(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5459807号
(P5459807)
(45)発行日　平成26年4月2日(2014.4.2)　(24)登録日　平成26年1月24日(2014.1.24)

(51)Int.Cl.　　F I
G06F 11/28 (2006.01)　　G06F 11/28　J
G06F 11/16 (2006.01)　　G06F 11/16　310E
G06F 11/18 (2006.01)　　G06F 11/18　310F

請求項の数 5　(全 20 頁)

(21)出願番号　特願2012-504697 (P2012-504697)
(86)(22)出願日　平成22年3月23日 (2010.3.23)
(65)公表番号　特表2012-523616 (P2012-523616A)
(43)公表日　平成24年10月4日 (2012.10.4)
(86)国際出願番号　PCT/US2010/028300
(87)国際公開番号　WO2010/117618
(87)国際公開日　平成22年10月14日 (2010.10.14)
審査請求日　平成25年3月22日 (2013.3.22)
(31)優先権主張番号　12/420,521
(32)優先日　平成21年4月8日 (2009.4.8)
(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　メドー　リッジ　ドライブ　１１１１

最終頁に続く

(54)【発明の名称】マルチプロセッサデータ処理システムにおけるデバッグシグナリング

(57)【特許請求の範囲】
【請求項１】
第１のプロセッサと、
第２のプロセッサと、
第１のプロセッサに接続された第１のクロックと、
第２のプロセッサに接続された第２のクロックと、
第１のプロセッサおよび第２のプロセッサに接続されるとともに第１のクロックに対して同期されていない第３のクロックと、を備えるシステムにおいて、
第１のプロセッサは、
第３のクロックを受信するように接続されたデバッグ回路と、
第１のクロックを受信するように接続された同期化回路であって、第３のクロックに対して同期されているデバッグモードに入るための第１の要求を受信し、第１の同期化デバッグ開始要求信号を供給する、同期化回路と、
第２のプロセッサから第２の同期化デバッグ開始要求信号を受信するための入力と、を備え、
第１の同期化デバッグ開始要求信号は第１のクロックに対して同期化されており、
第１のプロセッサは、第１の同期化デバッグ開始要求信号および第２の同期化デバッグ開始要求信号の両方がアサートされるまで、デバッグモードに入ることを待機する、システム。
【請求項２】

デバッグ回路はデバッグ要求レジスタビットを備え、デバッグ要求レジスタビットのアサートに応答して、デバッグモードに入るための第１の要求は、同期化回路に対し供給される、請求項１に記載のシステム。

【請求項３】

第１のプロセッサおよび第２のプロセッサはデバッグモードに入り、該デバッグモード中、ロックステップ方式により同じ命令を実行する、請求項１に記載のシステム。

【請求項４】

同期化回路はデバッグモードから出るための要求をデバッグ回路から受信し、第１の同期化されたデバッグ終了コマンド信号を供給し、第１の同期化されたデバッグ終了コマンド信号は第１のクロックに対して同期化されており、第１のプロセッサはさらに、

第１の同期化デバッグ開始要求信号を第１のプロセッサから第２のプロセッサに転送するための第１の出力と、

第１の同期化デバッグ終了命令信号を第１のプロセッサから第２のプロセッサに転送するための第２の出力と、

第２の同期化デバッグ終了命令信号を第２のプロセッサから受信するための入力とを備え、

第１のプロセッサは、第１の同期化されたデバッグ終了コマンド信号および第２の同期化されたデバッグ終了コマンド信号の両方がアサートされるまで、デバッグモードを出ることを待機する、請求項１に記載のシステム。

【請求項５】

第１のクロックを用いてデバッグ回路の第１の部分にクロックを供給する工程と、

第２のクロックを用いてデバッグ回路の第２の部分にクロックを供給する工程であって、第１のクロックは第２のクロックに対して同期されていない工程と、

デバッグモードに入るための第１の要求をデバッグ回路の第１の部分から同期化回路に転送する工程であって、デバッグモードに入るための第１の要求は第２のクロックに対して同期されていない工程と、

第１のプロセッサが、第１の同期化デバッグ開始要求を生成するために、デバッグモードに入るための第１の要求を第２のクロックと同期させる工程と、

第１のプロセッサが、第２のプロセッサから受信されている第２の同期化デバッグ開始要求を監視する工程と、

第１のプロセッサが、第１の同期化デバッグ開始要求および第２の同期化デバッグ開始要求の両方がアサートされるまで、デバッグモードに入ることを待機する工程と、を含む方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、一般にデータ処理に関し、より詳細には、マルチプロセッサデータ処理システムにおけるデバッグシグナリングに関する。

【背景技術】

【０００２】

集積回路データ処理システムの幾つかの用途では、平均的な信頼性のレベルよりも高い信頼性が要求される。例えば、フライ・バイ・ワイヤ、アンチロックブレーキ、自動車エアバッグのシステムや、故障によって事故が発生し得るシステムは、高信頼性の動作を要求するシステムの例である。

【０００３】

信頼性を改善する多くの方法が存在する。例えば、メモリでは、第１の部品の故障時に引き継ぎを行う冗長的な部品を追加することによって、信頼性が改善される。マルチプロセッサシステムでは、「ロックステップ」（ｌｏｃｋｓｔｅｐ）方式で複数のプロセッサを作動させることによって、より優れた信頼性が達成される。２つ以上のプロセッサがロックステップ方式で作動するとき、各プロセッサは同時にまたは互いの所定スキュー以内

（すなわち、互いの所定のクロック数以内）で同じ命令ストリームを実行する。しかしながら、そのようなマルチプロセッサシステムのデバッグ時には問題が生じる。例えば、１つのプロセッサの動作はデバッグ開始および終了命令の続く同期とは異なる場合があるので、マルチプロセッサシステム内の１つ以上のプロセッサのプロセッサクロック領域に対するデバッグポートの非同期性によって、ロックステップに残る問題が生じる。すなわち、マルチプロセッサシステム内の別のプロセッサは、デバッグ開始および終了命令を同じクロックサイクルにおいて同期できない場合があり、そのため、デバッグモードへの出入りを認識するのに遅延が生じて、ロックステップの損失を生じる。

【図面の簡単な説明】

【０００４】

【図１】一実施形態によるマルチプロセッサシステムのブロック図。

【図２】一実施形態による図１のマルチプロセッサシステム内のプロセッサのデバッグ制御の一部のブロック図。

【図３】一実施形態による図２のデバッグ制御のデバッグ命令レジスタの図。

【図４】一実施形態による図３のデバッグ命令レジスタの様々なフィールドを表形式で示す図。

【図５】一実施形態による図２のデバッグ制御のデバッグ制御レジスタの図。

【図６】一実施形態による図５のデバッグ制御レジスタのフィールドを表形式で示す図。

【図７】非ロックステップモードによるデバッグ開始シグナリングの一例における図１または図２の様々な信号のタイミング図。

【図８】ロックステップモードによるデバッグ開始シグナリングの一例における図１または図２の様々な信号のタイミング図。

【図９】ロックステップモードによるデバッグ開始シグナリングの別例における図１または図２の様々な信号のタイミング図。

【図１０】非ロックステップモードによるデバッグ終了シグナリングの一例における図１または図２の様々な信号のタイミング図。

【図１１】ロックステップモードによるデバッグ終了シグナリングの一例における図１または図２の様々な信号のタイミング図。

【図１２】ロックステップモードによるデバッグ終了シグナリングの別例における図１または図２の様々な信号のタイミング図。

【発明を実施するための形態】

【０００５】

本発明は例示の方法により説明されており、添付の図面により限定されるものではなく、図面において、同様の参照符号は類似の要素を示す。図中の要素は簡潔かつ明確に説明されており、必ずしも寸法通りに描かれていない。

【０００６】

一般に、複数のプロセッサ、コア、または中央処理ユニット（ＣＰＵ）が、ロックステップ方式によってなど同期される方式により動作する、マルチプロセッサデータ処理システムが提供される。しかしながら、デバッグモードへの出入りなどデバッグ動作の非同期性のため、ロックステップが失われることがある。例えば、一対のプロセッサをロックステップ方式により作動させるとき、同期されていない入力がプロセッサのクロックに対して同期されることを保証するため、各プロセッサにおいて同期化回路が使用される。しかしながら、このような同期化回路内に生じ得る準安定性のため、その２つのプロセッサの各々における同期化回路の同期された出力は、実際には異なることがあり、プロセッサクロックに対して非同期的に動作するデバッグ制御インタフェースを用いてデバッグ動作を実行する場合、ロックステップ動作が失われる。この場合、デバッガは、ロックステップの喪失を考慮して、再同期を試みる必要がある。したがって、一実施形態では、プロセッサ間でのデバッグモードの出入りを協調させるために、ロックステップデバッグインタフェースなどクロスシグナリングインタフェースが用いられる（例えば、両方のプロセッサが同時にまたは互いの所定のクロックサイクル数以内でデバッグモードに出入りすること

が保証される）。一実施形態では、このクロスシグナリングによって、１つのプロセッサにおけるデバッグの開始および終了を別のプロセッサが同じ要求を認識するまで条件付で遅延させることによって、プロセッサがロックステップ方式に維持されることを保証する。また、一実施形態では、プロセッサが実際にロックステップモードにより動作しているか否かに基づき、条件付で、このクロスシグナリングインタフェースが用いられる。

【０００７】

本明細書において使用されるものとして、用語「バス」は、１以上の各種の情報、例えばデータ、アドレス、制御又は状態を送信するために使用される複数の信号又は導体を参照するものとして使用される。本明細書において説明される導体は、単一の導体、複数の導体、単方向の導体、又は双方句の導体として参照して図示又は説明される。しかしながら、異なる実施形態は導体の実施を変更してもよい。例えば、別々の単方向の導体が双方向の導体に代えて使用されてもよく、反対に、双方向の導体が別々の単方向の導体に代えて使用されてもよい。また、複数の信号を連続的に送信したり、時間多重化して送信したりする単一の導体によって、複数の導体が置換されてもよい。同様に、複数の信号を搬送する単一の導体が、それらの信号の一部分を搬送する様々な異なる導体に分割されてもよい。従って、信号を送信するための多くの選択肢が存在する。

