデータ取得である。第４のクラスのモジュールは、トレース・エクスポート２４０、リアル・タイム・データ交換エクスポート２４１を含むデータ・エクスポートである。トレース・エクスポート２４０は、局部発振器２４５からのクロック信号によって制御される。局

部発振器２４５については、後に詳しく説明する。最後のクラスのモジュールは、走査入力／出力をＣＰＵコア２０１にインターフェースする走査アダプタ２５０である。最終的なデータのフォーマット化およびピンの選択は、ピン・マネージャおよびピン・マクロ２６０において行われる。

【００３１】

システム・オン・チップ(system-on-chip)の個々の実施形態のいずれに対しても、デバッグ機能のサイズおよびそれに付随する処理能力は、完全な機能を削除するか、または配備するイベント検出器およびトリガ・ビルダの数を制限することによって調節することができる。加えて、トレース機能は、プログラム・カウンタ・トレースのみからプログラム・カウンタおよびデータならびにＡＳＩＣおよびＣＰＵ発生データのトレースまで徐々に増大させることができる。また、リアル・タイム・データ交換機能を任意に配備することもできる。オンチップ・ツールをカスタム化できるので、アプリケーション開発の枠組みを変化させることができる。過去においては、所与のＣＰＵコアを用いたチップ設計は全て、固定した１組のデバッグ機能に限定されていた。現在では、チップの設計毎に最適化したデバッグ機能が得られる。この枠組みの変化によって、システム・アーキテクトには、製品開発リスクを管理するために必要なツールが手頃なコストで与えられることになった。尚、同じＣＰＵコアであっても、異なるピン出力を有する異なるペリフェラルと共に用いることによって、異なるシステム・オン・チップ製品を具体化できることを注記しておく。これらの異なる実施形態は、異なるデバッグおよびエミュレーション・リソースを必要とする場合もあり得る。本発明のモジュール性により、かかる実施形態の各々は、個々のシステム・オン・チップに応用するために必要なデバッグおよびエミュレーション・リソースのみを含むだけで済む。

【００３２】

リアル・タイム・エミュレーション・デバッグの基礎的な構成要素は、アプリケーションの開発に関係する基本的なデバッグおよび計測管理(instrumentation)に取り組むために用いられる。これは、あらゆる実行制御およびレジスタ可視性機能、ならびにブレークポイントやウオッチポイント機能のような最小のリアル・タイム・データ交換および分析機能を内蔵している。これらのデバッグ動作は、オンチップ・ハードウエア機構を用いて、アプリケーションの実行を制御し、レジスタやメモリへのアクセスを得る。リアル・タイム・エミュレーションが対応可能なデバッグ動作の一部は、ソフトウエア・ブレークポイントの設定、そのポイントにおける機械状態の観察、命令毎に正確な決定を観察するための１ステップ・コード前進、既知のメモリ位置への疑似書き込みの検出、およびメモリおよび周辺レジスタの視認および変更である。

【００３３】

リアル・タイム・エミュレーション機構は、ＣＰＵメガ・モジュール(CPU mega-module)内に組み込まれ、ＣＯＵコア２０１のファブリック(fabric)に編入される(weave)。これによって、ＣＰＵコア２０１を用いた設計が、デバッガ・アプリケーション・プログラム１１０のベースライン・デバッグ、測定管理、およびデータ転送機能に対応するだけの十分なデバッグ機構を有することに確証を与える。各ＣＰＵコア２０１は、基礎的な１組のエミュレーション機能を組み込んでいる。これらの機能には、実行(run)、１命令ステップ、休止および自由実行(halt and free run)のような実行制御、レジスタやメモリの表示および修正、ソフトウエアおよび最小限のハードウエア・プログラム・ブレークポイントを含むブレークポイント、ならびに最小限のハードウエア・データ・ブレークポイントを含むウオッチポイントを含むが、これらに限定されるのではない。

【００３４】

本発明は、データおよびプログラム・カウンタ・アドレスのためのデータ圧縮に関する。好適な実施形態では、本発明は、トレース収集２３０に組み込まれる。本発明は、現アドレスを以前のアドレスと比較し、異なるアドレス・バイトだけを送信する。