【０００８】

「アサート」または「セット」及び「ネゲート」（または「ディアサート」もしくは「クリア」）という語は、本明細書において、それぞれ、信号、状態ビット、または同様の装置を論理的に真または論理的に偽の状態にすることを指して用いられる。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ゼロ（０）である。また、論理的に真の状態が論理レベルゼロの場合、論理的に偽の状態は論理レベル１である。

【０００９】

本明細書に記載の各信号は正論理または負論理として設計され得る。ここで、負論理は、信号名の上の棒（バー）か、信号名に続く文字「Ｂ」で示される。負論理信号の場合、信号はアクティブローであり、論理的に真の状態は論理レベル０に相当する。正論理信号の場合、信号はアクティブハイであり、論理的に真の状態は論理レベル１に相当する。本明細書に記載の任意の信号は、負論理または正論理の信号として設計され得る。従って、代替の実施形態では、正論理信号として記載される信号が負論理信号として実装されてもよく、負論理信号として記載される信号が正論理信号として実装されてもよい。

【００１０】

図１には、一実施形態によるデータ処理システム１０の簡略化したブロック図を示す。システム１０は、プロセッサ１２、プロセッサ１４、他のモジュール２２、システム相互接続部２４、およびデバッグインタフェース２０を含む。プロセッサ１２は、デバッグ制御１６、ロックステップモードイネーブル３２、プログラムカウンタ１５、プロセッサ制御論理３８を含む。ロックステップモードイネーブル３２、プログラムカウンタ１５、およびプロセッサ制御論理３８の各々は、デバッグ制御１６に接続される。ロックステップモードイネーブル３２はロックステップモードインジケータ４２をデバッグ制御１６に与え、プロセッサ制御論理３８が信号４０を介してデバッグ制御１６と通信する。プロセッサ１４は、デバッグ制御１８、ロックステップモードイネーブル３４、プログラムカウンタ１７、およびプロセッサ制御論理３９を備える。ロックステップモードイネーブル３４、プログラムカウンタ１７、およびプロセッサ制御論理３９の各々は、デバッグ制御１８に接続される。ロックステップモードイネーブル３４はロックステップモードインジケータ３６をデバッグ制御１８に与え、プロセッサ制御論理３９が信号４１を介してデバッグ制御１８と通信する。図示された実施形態では、プロセッサ１２，１４は実質的に同一であり、例えば、汎用プロセッサ、デジタル信号プロセッサ（ＤＳＰ）等、任意の種類のプロセッサであってよい。他の実施形態では、プロセッサ１２，１４は異なってもよい。例えば、プロセッサ１２は汎用プロセッサであり、プロセッサ１４はＤＳＰであってもよい。また、２つのプロセッサしか示していないが、本明細書に記載の実施形態が２つ以上のプロセッサを有するシステムにも適用され得ることが当業者には理解される。加えて、他

の実施形態では、プロセッサ１２，１４は図示したものとは異なる論理ブロックを備えてもよく、図１に示されていない追加の論理ブロックが存在してもよい。例えば、プロセッサ１２，１４の各々は、１つ以上の実行ユニット、命令フェッチユニット、命令デコードユニット、バスインタフェースユニット等を備えてよい。したがって、処理制御論理３８，３９は、それぞれのプロセッサ１２，１４の動作を制御することが可能である。プロセッサ１２，１４の動作は既知であるので、本発明の実施形態を説明するために必要と考えられる範囲を超える説明は行わない。

【００１１】

プロセッサ１２，１４および他のモジュール２２（存在する場合）は、システム相互接続部２４に双方向的に接続される。なお、図１に示されていない、システム相互接続部２４に接続された追加の機能ブロックが存在してもよい。一実施形態では、システム相互接続部２４は、システムの各ブロックに接続された複数の導体を含むバスとして具体化されてもよい。別の実施形態では、システム相互接続部２４は、システムブロック間における同時通信を可能とする従来の「クロスバー」型バスであってもよい。別の実施形態では、システム相互接続部２４は、ＡＨＢ（Ａｄｖａｎｃｅｄ　Ｈｉｇｈ－ｐｅｒｆｏｒｍａｎｃｅ　Ｂｕｓ）であってよい。ＡＨＢは、ＡＲＭ社（ＡＲＭ　Ｌｔｄ　Ｃｏｍｐａｎｙ）によって公開されたＡＭＢＡ仕様第２版において紹介されているバスプロトコルである。さらに別の実施形態では、システム相互接続部２４は、別のタイプのシステム相互接続システムであってよい。

【００１２】

システム１０は、プロセッサ１２におけるデバッグ制御１６とプロセッサ１４におけるデバッグ制御１８との間のロックステップデバッグインタフェース３０を含む。一実施形態では、ロックステップデバッグインタフェース３０は、次の信号、すなわち、同期化デバッグ開始要求２５（デバッグ制御１６からデバッグ制御１８に与えられる）、同期化デバッグ終了命令２６（デバッグ制御１６からデバッグ制御１８に与えられる）、同期化デバッグ開始要求２７（デバッグ制御１８からデバッグ制御１６に与えられる）、および同期化デバッグ終了命令２８（デバッグ制御１８からデバッグ制御１６に与えられる）を含む。なお、デバッグ制御１６と１８との間に追加の信号が存在してもよい。デバッグインタフェース２０は、デバッグ制御１６，１８と外部デバッガ（図示せず）との間のインタフェースを行う。例えば、一実施形態では、デバッグインタフェース２０はＪＴＡＧポートまたはその一部であってよい。デバッグインタフェース２０は、外部デバッガとシステム１０のそれぞれのプロセッサに配置されたデバッグ制御との間において、デバッグの命令および結果の通信を行うことができる。

【００１３】

動作時には、プロセッサ１２，１４は、同じプロセッサクロック、すなわち、ＰＣＬＫ５４（図１には示さず）を用いて動作し、互いにロックステップ方式により動作可能である。一実施形態では、プロセッサ１２，１４の各々におけるロックステップモードイネーブルは、ロックステップモードをイネーブルするようにアサートされることが可能である。一実施形態では、ロックステップモードは、システム１０内においてプロセッサ１２，１４の内部または外部にある回路（図示せず）によって与えられたシステム信号に応答してイネーブルされる。代替の実施形態では、プロセッサ１２，１４が常にロックステップモードにより動作してもよい。プロセッサ１２，１４がロックステップ方式により作動する場合、各プロセッサは、同時にまたは互いの所定のスキュー以内（すなわち、互いの所定のクロック数以内）で同じ命令ストリームを実行する。例えば、一実施形態では、ロックステップ方式により作動する場合、プロセッサ１４における同じ命令の実行は、プロセッサ１２における同じ命令の実行に比較して、ＰＣＬＫについて１０サイクル未満だけスキューされる。また、プロセッサ１２，１４がロックステップモード方式により作動する場合（デバッグモードにおいても）、各プロセッサは、その対応するプログラムカウンタ（それぞれプログラムカウンタ１５およびプログラムカウンタ１７）を同時に更新する。

【００１４】

ロックステップ方式により作動する場合、デバッグ命令は、デバッグインタフェース２０を介して外部デバッガからデバッグ制御１６，１８へ与えられることが可能である。そのデバッグ命令に応答して、プロセッサ１２及びプロセッサ１４の各々はデバッグモードに入ることができる。しかしながら、ロックステップ方式による作動を維持するために、プロセッサ１２およびプロセッサ１４は、同時にまたは互いの所定のクロック数以内にデバッグモードに入る必要がある。幾つかの実施形態では、デバッグインタフェース２０は、ＴＣＬＫ５６などクロックを用いて動作する。このクロックは、プロセッサ１２，１４によって用いられるクロックに対して同期されておらず、また異なる周波数でもよく、一実施形態では十分に低い周波数である。したがって、一実施形態では、デバッグモードに入ることが必要なとき、各プロセッサは、自身がデバッグ命令を認識し、デバッグモードに入る準備が済むとともに、他方のプロセッサが同じデバッグ命令を認識し、やはりデバッグモードに入る準備が済むまで待機する。したがって、一実施形態では、１つのプロセッサがデバッグモードに入る準備が済むとき、そのプロセッサのデバッグ制御が同期化デバッグ開始要求信号をアサートするので、そのプロセッサがデバッグモードに入る準備が済むときを他方のプロセッサが認識できる。このようにして、プロセッサは、他方のプロセッサがデバッグモードに入る準備が済むまで、デバッグに入るのを待機する。同様な説明が、デバッグモードから出ることに適用される。ここで、プロセッサがデバッグモードから出るとき、そのプロセッサのデバッグ制御が同期化デバッグ終了命令信号をアサートするので、そのプロセッサがデバッグモードから出る準備が済むときを他方のプロセッサが認識できる。動作のさらなる詳細について、図２～１２を参照して記載する。

【００１５】

図２には、デバッグ制御１６の一部をブロック図により示す。デバッグ制御１６は、制御回路４３、同期化回路４８、デバッグ制御レジスタ４４、およびデバッグ命令レジスタ４６を備える。制御回路４３は、ロックステップモードインジケータ４２、プログラムカウンタ１５、ＰＣＬＫ５４、プロセッサ１４からの同期化デバッグ開始要求２７、プロセッサ１４からの同期化デバッグ終了命令２８、および同期化回路４８からの同期化された出力５２を受信する。なお、同期化出力５２は、同期化されたＤＲ５１および同期化ＧＯ５３を含むことができる。また、制御回路４３は、信号４０を介してプロセッサ制御論理３８と通信する。同期化回路４８はＰＣＬＫ５４を受信し、同期化された出力５２を制御回路４３に与える。デバッグ制御レジスタ４４およびデバッグ命令レジスタ４６は、デバッグインタフェース２０と通信し、同期化回路４８に非同期入力５０を与える。デバッグ回路５８は、デバッグ制御レジスタ４４およびデバッグ命令レジスタ４６を備え、テストクロック（ＴＣＬＫ）５６を受信する。したがって、デバッグ回路５８が、ＰＣＬＫ５４に対して同期されていなくてよいＴＣＬＫ５６にしたがって動作することが留意される。したがって、非同期入力５０はＰＣＬＫ５４に対して同期されていなくてもよい。しかしながら、これに代えて、ＴＣＬＫ５６がＰＣＬＫ５４に対して同期されていてよい。また、デバッグインタフェース２０から入って来る入力は、ＰＣＬＫ５４に同期されていなくてもよい。したがって、デバッグ回路（例えば、デバッグ回路５８）の第１の部分は第１のクロック（例えば、ＴＣＬＫ５６）を用いて動作する一方、デバッグ回路の第２の部分（例えば、同期化回路４８および制御回路４３）は第１のクロックに対して同期されていない第２のクロック（例えば、ＰＣＬＫ５４）を用いて動作することが可能である。図２の動作については、図３～６を参照してデバッグ制御レジスタ４４およびデバッグ命令レジスタ４６について説明した後、より詳しく記載する。

【００１６】

図３には、本発明の一実施形態によるデバッグ命令レジスタ４６を示す。デバッグ命令レジスタ４６は、読取／書込命令ビットを格納するためのＲ／Ｗフィールド６０、入命令ビットを格納するためのＧＯフィールド６２、出命令ビットを格納するためのＥＸフィールド６４、およびレジスタ選択インジケータを格納するためのＲＳフィールド６６を含む。図４には、デバッグ命令レジスタ４６における各フィールドの動作について記載する表を示す。読取／書込命令ビットによって、データ転送の方向が指定される。例えば、Ｒ／

Ｗ６０が論理レベル０にクリアされるとき、命令に関連したデータはＲＳ６６によって指定されるレジスタに書き込まれ、論理レベル１に設定されるとき、ＲＳ６６によって指定されるレジスタに格納されているデータが読み取られる。ＧＯ６２が論理レベル０にクリアされるとき、動作は実行されない。ＧＯ６２が論理レベル１に設定されるとき、（プロセッサ１２内の）命令レジスタ（ＩＲ）における命令が実行される。この命令を実行するために、プロセッサ１２はデバッグモードから抜け、命令を実行する。ＥＸ６４が論理レベル０にクリアされるとき、プロセッサ１２は、命令の実行直後、デバッグモードに戻る。プロセッサ１２は、ＥＸ６４が論理レベル１に設定されるとき、通常の動作を継続し、他のデバッグ要求源はアサートされない。したがって、ＥＸ６４が論理レベル１に設定されるとき、プロセッサ１２はデバッグモードを抜けて、別のデバッグ要求が生成されるまで通常動作を再開する。

【００１７】

図５には、本発明の一実施形態によるデバッグ制御レジスタ４４を示す。デバッグ制御レジスタ４４は、デバッグ要求制御ビットを格納するためのＤＲフィールド７０を備える。図６には、ＤＲ７０の動作を説明する表を示す。ＤＲ７０は、無条件にプロセッサ１２にデバッグモードに入ることを要求するために用いられる。したがって、ＤＲ７０が論理レベル０にクリアされるとき、デバッグモード要求は行われず、論理レベル１に設定されるとき、プロセッサ１２は次の命令境界においてデバッグモードに入る。

【００１８】

なお、デバッグ命令レジスタ４６およびデバッグ制御レジスタ４４は、任意の数のフィールドを各々有する１つ以上のレジスタを用いて実装されてよく、各フィールドは任意の数のビットを有し、任意に編成されてよい。

【００１９】

従って、戻って図２を参照すると、例えば、外部デバッガからの命令に応答して、デバッグモードに出入りするための要求を生成するために、デバッグインタフェース２０がレジスタ４４，４６のフィールドを設定またはクリアしてよい。デバッグ回路５８はＴＣＬＫ５６にしたがって動作するので、レジスタ４４，４６のフィールドは、デバッグインタフェース２０からの命令に応答して、ＴＣＬＫ５６にしたがって更新される（例えば、ＴＣＬＫの立ち上がりまたは立ち下がり縁に応答して）。同期化回路４８は、レジスタ４４，４６から非同期入力５０を受信する。例えば、ＤＲ７０またはＧＯ６２が設定またはクリアされるとき、それらの入力は同期化回路４８によって入力として受信される。これらのビットはＴＣＬＫ５６に同期して設定またはクリアされるので、ＰＣＬＫ５４に対して同期されていなくてもよい。同期化回路４８は、ＰＣＬＫ５４に従って動作し、ＤＲ７０およびＧＯ６２をＰＣＬＫ５４に同期させて、ＤＲ７０の同期化されたバージョン、すなわち、同期化されたＤＲ５１と、ＧＯ６２の同期化されたバージョン、すなわち、同期化ＧＯ５３とを生成し、それらは各々ＰＣＬＫ５４に対し同期化される。したがって、同期化回路４８は、制御回路４３に対し対応する同期された出力５２（ここで、同期化ＤＲ５１および同期化ＧＯ５３は同期化出力５２に含まれてよい）を与えるように、デバッグ回路５８から受信された非同期入力５０を同期させるように動作する。

【００２０】

これらの同期化出力を用いて、制御回路４３は、プロセッサ１４に対し、制御回路４３が同期され、適切な動作（デバッグモードの出入りなど）を行う準備が済んだことを適切にアラートする。例えば、デバッグモードに入ることのＤＲ７０のアサートに応答して、同期化回路４８は同期化ＤＲ５１を制御回路４３に与える。この点において、プロセッサ１２がデバッグモードに入る準備が済んだと考えられてもよい。したがって、制御回路４３は、同期化ＤＲ５１のアサートに応答して、プロセッサ１４のデバッグ制御１８に同期化デバッグ開始要求２５を与えて、プロセッサ１２がデバッグモードに入る準備が済んだことをデバッグ制御１８に指示することが可能である。しかしながら、プロセッサ１２は、プロセッサ１４のデバッグ制御１８からのアサートされた同期化デバッグ開始要求２７が受信され、プロセッサ１４もデバッグモードに入る要求を受信し（その対応するデバッ

グ制御レジスタのＤＲビットのアサートを介して）、それに応じて、デバッグモードに入る準備が済んでいることを示すまで、デバッグモードに入らない。したがって、プロセッサ１２は、プロセッサ１４がデバッグモードに入る準備が済むまで、デバッグモードに入らない。反対に、プロセッサ１４は、プロセッサ１２がデバッグモードに入る準備が済むまで、デバッグモードに入らない。このようにして、両方のプロセッサはロックステップ方式により、すなわち同時に、デバッグモードに入ることが可能である。プロセッサ１４は、同様に、プロセッサ１２が、プロセッサ１２がデバッグモードに入る準備が済んだことをプロセッサ１４にアラートし、プロセッサ１４自身もデバッグモードに入る準備が済むまで、デバッグモードに入らない。

【００２１】

同様な動作が、デバッグモードから出ることに適用される。例えば、デバッグモードから出ることのＧＯ６２のアサートに応答して、同期化回路４８は同期化ＧＯ５３を制御回路４３に与える。この点において、プロセッサ１２がデバッグモードから出る準備が済んだと考えられてもよい。したがって、制御回路４３は、同期化ＧＯ５３のアサートに応答して、プロセッサ１４またはデバッグ制御１８に同期化デバッグ終了命令２６を与えて、プロセッサ１２がデバッグモードから出る準備が済んだことをデバッグ制御１８に指示することが可能である。しかしながら、プロセッサ１２は、プロセッサ１４のデバッグ制御１８からのアサートされた同期化デバッグ終了命令２７が受信され、プロセッサ１４もデバッグモードから出る命令を受信し（その対応するデバッグ命令レジスタのＧＯビットのアサートを介して）、それに応じて、デバッグモードから出る準備が済んでいることを示すまで、デバッグモードから出ない。したがって、プロセッサ１２は、プロセッサ１４がデバッグモードから出る準備が済むまで、デバッグモードから出ない。反対に、プロセッサ１４は、プロセッサ１２がデバッグモードから出る準備が済むまで、デバッグモードから出ない。このようにして、両方のプロセッサはロックステップ方式により、すなわち同時に、デバッグモードから出ることが可能である。プロセッサ１４は、同様に、プロセッサ１２が、プロセッサ１２がデバッグモードから出る準備が済んだことをプロセッサ１４にアラートし、プロセッサ１４自身もデバッグモードから出る準備が済むまで、デバッグモードから出ない。

【００２２】

一実施形態では、プロセッサ１２およびプロセッサ１４がデバッグモードに入ると、デバッグインタフェース２０を介して外部デバッグによって要求される続く動作は、両方のプロセッサにおいてＴＣＬＫ５６クロックを用いて実行されることができ、プロセッサ１２とプロセッサ１４との間では、それらの動作の同期および続くイベントのハンドシェークが必要ないので、通信が簡略化される。代替の実施形態では、プロセッサクロックＰＣＬＫ５４は、要求される動作を実行するために両方のプロセッサにおいて用いられてもよく、ＰＣＬＫ５４は、デバッグ制御１６内の制御５８に同期されている代替のクロックに切り替えられてもよいので、制御シグナリングのさらなる同期化の必要が回避される。このデバッグまたはテストクロックと汎用プロセッサのクロックとの間のクロックハンドオフによって、幾つかの実施形態においてより高度にシグナリングを簡略化することが可能となる。