【００３５】

図４は、このプロセスをフロー・チャートの形態で示す。プロセス４００は、開始ブロ

ック４０１にて開始する。まず、プロセス４００は、プログラム・カウンタまたはデータ・アドレスのバイト３を２４ビット・マスクと比較する（処理ブロック４０２）。これらのバイトが等しくない場合（判断ブロック４０３においてＮｏ）、プロセス４００はアドレスの４バイト全て（バイト３から０までの３２ビット）をエクスポートする（処理ブロック４０４）。これらのバイトが等しい場合（判断ブロック４０３においてＹｅｓ）、プロセス４００はアドレスのバイト２をマスクと比較する（処理ブロック４０５）。これらのバイトが等しくない場合（判断ブロック４０６においてＮｏ）、プロセス４００はアドレスの下位３バイト（バイト２から０までの２４ビット）をエクスポートする（処理ブロック４０７）。これらのバイトが等しい場合（判断ブロック４０６においてＹｅｓ）、プロセス４００はアドレスのバイト１をマスクと比較する（処理ブロック４０８）。これらのバイトが等しくない場合（判断ブロック４０９においてＮｏ）、プロセス４００はアドレスの下位２バイト全て（バイト１および０の１６ビット）をエクスポートする（処理ブロック４１０）。これらのバイトが等しい場合（判断ブロック４０９においてＹｅｓ）、プロセスはアドレスの最下位バイト（バイト０の８ビット）のみをエクスポートする（処理ブロック４１１）。

【００３６】

このマスクの漸進的性質は、アドレスが特定の位置範囲内で発生される場合に、各基準と共に送る必要があるバイト数を削減するために利用できる。初期の基準アドレスまたはプログラム・カウンタ値は、送られるときに、４バイト全てを含んでいた。後続のレジスタ分岐またはデータ基準が図４のいずれかのテキスト条件を満たす場合、この方式で送出するバイト数は常に最大バイト数よりも少なくなる。データ・アドレスまたはプログラム・カウンタ・アドレスのタイミングによる分析(profiling)を行う場合、データ・アドレスおよびプログラム・コードの局在性(locality)があると、この方式は帯域幅の保存に関して一層効率的となる。新たな基準アドレスを、バイト毎に、ストレージ・レジスタに収容されている、以前の有効な基準アドレスと比較する。

【００３７】

同様の技法は、プログラマブル拡大圧縮マスク(programmable extended compression mask)にも用いることができる。現データまたはプログラム・カウンタ・アドレスを以前のアドレスと比較する代わりに、中央演算装置が供給する所定のアドレスとの比較を行う。この比較アドレスは、メモリ・マップされたデータ・レジスタによって指定することが好ましい。このマスクのプログラマブルな性質のため、コードおよびデータの位置に関して既知の情報を用いて、これらのアドレスの圧縮に役立てることができる。この方式では、トレース収集セッションに先だって、その内部のコードのアプリケーションまたはセグメントの実行において、どのデータおよびコード位置が参照される可能性が高いかについてのインテリジェントな判定を可能にする。複数のメモリ・マップ比較レジスタを用いれば、データおよびコードについて一層多くの区間記述(section description)を処理することができる。これは、より広い範囲のデータ・アドレスおよびコードを処理することができる。比較レジスタが複数個ある場合、トレース・データ・パケットをエクスポートする際、比較によってヒットした比較レジスタに付随するインデックスを、オフセット・アドレスと共に供給する。

【００３８】

図５は、アドレスをダイナミック・トレース・ストリームの一部としてトレースするための、漸進およびプログラマブル圧縮方式の併合を示す。トレース・アドレスは、アドレス入力５０１を介して受け取られる。トレース・ストリーム制御回路５０３は、出力５０２を介して８ビット・エンコード・トレース・パケットを供給する。トレース・ストリーム制御回路５０３は、トレース・アドレスと、ＮＯＲゲート５０４からのByte1-valid信号と、ＮＯＲゲート５０５からのByte2_valid信号と、ＮＯＲゲート５０５からのByte#_valid信号とを受け取る。ＮＯＲゲート５０４～５０６は、比較部５１０および５２０によって駆動される。