【００２３】

なお、一実施形態では、同期化デバッグ開始要求２５は同期化ＤＲ５１の遅延されたバージョンである（例えば、組み合わせまたは連続した論理を通じた伝搬による遅延）。すなわち、同期化デバッグ開始要求２５は同期化ＤＲ５１と同じ信号であってよく、ＰＣＬＫ５４の１以上のクロックサイクルだけ遅延されているだけである。すなわち、同期化ＤＲ５１および同期化デバッグ開始要求２５の両方は、同じイベントに応答してアサートされる（すなわち、それらの両方は、デバッグ回路５８を介して受信されたデバッグモードに入るための要求を同期化回路４８が同期化することに応答してアサートされる）。同期化デバッグ終了命令２６の場合も、同期化デバッグ終了命令２６が単に同期化ＧＯ５３の遅延されたバージョンであるという点で同じであってよい。すなわち、同期化ＧＯ５３お

よび同期化デバッグ終了命令信号２６の両方は、同じイベントに応答してアサートされる（すなわち、それらの両方は、デバッグ回路５８を介して受信されたデバッグモードから出るための要求を同期化回路４８が同期化することに応答してアサートされる）。

【００２４】

また、同期化回路４８によって出力される同期化ＤＲ５１の任意の部分または同期化デバッグ開始要求２５の任意の部分は、プロセッサ１２がデバッグモードに入る時を決定する同期化デバッグ開始要求信号として用いられることができる。例えば、一実施形態では、プロセッサ１２は、プロセッサ１４から受信された同期化ＤＲ５１および同期化デバッグ開始要求２７の両方がアサートされるまで、デバッグモードに入るのを待機する。代替の実施形態では、プロセッサ１２は、プロセッサ１４から受信された同期化デバッグ開始２５および同期化デバッグ開始要求信号２７の両方がアサートされるまで、デバッグモードに入るのを待機する。したがって、上述のように、同じイベントに応答して同期化ＤＲ５２および同期化デバッグ開始要求２５の両方がアサートされるので、同期化ＤＲ５１または同期化デバッグ開始要求２５が同期化デバッグ開始要求信号（デバッグモードに入る時を決定する際にプロセッサ１２によって用いられる）として用いられてもよいことが留意される。同様に、代替の実施形態では、同期化回路４８によって出力される同期化ＧＯ５３の任意の部分または同期化デバッグ終了命令２６の任意の部分は、プロセッサ１２がデバッグモードから出る時を決定する同期化デバッグ終了命令信号として用いられることができる。

【００２５】

図２を参照すると、任意のタイプの周知の同期化回路を用いて、入力５０をＰＣＬＫ５４に対し同期させることができる。例えば、一実施形態では、各入力および対応する出力について、同期化回路は、ＰＣＬＫ５４によってクロックの与えられる複数の直列に接続されたＤタイプのフリップフロップを含むことが可能である。非同期入力は最初の直列に接続されたフリップフロップのデータ入力において与えられることが可能であり、対応する同期出力は最後の直列に接続されたフリップフロップのデータ出力において与えられることが可能である。ここで、データ出力は次いでＰＣＬＫ５４に対して同期される。例えば、一実施形態では、ＤＲ７０およびＧＯ６２の各々は一連の３つの直列に接続されたＤタイプフリップフロップに対し与えられることが可能である。ここで、直列における第１のフリップフロップのデータ入力がＤＲ７０またはＧＯ６２を受信し、直列における第３のかつ最後のフリップフロップのデータ出力がＤＲ５１またはＧＯ５３を与える。これに代えて、任意の数のフリップフロップまたは他のタイプの同期化回路が用いられてもよい。

【００２６】

一実施形態では、デバッグ制御回路１８はデバッグ制御回路１６と同じであり、制御回路、同期化回路、デバッグ制御レジスタ、およびデバッグ命令レジスタを備え、それらはデバッグ制御回路１６と同じように接続され、同じように動作する。したがって、この実施形態では、デバッグ制御回路１６およびプロセッサ１２に対して上述においてしたのと同じ説明がデバッグ制御回路１８およびプロセッサ１４にも適用される。デバッグ制御回路１８のデバッグ制御レジスタのＤＲビットおよびデバッグ命令レジスタのＧＯビットも、デバッグインタフェース２０を介してアサートまたはネゲートされる。幾つかの実施形態では、プロセッサ１２をデバッグモードに入らせる（例えば、プロセッサ１２におけるＤＲ５０をアサートすることによって）第１の要求は、プロセッサ１４をデバッグモードに入らせる（例えば、プロセッサ１４におけるＤＲビットをアサートすることによって）第２の要求と同じ、デバッグモードに入るための要求と考えられる。しかしながら、代替の実施形態では、複数のプロセッサが含まれるため、複数のプロセッサに対するそうした要求は、複数の個別の要求と考えられてもよい。プロセッサ１２と同様、プロセッサ１４は、デバッグモードへの出入りを行う準備が済み（プロセッサ１４の同期化ＤＲまたは同期化ＧＯ信号のアサートによって示される）、プロセッサ１２からアサートされた同期化デバッグ開始要求２７または同期化デバッグ終了命令２８がそれぞれ受信されるまで、デ

バッグモードに出入りしない。一実施形態では、プロセッサ１２，１４の両方は、ＰＣＬＫ５４のような同じプロセッサクロックにしたがって動作する。ここで、プロセッサ１２，１４の各々のデバッグ回路の非同期部分は、ＴＣＬＫ５６のような同じテストクロックにしたがって動作する。しかしながら、代替の実施形態では、第３のクロックがプロセッサ１４に対して用いられ、プロセッサ１４は自身のプロセッサクロック（プロセッサ１２によって用いられるＰＣＬＫ５４と同じ場合も異なる場合もある）にしたがって動作することができる。

【００２７】

一実施形態では、各プロセッサの対応するデバッグ回路制御のＤＲまたはＧＯビットはデバッグインタフェース２０によって同時に設定またはクリアされることが可能であるが、プロセッサ１２の同期化回路４８からの同期化された出力およびプロセッサ１４の同期化回路からの同期化された出力は同時に与えられない。プロセッサ１２，１４の同期化回路は同一であった場合にも、様々な原因に基づいて、同期化された出力を与えるように入力を同期化するために、１つサイクル以上が取られることがある。デバッグモードに入るまたはデバッグモードから出るために各プロセッサが単にそれぞれの同期化回路の出力に依存し、デバッグモードに入るまたはデバッグモードから出る準備が済んだか否かに関するインジケータを他のプロセッサから受信しない場合、同期化時間の差のために、２つのプロセッサが１サイクル以上離れてデバッグモードに入るまたはデバッグモードから出ることがあり、したがって、ロックステップが失われる。したがって、ロックステップデバッグインタフェース３０を含むことおよびシステムにおける他のロックステッププロセッサは、デバッグモードに入るまたはデバッグモードから出る準備が済むまで、この信号を用いてデバッグモードの開始または終了を遅延し、上述に説明したように、ロックステップの損失を防ぐことが可能である。代替の実施形態では、異なるタイプの同期化回路は各プロセッサに用いられることにおいて、上述したようなロックステップデバッグインタフェースの使用もロックステップの損失を防ぐことに役立つ可能性がある。

【００２８】

また、一実施形態では、各プロセッサのデバッグ制御回路に対応するロックステップモードイネーブルインジケータによって示されるように、プロセッサ１２，１４はロックステップモードで動作する時のみ、ロックステップデバッグインタフェース３０が用いられる、あるいはデバッグモードの開始または終了が遅延される。なお、幾つかの実施形態では、ロックステップモードで動作する時、プロセッサ１２，１４が、その２つのプロセッサとの間に遅延を有するように動作し、同じクロックサイクルで動作することはできない。さもなければ、ロックステップモードがイネーブルしない時、対応する制御回路がそれの対応する同期化回路から同期化されたＤＲまたはＧＯ命令を受信すると、各プロセッサがデバッグモードに入るまたはデバッグモードから出ることができる。すなわち、プロセッサ１２，１４はロックステップモードで動作しないとき、独立モードで動作する、ここで、プロセッサ１２，１４は別の独立プロセッサとして機能する。

【００２９】

システム１０の動作について図７～１２のタイミング図を参照して以下に詳しく説明する。しかしながら、同じ説明をシステム内の任意のプロセッサに適用することが可能である。タイミング図の各々において、ＰＣＬＫ５４およびＴＣＬＫ５６は第１の２つの信号として与えられる（また、プログラムカウンタ１５およびプログラム１７の図７～１２の各々に用いられたプログラムカウンタ値は１６進数式であることを注意されたい）。

【００３０】

図７には、ロックステップがディザブルされた状態でデバッグモードに入るタイミングの例を示す。したがって、図７の最後の信号、ロックステップモード、はネゲートされる。この例において、デバッグモードに入ることはＤＲ７０のアサートによって要求される（ここで、プロセッサ１４のＤＲビットも同時にアサートされることが可能である）。矢印８１で示すように、ＴＣＬＫ５６の立ち上がり縁に応答して、ＤＲ７０のアサートはサイクル１の間に起きる。この点において、ＤＲ７０はＰＣＬＫ５４に対して同期されてい

ない。矢印８２で示すように、ＤＲ７０はＰＣＬＫ５４（同期化回路４８によって）さらに同期化され、サイクル３で同期化ＤＲ５１を生成する。矢印８３で示すように、同期化ＤＲ５１のアサートに応答して、制御回路４３が後のサイクル３で同期化デバッグ開始要求２５（デバッグ制御１６の出力）をアサートする。また、矢印８４で示すように、同期化ＤＲ５１のアサートに応答して、プロセッサ１２がサイクル４でデバッグモードに入る（デバッグモード信号が論理レベル１のとき、プロセッサ１２はデバッグモードであり、論理レベルがゼロのとき、プロセッサはデバッグモードではない）。矢印８５で示すように、上述のように、ＰＣＬＫ５４とＴＣＬＫ５６との間の関係は固定されないので、同期化されたバージョンのＤＲ（同期化ＤＲ）はプロセッサ１２，１４で異なることが可能である。したがって、プロセッサ１４のＤＲビットはＤＲ７０と同時に設定できるが、プロセッサ１４の同期化されたＤＲはサイクル４まで、プロセッサ１４の同期化回路によって出力されず、該サイクル４は、プロセッサ１２で同期化ＤＲ５１が与えられるときより１つ後のサイクルである。後のサイクル４では、矢印８６で示すように、プロセッサ１４の制御回路が同期化デバッグ開始要求２７（デバッグ制御１８の出力）をアサートする。また、矢印８７で示すように、サイクル４における同期化ＤＲのアサートに応答して、プロセッサ１４がサイクル５でデバッグモードに入る。したがって、プロセッサ１４がプロセッサ１２より１つ後のサイクルでデバッグモードに入る。しかしながら、ロックステップモードがイネーブルされていないので、デバッグ制御１６（同期化デバッグ開始要求２７）の入力におけるインタフェース信号およびデバッグ制御１８（同期化デバッグ開始要求２５）の入力におけるインタフェース信号が無視され、したがって、プロセッサが無条件でデバッグモードに入る。したがって、デバッグモードにおいて、各プロセッサのプログラムカウンタが異なる値になり（サイクル６および７に認識されるように）、このときプロセッサ１２，１４は同期されていない。

【００３１】

図８には、ロックステップモードがイネーブルされた状態でデバッグモードに入る時間の例を示す。したがって、図８における最後の信号、ロックステップモード、がアサートされる。この例において、デバッグモードに入ることは、ＤＲ７０のアサートによって要求される（ここで、プロセッサ１４のＤＲビットも同時にアサートされることが可能である）。矢印９１で示すように、ＴＣＬＫ５６の立ち上がり縁に応答して、ＤＲ７０のアサートはサイクル１の間に起きる。この点では、ＤＲ７０はＰＣＬＫ５４に対して同期されていない。矢印９２で示すように、ＤＲ７０はＰＣＬＫ５４（同期化回路４８によって）にさらに同期化され、サイクル３で同期化ＤＲ５１を生成する。矢印９３で示すように、同期化ＤＲ５１のアサートに応答して、制御回路４３が後のサイクル３で同期化デバッグ開始要求２５（デバッグ制御１６の出力）をアサートする。矢印９４で示すように、この同期化デバッグ開始要求２５のアサートがデバッグ制御１８の入力を次に伝達する。上述で説明したように、ＰＣＬＫ５４とＴＣＬＫ５６との間の関係が固定されないので、同期化されたバージョンのＤＲ（同期化されたＤＲ）はプロセッサ１２，１４で異なることが可能である。したがって、矢印９５で示すように、プロセッサ１４のＤＲビットはＤＲ７０に同時に設定できるが、プロセッサ１４の同期化されたＤＲはサイクル４までプロセッサ１４の同期化回路によって出力できず、該サイクル４は、同期化ＤＲ５１はプロセッサ１２に与えられた時より１つ後のサイクルである。矢印９６によって示すように、サイクル４の後、プロセッサ１４の制御回路が同期化デバッグ開始要求２７（デバッグ制御１８の出力で）をアサートする。矢印９７で示すように、この同期化デバッグ開始要求２７のアサートがデバッグ制御１６の出力を伝達する。

【００３２】

同期化ＤＲ５１のアサートに応答してプロセッサ１２がデバッグモードに入ることを示す図７と異なって、矢印９８で示すように、同期化ＤＲ５１がアサートされ、同期化デバッグ開始要求２７のアサートが受信されるまでプロセッサ１２がデバッグモードの開始を遅延する。したがって、図７のようにサイクル４でデバッグモードに入るよりプロセッサ１２がサイクル５までデバッグモードの開始を遅延する。同様に、矢印９９で示すように

、プロセッサ１４の同期化されたＤＲがアサートされ、同期化デバッグ開始要求２５のアサートが受信された時、プロセッサ１４がデバッグモードに入る。したがって、プロセッサ１４もサイクル５でデバッグモードに入る。よって、両方のプロセッサは同期しており、ロックステップを損失しない。したがって、同期化デバッグ開始要求信号の使用によって、ロックステップが維持できる。図７の例に比較すると、サイクル５～７に示すように、プロセッサ１２，１４のプログラムカウンタ値はデバッグモードの間で同じ値である。

【００３３】

図９が、ロックステップ動作がイネーブルされた状態でデバッグモードに入るタイミングの別の例を示す。したがって、図９の最後の信号、ロックステップモード、がアサートされる。この例において、デバッグモードの開始はＤＲ７０のアサートによって要求される（ここで、プロセッサ１４のＤＲビットも同時にアサートされることが可能である）。矢印１０１で示すように、ＴＣＬＫ５６の立ち上がり縁に応答して、ＤＲ７０はサイクル１でアサートされる。この点で、ＤＲ７０はＰＣＬＫ５４に対して同期されていない。矢印１０５で示すように、ＤＲ７０はＰＣＬＫ５４（同期化回路４８によって）にさらに同期化され、サイクル３で同期化ＤＲ５１を生成する。矢印１０６で示すように、同期化ＤＲ５１のアサートに応答して、制御回路４３が後のサイクル３で同期化デバッグ開始要求２５（デバッグ制御１６から出力）をアサートする。矢印１０３で示すように、同期化デバッグ開始要求２５のアサートが、デバッグ制御１８の入力をさらに伝達する。上述で説明したように、ＰＣＬＫ５４とＴＣＬＫ５６との間の関係は固定されないので、同期化されたバージョンのＤＲ（同期化されたＤＲ）はプロセッサ１２，１４で異なることが可能である。しかしながら、図９において、矢印１０２で示すように、プロセッサ１４の同期化されたＤＲもサイクル４でプロセッサ１４の同期化回路によって出力される。後のサイクル３で、矢印１０９で示すように、プロセッサ１４の制御回路が同期化デバッグ開始要求２７（デバッグ制御１８の出力で）をアサートする。矢印１０４で示すように、この同期化デバッグ開始要求２７のアサートが、デバッグ制御１６の入力をさらに伝達する。

【００３４】

また図９を参照して、矢印１０７で示すように、同期化デバッグ開始要求２５がデバッグ制御１６の出力でアサートされ、アサートおよび同期化デバッグ開始要求２７はデバッグ制御１６によって受信される時、プロセッサ１２がデバッグモードに入る。同様に、矢印１０８で示すように、同期化デバッグ開始要求２７がデバッグ制御１８の出力でアサートされ、アサートおよび同期化デバッグ開始要求２５はデバッグ制御１８によって受信される時、プロセッサ１４がデバッグモードに入る。したがって、この例において、デバッグモードに入るために別のプロセッサからアサートされた同期化デバッグ開始要求を受信することに加えて同期化されたＤＲのアサートを待機するより、別のプロセッサからアサートされた同期化デバッグ開始要求を受信することに加えてプロセッサの各々がプロセッサの各々の同期化デバッグ開始要求を受信するまで各プロセッサが待機する。したがって、この例において、ロックステップを損失することなく、両方のプロセッサは同期状態で維持される。したがって、同期化デバッグ開始要求信号の使用によって、ロックステップは維持できる。

【００３５】

図１０が、ロックステップ動作状態でデバッグモードから出るタイミングの例を示す。したがって、図１０の最後の信号、ロックステップモード、がネゲートされる。この例において、デバッグモードの終了はＧＯ６２のアサートによって要求される（ここで、プロセッサ１４のＧＯビットも同時にアサートされることが可能である）。矢印１１１で示すように、ＴＣＬＫ５６の立ち上がり縁に応答して、ＧＯ６２がサイクル１でアサートされる。この点において、ＧＯ６２はＰＣＬＫ５４に対して同期されていない。矢印１１２で示すように、ＧＯ６２はＰＣＬＫ５４（同期化回路４８によって）にさらに同期化され、サイクル３で同期化ＧＯ５３を生成する。矢印１１３で示すように、同期化ＧＯ５３のアサートに応答して、制御回路４３が後のサイクル３で同期化デバッグ終了命令２６（デバッグ制御１６の出力）をアサートする。また、矢印１１４で示すように、同期化ＧＯ５３

のアサートに応答して、プロセッサ１２がサイクル４でデバッグモードから出る。上述で説明したように、ＰＣＬＫ５４とＴＣＬＫ５６との間の関係は固定されないので、同期化されたバージョンのＧＯ（同期化ＧＯ）がプロセッサ１２，１４と異なることが可能である。したがって、プロセッサ１４のＧＯビットはＧＯ６２に同時に設定されることが可能であるが、矢印１１５で示すように、プロセッサ１４の同期化ＧＯはサイクル４までプロセッサ１４の同期化回路によって出力できなく、該サイクル４は、同期化ＧＯ５３がプロセッサ１２に与えられた時より１つ後のサイクルである。後のサイクル４において、矢印１１６で示すように、プロセッサ１４の制御回路が同期化デバッグ終了命令２８（デバッグ制御１８の出力で）をアサートする。また、矢印１１７で示すように、サイクル４における同期化ＧＯのアサートに応答して、プロセッサ１４がサイクル５でデバッグモードに入る。したがって、この例において、プロセッサ１４がプロセッサ１２の１つ後のサイクルでデバッグモードから出る。しかしながら、ロックステップモードがイネーブルされないので、デバッグ制御１６の入力におけるインタフェース信号（同期化デバッグ終了命令２８）およびデバッグ制御１８の入力におけるインタフェース信号（同期化デバッグ終了命令２６）が無視され、したがって、プロセッサがデバッグモードから出ることに条件を付けない。したがって、各プロセッサのプログラムカウンタは異なる値になり（サイクル６および７から参照するように）、よって、このときプロセッサ１２，１４は同期されていない。