【００３９】

図５は、漸進（プログレッシブ）トレース比較部５１０を示す。入力５０１におけるトレース・アドレスのバイト１は、バイト１比較器５１１の一方の入力に供給される。入力５０１におけるトレース・アドレスのバイト２は、バイト２比較器５１２の一方の入力に供給される。入力５０１におけるトレース・アドレスのバイト３は、比較器５１３の一方の入力に供給される。比較器５１１、５１２および５１３は、対応するバイトを比較レジスタ５１６から受け取る。各比較器５１１、５１２および５１３は、入力５０１から受け取ったトレース・アドレスの対応するバイトの８ビットと比較レジスタ５１６に格納されている比較アドレスとを比較する。比較器５１１、５１２および５１３のそれぞれの比較出力は、ＮＯＲゲート５０４、５０５および５０６の１つの入力に供給される。

【００４０】

初期状態では、比較アドレス・レジスタ５１６は空、即ち、全て０である。初期化時に、最初のトレース・アドレスが、マルチプレクサ５１５を介して、比較アドレス・レジスタ５１６に格納される。２番目のトレース・アドレスは、現在比較アドレス・レジスタ５１６に格納されている最初のトレース・アドレスと比較される。比較器５１３において一致が得られた場合、トレース・ストリーム制御５０３はバイト３を出力しない。比較器５１３および５１２において一致が得られた場合、トレース・ストリーム制御５０３は、バイト３および２を出力しない。比較器５１３、５１２および５１１において一致が得られた場合、トレース・ストリーム制御５０３はバイト０のみを出力する。表１は、トレース・ストリーム制御回路５０３の出力機能を示す。

【表１】

バイト１	バイト２	バイト３	出力バイト
－	－	不一致	３２１０
－	－	一致	－２１０
－	一致	一致	――１０
一致	一致	一致	―――０

表１

【００４１】

尚、「－」はドント・ケア・エントリである。例えば、比較器５１２または５１１は一致を検出したが、比較器５１３が検出しない場合、４バイト全てが出力される。

【００４２】

比較アドレス・レジスタ５１６に格納されているアドレスは、変更されるまで、マルチプレクサ５１５の「０」入力を通って再循環する。これが発生するのは、通例では、３つの比較器５１１、５１２および５１３全てが一致を検出できなかったときである。バイト比較が全て不一致である場合、恐らくトレース・アドレスが異なるアドレス領域に移動していると考えられる。比較アドレスをこの新たなアドレス領域にリセットし、リセットした比較アドレスを用いて今後の比較を行うのが最良である。valid_address信号が「１」になると、マルチプレクサ５１５はトレース・アドレス入力５０１上において受け取ったトレース・アドレスを選択する。比較レジスタ５１６は、マルチプレクサ５１５が選択したこのアドレスを格納する。varid_address信号は、通常では、１回のサイクルの間「１」になっている。その後、新たに格納したトレース・アドレスを用いて比較を行う。

【００４３】

図５は、漸進トレース比較部５１０と同様の、プログラマブル・トレース比較部５２０を示す。入力５０１におけるトレース・アドレスのバイト１は、バイト１比較器５２１の一方の入力に供給される。入力５０１におけるトレース・アドレスのバイト２は、バイト２比較器５２２の一方の入力に供給される。入力５０１におけるトレース・アドレスのバイト３は、比較器５２３の一方の入力に供給される。比較器５２１、５２２および５２３は、比較レジスタ５２６から、対応するバイトを受け取る。各比較器５２１、５２２および５２３は、入力５０１から受け取ったトレース・アドレスの対応するバイトの８ビット

と、レジスタ５２６に格納されている比較アドレスとを比較する。比較器５２１、５２２および５２３のそれぞれの比較結果は、ＮＯＲゲート５０４、５０５、５０６の１つの入力に供給される。

【００４４】

比較アドレス・レジスタ５２６に格納されている比較アドレスは、通常では、マルチプレクサ５２５の「０」入力を介して再循環する。中央演算装置は、メモリ・マップ・レジスタに書き込むことによって、この比較アドレスを変更することができる。比較アドレス・レジスタ５２６に対応するメモリ・マップ・レジスタへの書き込み時に、ライト・バス５２７上にアドレスが現れ、Reg_write信号は「１」となる。マルチプレクサ５２５は、ライト・バス５２７上のアドレスを選択し、比較アドレス・レジスタ５２６に格納する。Reg_write信号は、通常では、１回のサイクルのみ、即ち、書き込みを完了するための最短時間だけ「１」になっている。その後、中央演算装置が供給し、現時点では比較アドレス・レジスタ５２６に格納されているアドレスを用いて比較が行われる。