【００３６】

図１１が、ロックステップモードがイネーブルされた状態でデバッグモードに入るまたはデバッグモードから出るタイミングの例を示す。したがって、図１１の最後の信号、ロックステップ、がアサートされる。この例において、デバッグモードから出ることはＧＯ６２のアサートによって要求される（ここで、プロセッサ１４のＧＯビットも同時にアサートされることが可能である）。矢印１２１で示すように、ＴＣＬＫ５６の立ち上がり縁に応答して、ＧＯ６２がサイクル１でアサートされる。この点において、ＧＯ６２はＰＣＬＫ５４に対して同期されていない。矢印１２２で示すように、ＧＯ６２はＰＣＬＫ５４（同期化回路４８によって）にさらに同期化され、サイクル３で同期化ＧＯ５３を生成する。矢印１２３で示すように、同期化ＧＯ５３のアサートに応答して、制御回路４３が同期化デバッグ終了命令２６（デバッグ制御１６の出力で）をアサートする。矢印１２４で示すように、この同期化デバッグ終了命令２６のアサートがデバッグ制御１８の入力を伝達する。上述で説明したように、ＰＣＬＫ５４とＴＣＬＫ５６との間の関係が固定されないので、同期化されたバージョンのＧＯ（同期化ＧＯ）がプロセッサ１２，１４で異なることが可能である。したがって、矢印１２５で示すように、プロセッサ１４のＧＯビットはＧＯ６２に同時に設定されることが可能であるが、プロセッサ１４の同期化ＧＯはサイクル４までプロセッサ１４の同期化回路によって出力されない、該サイクル４は、同期化ＧＯ５３がプロセッサ１２に与えられた時より１つ後のサイクルである。後のサイクル４で、矢印１２６で示すように、プロセッサ１４の制御回路が同期化デバッグ終了命令２８（デバッグ制御１８の出力で）をアサートする。矢印１２７で示すように、この同期化デバッグ終了命令２８のアサートがデバッグ制御１６の入力を伝達する。

【００３７】

同期化ＧＯ５３のアサートに応答してプロセッサ１２がデバッグモードから出ることを示す図１０と異なって、矢印１２８で示すように、同期化ＧＯ５３がアサートされ、同期化デバッグ終了命令２８がアサートされることが受信されるまでプロセッサ１２がデバッグモードの終了を遅延する。したがって、図１０で行ったように、プロセッサ１２が、サイクル４よりサイクル５までデバッグモードの終了を遅延する。同様に、矢印１２９で示すように、プロセッサ１４の同期化ＧＯがアサートされ、同期化デバッグ終了命令２６のアサートが受信されるまでプロセッサ１４がデバッグモードから出る。したがって、プロセッサ１４もサイクル５でデバッグモードから出る。したがって、ロックステップを損失しなく、両方のプロセッサは同期化で維持する。したがって、同期化デバッグ終了命令信号の使用を介して、ロックステップが維持される。

【００３８】
図１２が、ロックステップ動作をイネーブルされた状態でデバッグモードから出るタイミングの例を示す。したがって、図１２の最後の信号、ロックステップモード、がアサートされる。この例において、デバッグモードの終了は、ＧＯ６２のアサートによって要求される（ここで、プロセッサ１４のＧＯビットも同時にアサートされることが可能である）。矢印１３１で示すように、ＴＣＬＫ５６の立ち上がり縁に応答して、ＧＯ６２がサイクル１でアサートされる。この点において、ＧＯ６２はＰＣＬＫ５４に対して同期されていない。矢印１３３で示すように、ＧＯ６２はＰＣＬＫ５４（同期化回路４８によって）にさらに同期化され、同期化ＧＯ５３をサイクル３で生成する。矢印１３４で示すように、同期化ＧＯ５３のアサートに応答して、制御回路４３が後のサイクル３で同期化デバッグ終了命令２６（デバッグ制御１６の出力）をアサートする。上述で説明したように、ＰＣＬＫ５４とＴＣＬＫ５６との間の関係は固定されないので、同期化されたバージョンのＧＯ（同期化ＧＯ）はプロセッサ１２，１４で異なることが可能である。しかしながら、図１２において、矢印１３２で示すように、プロセッサ１４の同期化ＧＯもサイクル４プロセッサ１４の同期化回路によって出力される。後のサイクル３において、矢印１３９で示すように、プロセッサ１４の制御回路が同期化されたデバッグ開始命令２８（デバッグ制御１８の出力で）をアサートする。矢印１３６で示すように、この同期化デバッグ終了命令２８のアサートがデバッグ制御１６の入力を伝達する。

【００３９】
また図１２を参照して、矢印１３７で示すように、同期化デバッグ終了命令２６がデバッグ制御１６の出力でアサートされ、アサートされた同期化デバッグ終了命令２８はデバッグ制御１６によって受信される時、プロセッサ１２がデバッグモードから出る。同様に、矢印１３８で示すように、同期化デバッグ終了命令２８がデバッグ制御１８の出力でアサートされ、アサートされた同期化デバッグ終了命令２６がデバッグ制御１８によって受信される時、プロセッサ１４がデバッグモードから出る。したがって、この例において、デバッグモードから出るために別のプロセッサからアサートされた同期化デバッグ終了命令を受信することに加えて同期化ＧＯを待機するより、別のプロセッサからアサートされた同期化デバッグ終了命令を受信することに加えてプロセッサの各々がプロセッサの各々の同期化デバッグ終了命令をアサートするまでプロセッサの各々が待機する。したがって、この例において、両方のプロセッサが、ロックステップを損失しなくロックステップで維持する。したがって、同期化デバッグ終了命令信号の使用を介して、ロックステップが維持できる。

【００４０】
したがって、クロスシグナリングの使用によって、ロックステップモードで動作するシステム内のプロセッサがロックステップを損失せず、デバッグモードに入るまたはデバッグモードから出ることが保証されることが理解される。このようにして、デバッグを開始または終了しても、システムにおける複数のプロセッサが同期状態を維持する。

【００４１】
例えば、一実施形態では、システム１０の図示された素子は単一集積回路または同じデバイス以内に配置される。あるいは、システム１０は、任意の数の別の集積回路または別の互いに相互接続されたデバイスを含むことが可能である。例えば、他のモジュール２２がある場合、そのモジュールは、プロセッサ１２，１４と同じ集積回路または別の集積回路に配置されてもよく、またはシステム１０の他の素子から別々に離れた別の周辺装置またはスレーブ装置に配置されてもよい。

【００４２】
以下に本発明の様々の実施形態を説明する。
項目１は、第１のプロセッサと、第２のプロセッサと、第１のプロセッサに接続された第１のクロックと、第２のプロセッサに接続された第２のクロックと、第１のプロセッサおよび第２のプロセッサに接続された第３のクロックと、を備えるシステムである。第１のプロセッサは、第３のクロックを受信するように接続されたデバッグ回路と、第１のク

ロックを受信するように接続された同期化回路であって、デバッグモードに入るための第１の要求を受信し、第１の同期化デバッグ開始要求信号を供給する、同期化回路と、第２のプロセッサから第２の同期化デバッグ開始要求信号を受信するための入力とを備える。第１の同期化デバッグ開始要求信号は第１のクロックに対して同期化されており、第１のプロセッサは、第１の同期化デバッグ開始要求信号および第２の同期化デバッグ開始要求信号の両方がアサートされるまで、デバッグモードに入ることを待機する。項目２は、項目１のシステムを含み、第３のクロックは第１のクロックに対して同期されていない。項目３は、項目２のシステムを含み、第２のクロックは第２のクロックに対して同期されていない。項目４は、項目１のシステムを含み、デバッグ回路はデバッグ要求レジスタビットを備え、デバッグ要求レジスタビットのアサートに応答して、デバッグモードに入るための第１の要求は、同期化回路に対し供給される。項目５は、項目１のシステムを含み、第１のプロセッサは第１のプログラムカウンタを備え、第２のプロセッサは第２のプログラムカウンタを備え、第１のプロセッサ及び第２のプロセッサがデバッグモードにあるとき、同時に第１のプロセッサは第１のプログラムカウンタをインクリメントし、第２のプロセッサは第２のプログラムカウンタをインクリメントする。項目６は、項目１のシステムを含み、第１のプロセッサおよび第２のプロセッサはデバッグモードに入り、該デバッグモード中、ロックステップ方式により同じ命令を実行する。項目７は、項目１のシステムを含み、第１のプロセッサおよび第２のプロセッサはデバッグモードに入り、該デバッグモード中、同じ命令を実行し、第１のプロセッサによる前記同じ命令の実行は、第２のプロセッサによる前記同じ命令の実行と比較して、第１のクロックにおける１０サイクル以下だけスキューされる。項目８は、項目１のシステムを含み、同期化回路はデバッグモードから出るための要求をデバッグ回路から受信し、第１の同期化されたデバッグ終了コマンド信号を供給し、第１の同期化されたデバッグ終了コマンド信号は第１のクロックに対して同期化されており、第１のプロセッサはさらに、第１の同期化デバッグ開始要求信号を第１のプロセッサから第２のプロセッサに転送するための第１の出力と、第１の同期化デバッグ終了命令信号を第１のプロセッサから第２のプロセッサに転送するための第２の出力と、第２の同期化デバッグ終了命令信号を第２のプロセッサから受信するための入力とを備え、第１のプロセッサは、第１の同期化されたデバッグ終了コマンド信号および第２の同期化されたデバッグ終了コマンド信号の両方がアサートされるまで、デバッグモードを出ることを待機する。項目９は、項目１のシステムを含み、第２のプロセッサは、第３のクロックを受信するように接続されたデバッグ回路と、第２のクロックを受信するように接続された同期化回路であって、デバッグモードに入るための第２の要求を受信し、第２の同期化デバッグ開始要求信号を第１のプロセッサに供給する、同期化回路と、第１のプロセッサから第１の同期化デバッグ開始要求信号を受信するための入力とを備える。項目１０は、項目９のシステムを含み、第２のプロセッサは、第１の同期化デバッグ開始要求信号および第２の同期化デバッグ開始要求信号の両方がアサートされるまで、デバッグモードに入ることを待機する。

【００４３】

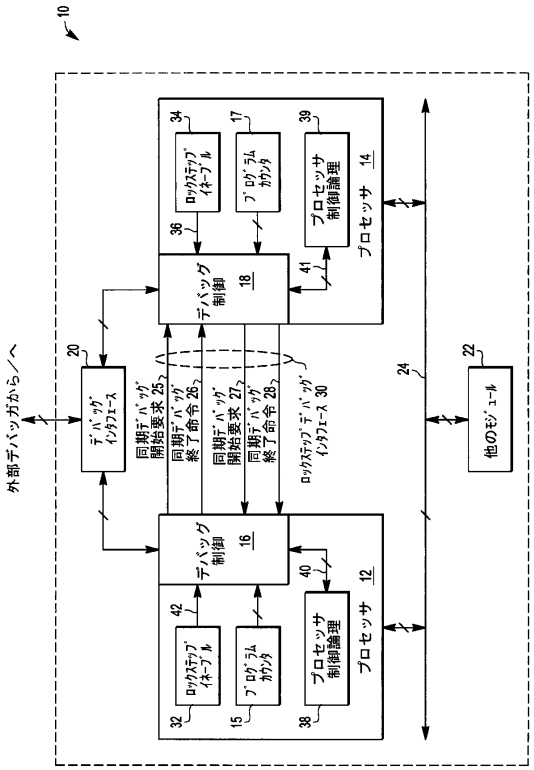
項目１１は、第１のクロックを用いてデバッグ回路の第１の部分にクロックを供給する工程と、第２のクロックを用いてデバッグ回路の第２の部分にクロックを供給する工程であって、第１のクロックは第２のクロックに対して同期されていない工程と、デバッグモードに入るための第１の要求をデバッグ回路の第１の部分から同期化回路に転送する工程であって、デバッグモードに入るための第１の要求は第２のクロックに対して同期されていない工程と、第１のプロセッサが、第１の同期化デバッグ開始要求を生成するために、デバッグモードに入るための第１の要求を第２のクロックと同期させる工程と、第１のプロセッサが、第２のプロセッサから受信されている第２の同期化デバッグ開始要求を監視する工程と、第１のプロセッサが、第１の同期化デバッグ開始要求および第２の同期化デバッグ開始要求の両方がアサートされるまで、デバッグモードに入ることを待機する工程と、を含む方法である。項目１２は、項目１１の方法において、第１の同期化デバッグ開始要求を第１のプロセッサから第２のプロセッサに転送する工程をさらに含む。項目１３

は、項目１１の方法において、第１のプロセッサおよび第２のプロセッサに対するロックステップ動作をイネーブルする工程をさらに含み、ロックステップ動作は、デバッグモード中、第１のプロセッサおよび第２のプロセッサにおいて同じ命令を実行することを含む。項目１４は、項目１１の方法において、デバッグモードから出るための要求をデバッグ回路の第１の部分から同期化回路に転送する工程であって、デバッグモードから出るための要求は第２のクロックに対して同期されていない工程と、第１のプロセッサが、第１の同期化デバッグ終了命令を生成するために、デバッグモードから出るための要求を第２のクロックと同期させる工程と、第１のプロセッサが、第２のプロセッサから受信されている第２の同期化デバッグ終了命令を監視する工程と、第１のプロセッサが、第１の同期化デバッグ終了命令および第２の同期化デバッグ終了命令の両方がアサートされるまで、デバッグモードから出ることを待機する工程と、をさらに含む。項目１５は、項目１１の方法において、第２のプロセッサが、第２の同期化デバッグ開始要求を生成するために、デバッグモードに入るための第２の要求を第３のクロックと同期させる工程をさらに含み、第１のクロックは第３のクロックに対して同期されていない。項目１６は、項目１５の方法において、第２のクロックおよび第３のクロックは同期されていない。項目１７は、項目１５の方法において、デバッグモードに入る前に、第２のプロセッサが、第１のプロセッサから受信されている第１の同期化デバッグ開始要求を監視する工程と、第１の同期化デバッグ開始要求および第２の同期化デバッグ開始要求の両方がアサートされた後、第２のプロセッサが、デバッグモードにおいて１つ以上の命令を実行する工程と、をさらに含む。項目１８は、項目１１の方法において、第１のプロセッサおよび第２のプロセッサは同じ集積回路上に形成される。

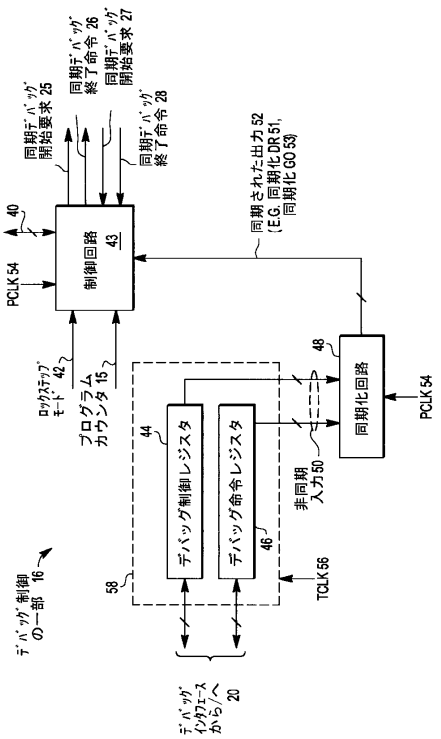
【００４４】

項目１９は、第１のプロセッサおよび第２のプロセッサにテストクロックを入力する工程と、第１のプロセッサに第１のプロセッサクロックを入力する工程と、第２のプロセッサに第２のプロセッサクロックを入力する工程と、を備え、テストクロックは第１のプロセッサクロックに対して同期されておらず、テストクロックは第２のプロセッサクロックに対して同期されておらず、さらに、第１のプロセッサが、デバッグモードに入るための第１の要求を受信する工程と、第２のプロセッサが、デバッグモードに入るために前記要求を受信する工程であって、デバッグモードに入るための第１の要求およびデバッグモードに入るための第２の要求はテストクロックに対して同期されている工程と、第１のプロセッサが、第１の同期化デバッグ開始要求を生成するために、デバッグモードに入るための第１の要求を第１のプロセッサクロックに対して同期させる工程と、第２のプロセッサが、第２の同期化デバッグ開始要求を生成するために、デバッグモードに入るための前記要求を前記プロセッサクロックに対して同期させる工程と、第１のプロセッサが、第２のプロセッサから第２の同期化デバッグ開始要求を受信し、第２の同期化デバッグ開始要求を用いて、デバッグモードに入る時を決定する工程と、第２のプロセッサが、第１のプロセッサから第１の同期化デバッグ開始要求を受信し、第１の同期化デバッグ開始要求を用いて、デバッグモードに入る時を決定する工程と、を備える方法である。項目２０は、項目１９の方法において、デバッグモードに入る前に、第１のプロセッサが、第１の同期化デバッグ開始要求及び第２の同期化デバッグ開始要求の両方がアサートされるまで待機し、デバッグモードに入る前に、第２のプロセッサが、第２の同期化デバッグ開始要求及び第１の同期化デバッグ開始要求の両方がアサートされるまで待機する。

【図１】



【図２】



【図３】

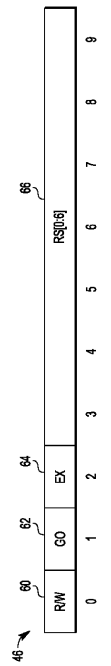


FIG. 3

【図４】

ビット	名前	説明
0	R/W	読取/書込命令ビット このR/Wビットによって、データ転送の向きが指定される。 0－命令に関連するデータをRS[0:6]によって指定されるレジスタに書き込む。 1－RS[0:6]によって指定されるレジスタに含まれるデータを読み取る。
1	GO	GO命令ビット 0－非活性（作動せず） 1－IRにおいて実行可能な命令 GOビットがセットされる場合、チップはレジスタにある命令を実行する。 命令を実行するために、プロセッサはデバッグモードに留まり、命令を実行する。そして、 EXビットがクリアされる場合、命令の実行の直後にデバッグモードにプロセッサは EXビットがセットされている場合、通常動作を続け、他のデバッグ要求ソースはアサートされない。
2	EX	終了命令ビット 0－デバッグモードに留まる 1－デバッグモードを抜ける EXビットがセットされている場合、プロセッサはデバッグモードを抜け、別のデバッグ要求が 発生するまで通常動作を続ける。
3.9	RS	レジスタ選択 レジスタ選択ビットは、いずれのレジスタが読取（書込）動作用のソースであるかを決める。