【００４５】

図５に示す比較ユニット５１０および５２０のように、比較ユニットが１つよりも多い場合、トレース・ストリーム制御回路５０３は、何らかの方法（図示せず）で、動作状態にある比較ユニットを識別する必要がある。トレース・ストリーム比較回路５０３は、比較ユニット識別マーカを出力トレース・ストリーム５０２内に置く。更に、受信ユニットは、圧縮トレース・ストリームを追跡する(follow)手段を有する。新たな比較アドレスは、アドレス全体を受信ユニットに送信した直後に、比較アドレス・レジスタ５１６に格納される。中央演算装置のプログラムの制御によって、受信ユニットは、比較アドレス・レジスタ５２６に格納されている、中央演算装置が指定したアドレスを先験的に知ることができる。どの比較ユニットを用いるかがわかっていると、トレース・アドレスの再組立が可能になる。

【００４６】

漸進比較ユニット５１０またはプログラマブル比較ユニット５２０が１つよりも多い場合、更に複雑となる。プログラマブル比較ユニット５２０が複数個あると、問題は最小となる。恐らく、ユーザは、中央演算装置が指定したトレース・アドレスの制御を行うか、あるいは少なくともその知識を有する。その知識を比較ユニット識別マーカと照合することは、実施可能である。

【００４７】

漸進比較ユニット５１０が複数個ある場合、一層問題になる。各比較アドレス・レジスタ５１６内に格納されているデータを追跡するために、何らかの技法が必要となる。これは、トレース・ストリーム制御回路５０３が、いつ比較アドレス・レジスタ５１６にロードされたか、およびそのレジスタの識別を、トレース・ストリーム内で明示的に指示することによって、行うことができる。あるいは、３２ビット・トレース・アドレス全てが送信されたときにはいつでも、最も古い比較アドレス・レジスタが置換されることを想定することができる。複数の比較アドレス・レジスタの内容を追跡することによって、２つ以上のアドレス領域の比較が可能となり、トレース・ストリームの密度を高めることも可能となる。

【００４８】

本願では、対応するアドレス・バイトによって比較を行う技法について記載した。これは、単に便利な設計選択事項を表すに過ぎないことは当業者には認められよう。比較は、ニブル（４ビット）、ワード（１６ビット）、または個々のビットを含むその他の便利なデータ長であればいずれででも行うことができる。選択する比較長は、比較されるアドレスの長さの整数分の１とし、アドレス全体を通じて等しい長さであることが好ましい。用いる比較長が短い程、制御プロセスを追加するという負担があるが、データ圧縮を行える可能性が高くなる。比較長が長い程、必要な制御プロセスは少なくて済むが、潜在的に可能なデータ圧縮は減少する。

【００４９】

以上の説明に関して更に以下の項を開示する。
（１）トレース・アドレスの比較方法であって、
比較アドレスを格納するステップと、
現トレース・アドレスのそれぞれの等しい長さの区間（セクション）を、格納した比較アドレスと比較するステップと、
前記格納した比較アドレスのそれぞれの区間と一致しない前記現トレース・アドレスの最下位区間、または前記格納した比較アドレスと一致しない前記現トレース・アドレスの区間よりも下位にある前記トレース・アドレスの下位区間のみを送信することによって、前記現トレース・アドレスの指示を送信するステップと、
から成る、トレース・アドレスの比較方法。
（２）前記比較するステップは、前記現トレース・アドレスの区間を、前記格納した比較アドレスの対応する区間と、最上位区間から最下位区間まで順次比較する、第１項記載のトレース・アドレスの比較方法。
（３）　更に、中央演算装置のメモリ・マップ・レジスタへの書き込み動作によって、前記比較アドレスを指定するステップを含む、第１項記載のトレース・アドレスの比較方法。
（４）更に、前記現トレース・アドレスに、前記格納した比較アドレスのそれぞれの区間と一致する区間がない場合、前記現トレース・アドレスを前記比較アドレスとして格納するステップを含む、第１項記載のトレース・アドレスの比較方法。
（５）前記区間は、前記格納した以前のトレース・アドレスおよび前記現トレース・アドレスの長さの整数分の１である、第１項記載のトレース・アドレスの比較方法。
【００５０】
（６）トレース圧縮装置であって、
現トレース・アドレスを受け取る入力と、
比較アドレスを格納する比較アドレス・レジスタと、
複数の区間比較器であって、各区間比較器が前記比較アドレス・レジスタに接続されて前記比較アドレスの１区間を受け取り、更に前記入力に接続されて前記現トレース・アドレスの対応する区間を受け取り、各々、前記対応する区間の一致または不一致を示す、複数の区間比較器と、
前記入力と前記区間比較器の各々とに接続されたトレース・ストリーム制御部であって、前記格納した比較アドレスのそれぞれの区間と一致しない前記現トレース・アドレスの最下位区間、または前記格納した比較アドレスと一致しない前記現トレース・アドレスの区間よりも下位にある、前記現トレース・アドレスの下位区間のみを送信する、トレース・ストリーム制御部と、
を備えているトレース圧縮装置。
（７）更に、
前記比較アドレス・レジスタの出力を受け取る第１入力と、メモリ・マップ書き込みデータを受け取る第２入力と、前記比較レジスタの入力に接続された出力と、制御入力とを有するマルチプレクサであって、前記制御入力における信号に応じて、前記第１入力または前記第２入力の一方を前記出力に結合する、マルチプレクサを備えており、
前記マルチプレクサの前記制御入力が、メモリ・マップ書き込み指示を受け取り、該マルチプレクサが前記第２入力を選択することによって、前記比較アドレス・レジスタに対応するアドレスへのメモリ・マップ・レジスタ書き込み時に、前記メモリ・マップ書き込みデータを前記比較レジスタに格納する、第６項記載のトレース圧縮装置。
（８）更に、
前記比較アドレス・レジスタの出力を受け取る第１入力と、前記現トレース・アドレスを受け取る第２入力と、前記比較アドレス・レジスタの入力に接続された出力と、制御入力とを有するマルチプレクサであって、前記制御入力における信号に応じて前記第１入力または前記第２入力の一方を前記出力に結合する、マルチプレクサを備えており、
前記トレース・ストリーム制御部が、前記マルチプレクサの前記制御入力に接続されて

おり、前記現トレース・アドレスに前記比較アドレス・レジスタの対応する区間と一致する区間がない場合、前記トレース・ストリーム制御部が前記マルチプレクサの前記制御入力に信号を供給し、前記マルチプレクサに前記第２入力を選択させることによって、前記現トレース・アドレスを前記比較アドレス・レジスタに格納する、第６項記載のトレース圧縮装置。

（９）前記区間は、格納されている以前のアドレスおよび現トレース・アドレスの長さの整数分の１である、第６項記載のトレース・アドレス圧縮装置。

（１０）前記以前のトレース・アドレスおよび前記現アドレスは、３２ビットの長さを有し、

前記複数の区間比較器は、それぞれ、前記以前のトレース・アドレスおよび前記現トレース・アドレスのそれぞれ上位３バイトを比較する３つのバイト比較器から成り、

前記トレース・ストリーム制御部は、

最上位ビット比較器が一致を検出しない場合、前記現トレース・アドレスの４バイトを送信し、

最上位バイト比較器が一致を検出し、第２位バイト比較器が一致を検出しない場合、前記現トレース・アドレスの上位３バイトを送信し、

最上位バイト比較器および第２位バイト比較器が一致を検出し、第３位バイト比較器が一致を検出しない場合、前記現トレース・アドレスの上位２バイトを送信し、

最上位バイト比較器、第２位バイト比較器、および第３位バイト比較器が各々一致を検出した場合、前記現トレース・アドレスの最下位バイトを送信する、

ように動作する、第６項記載のトレース・アドレス圧縮装置。

【００５１】

（１１）トレース・アドレス圧縮方法および装置は、現トレース・アドレス（５０１）のそれぞれの区間を、格納されている比較アドレス（５１６、５２６）と比較する。それぞれの区間（５１１、５２１；５１２、５２２；５１３、５２３）に対する比較結果は、トレース・ストリーム・コントローラ（５０３）に供給される。トレース・ストリーム・コントローラ（５０３）は、比較アドレスと一致しない現トレース・アドレスの下位区間、または比較アドレスと一致しない現トレース・アドレスのいずれの区間よりも下位にある下位区間のみを送信する。これによって、送信するデータ量を削減する。比較アドレス（５２６）は、メモリ・マップ・レジスタ書き込み動作から得られる。比較アドレス（５１６）は、完全な不一致があった場合には、現トレース・アドレス（５０１）で更新する（５１５）ことができる。