【図５】

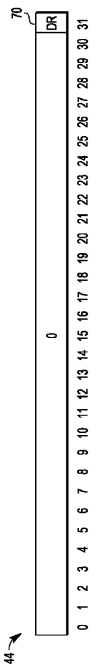
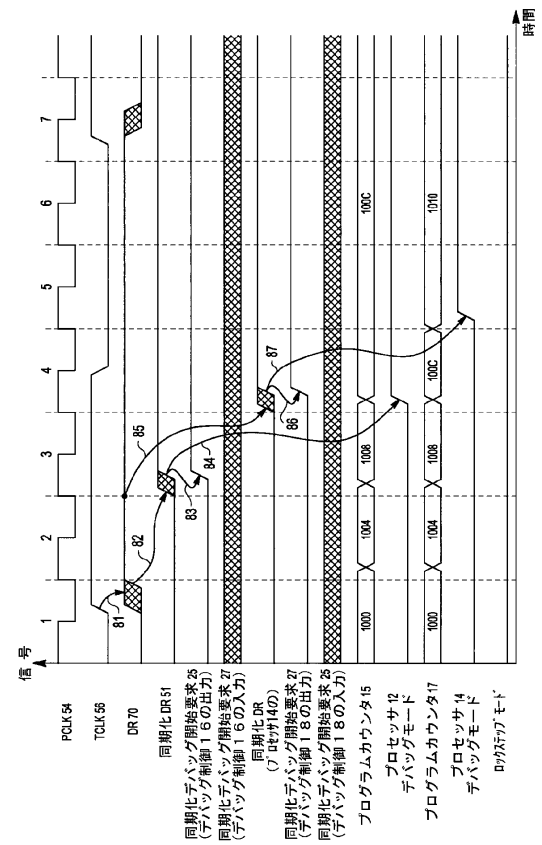


FIG. 5

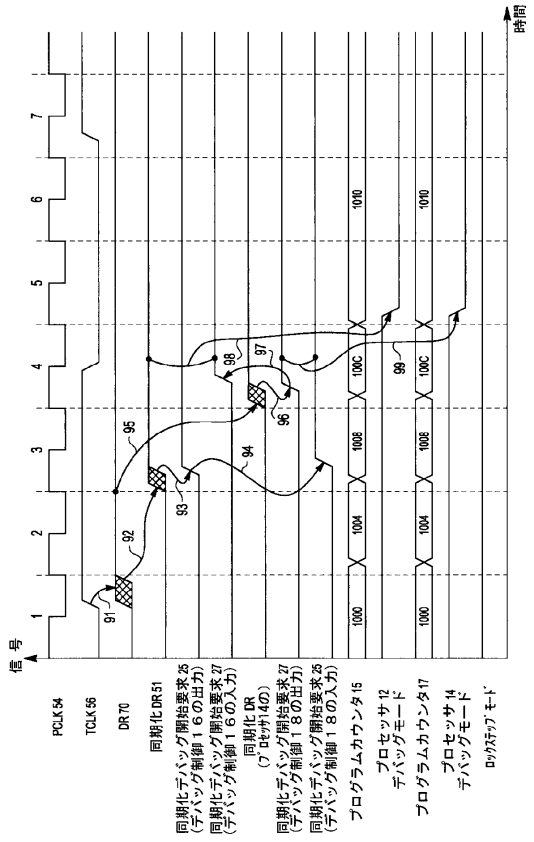
【図６】

ビット	名前	説明
31	DR	デバッグ要求制御ビット このビットは、プロセッサがデバッグモードに入ることを無条件に要求するために用いられる。 0－デバッグモード要求なし 1－無条件のデバッグモード要求 DRビットがセットされるとき、プロセッサは次の命令境界においてデバッグモードに入る

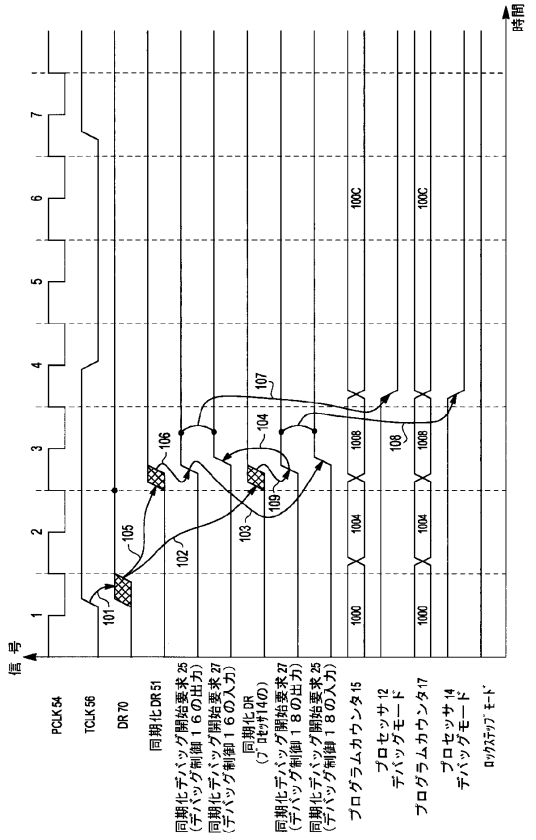
【図７】



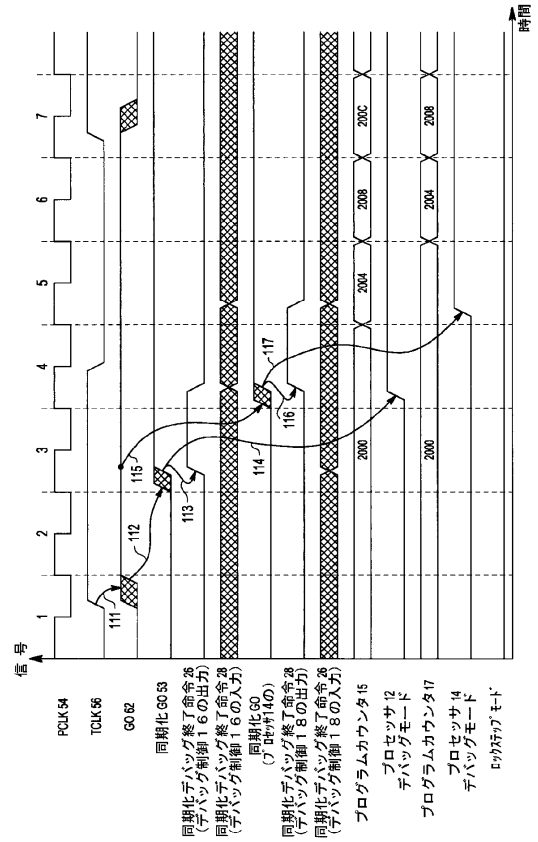
【図８】



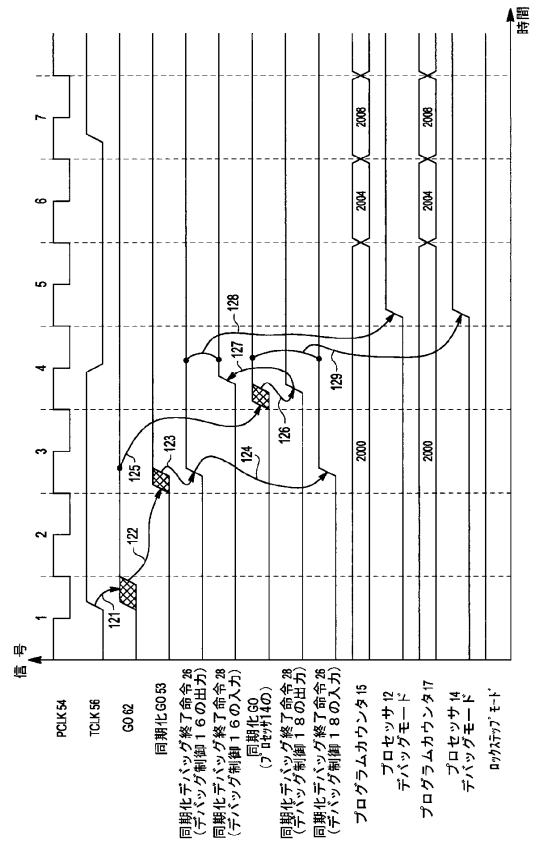
【図９】



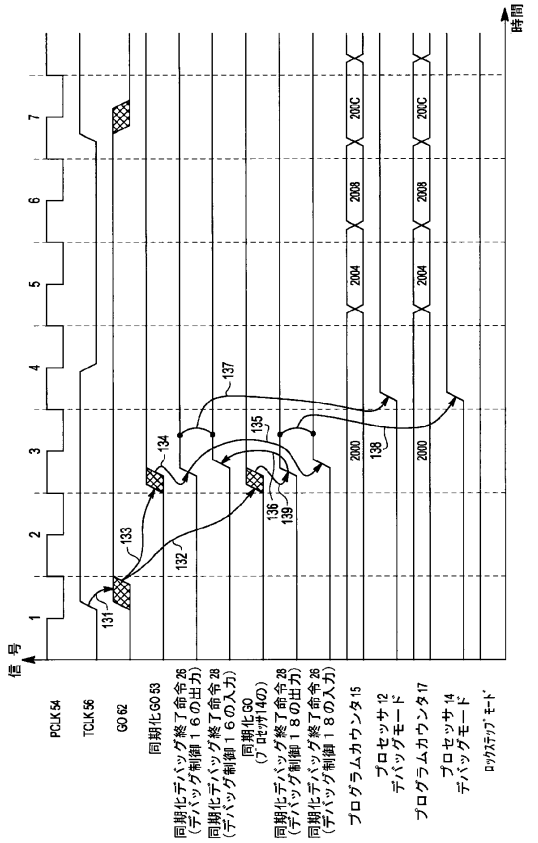
【図１０】



【図１１】



【図１２】



フロントページの続き

(72)発明者　グムルジャ、ジミー
アメリカ合衆国　７８７４６　テキサス州　オースティン　カノネロ　ドライブ　１８０１

審査官　多賀　実

(56)参考文献　特表２００６－５１２６３４（ＪＰ，Ａ）
特表２００８－５１８２９９（ＪＰ，Ａ）
特開２００６－０７９１４２（ＪＰ，Ａ）
特開２００６－１７８６１７（ＪＰ，Ａ）
特開平７－２６１８１４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１１／２２
Ｇ０６Ｆ　１１／２８
Ｇ０６Ｆ　　９／４８
Ｇ０６Ｆ　１１／１６－１１／２０
Ｇ０６Ｆ　１５／７８