【図面の簡単な説明】

【００５２】

【図１】典型的な集積回路の可視性および制御を、システム集積度の増大による時間の関数として示す図。

【図２】本発明を適用可能なエミュレーション・システムを示す図。

【図３】構成変更可能なエミュレーション機能を用いた典型的な集積回路を示すブロック図。

【図４】本発明の漸増（プログレッシブ）圧縮マスクの比較動作を示すフロー・チャート。

【図５】漸進およびプログラマブル圧縮マスク双方を含む回路を示すブロック図。

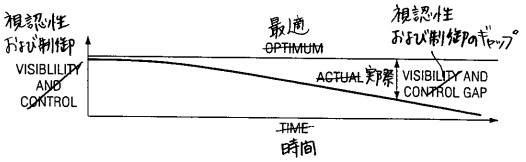
【符号の説明】

【００５３】

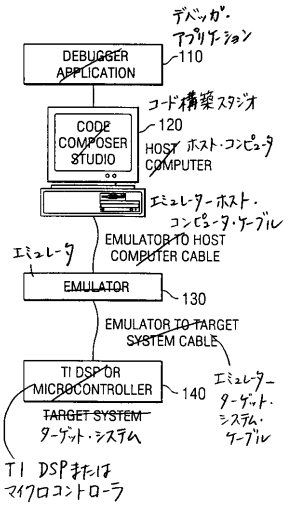
１００　エミュレータ・システム
１１０　デバッガ・アプリケーション・プログラム
１２０　ホスト・コンピュータ
１３０　エミュレーション・コントローラ
１４０　オンチップ・デバッグ機構
２０１　ＣＰＵコア

２１０　バス・イベント検出器
２１１　補助イベント検出器
２１３　カウンタ／ステートマシン
２２０　トリガ・ビルダ
２３０　トレース収集
２４０　トレース・エクスポート
２４１　リアル・タイム・データ交換エクスポート
２４５　局部発振器
２５０　走査アダプタ
２６０　ピン・マネージャおよびピン・マクロ
５０１　アドレス入力
５０２　出力
５０３　トレース・ストリーム制御回路
５０４、５０５、５０６　ＮＯＲゲート
５１０　漸進トレース比較部
５１１　バイト１比較器
５１２　バイト２比較器
５１３　バイト３比較器
５１５　マルチプレクサ
５１６　比較アドレス・レジスタ
５２０　プログラマブル比較ユニット
５２１、５２２、５２３　比較器
５２５　マルチプレクサ
５２６　比較アドレス・レジスタ
５２７　ライト・バス

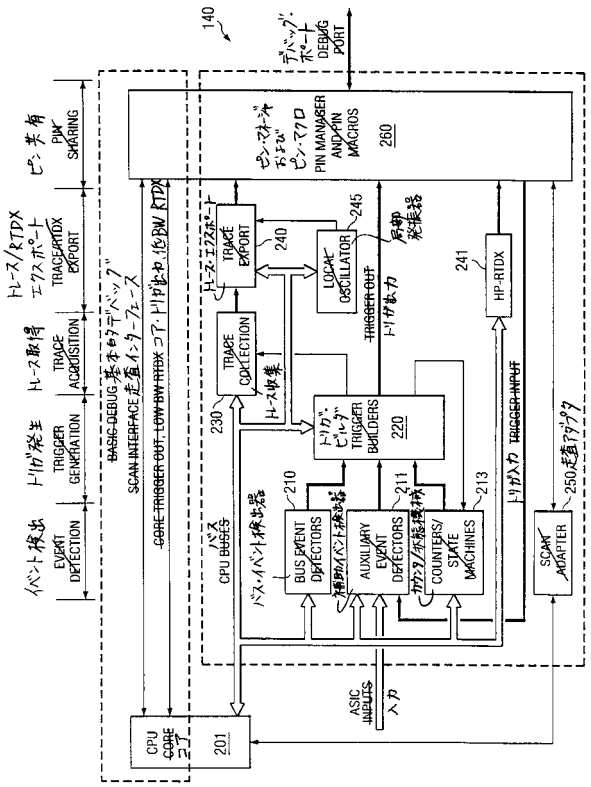
【図１】



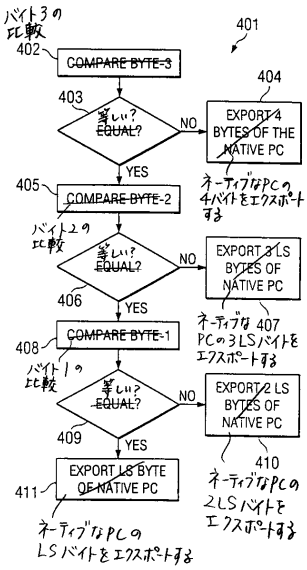
【図２】



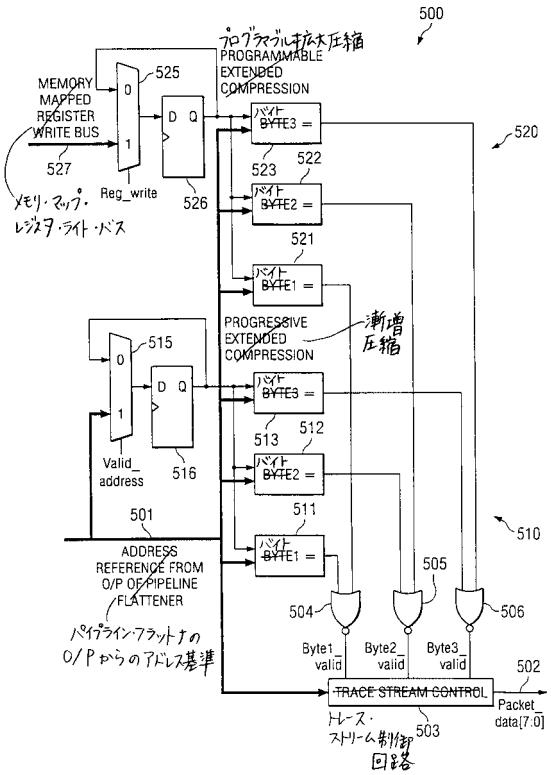
【図３】



【図４】


バイト3の比較
401
402
COMPARE BYTE-3
403
等しい? EQUAL?
NO
404
EXPORT 4 BYTES OF THE NATIVE PC
YES
405
COMPARE BYTE-2
ネーティブなPCの4バイトをエクスポートする
バイト2の比較
等しい? EQUAL?
NO
EXPORT 3 LS BYTES OF NATIVE PC
406
YES
407
ネーティブなPCの3LSバイトをエクスポートする
408
COMPARE BYTE-1
バイト1の比較
等しい? EQUAL?
NO
EXPORT 2 LS BYTES OF NATIVE PC
409
YES
410
ネーティブなPCの2LSバイトをエクスポートする
411
EXPORT LS BYTE OF NATIVE PC
ネーティブなPCのLSバイトをエクスポートする

【図５】


500
プログラマブル拡大圧縮
PROGRAMMABLE EXTENDED COMPRESSION
MEMORY MAPPED REGISTER WRITE BUS
メモリ・マップ・レジスタ・ライト・バス
525
0
1
D Q
527
Reg_write
526
520
バイト BYTE3 =
バイト BYTE2 =
バイト BYTE1 =
523
522
521
PROGRESSIVE EXTENDED COMPRESSION
漸増圧縮
515
0
1
D Q
Valid_address
516
バイト BYTE3 =
バイト BYTE2 =
バイト BYTE1 =
513
512
511
501
ADDRESS REFERENCE FROM O/P OF PIPELINE FLATTENER
パイプライン・フラットナのO/Pからのアドレス基準
510
505
504
506
Byte1_valid
Byte2_valid
Byte3_valid
502
TRACE STREAM CONTROL
Packet_data[7:0]
503
トレース・ストリーム制御回路

フロントページの続き

(72)発明者　ルイス、ナルディーニ
アメリカ合衆国　テキサス、ダラス、　フリント　フォールズ　ドライブ　８８２５
(72)発明者　マニーシャ　アガルワラ
アメリカ合衆国　テキサス、リチャードソン、　フォックスクリーク　ドライブ　２６０８
(72)発明者　ジョン　エム、ジョンセン
アメリカ合衆国　テキサス、ダラス、　ロイヤルトン　ドライブ　６４６５

審査官　多胡　滋

(56)参考文献　特開２００２－１４９４４２（ＪＰ，Ａ）
特開２００１－３５６９３５（ＪＰ，Ａ）
特開２０００－３２２２９９（ＪＰ，Ａ）

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