(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6317365号
(P6317365)
(45)発行日　平成30年4月25日(2018.4.25)　(24)登録日　平成30年4月6日(2018.4.6)

(51)Int.Cl.　　F I
G06F 15/173 (2006.01)　　G06F 15/173 660D
G06F 9/52 (2006.01)　　G06F 15/173 683C
　　G06F 9/46 475A

請求項の数 15　(全 28 頁)

(21)出願番号　特願2015-546462(P2015-546462)
(86)(22)出願日　平成25年10月10日(2013.10.10)
(65)公表番号　特表2016-501414(P2016-501414A)
(43)公表日　平成28年1月18日(2016.1.18)
(86)国際出願番号　PCT/US2013/064366
(87)国際公開番号　WO2014/088698
(87)国際公開日　平成26年6月12日(2014.6.12)
審査請求日　平成28年9月27日(2016.9.27)
(31)優先権主張番号　61/734,190
(32)優先日　平成24年12月6日(2012.12.6)
(33)優先権主張国　米国(US)

(73)特許権者　505000815
コーヒレント・ロジックス・インコーポレーテッド
アメリカ合衆国・７８７４６・テキサス州・オースティン・サウス　キャピタル　オブ　テキサス　ハイウェイ・１１２０・ビルディング　３，スイート　３１０
(74)代理人　100064621
弁理士　山川　政樹
(74)代理人　100098394
弁理士　山川　茂樹
(72)発明者　ドブス，カール・エス
アメリカ合衆国・７８７３７・テキサス州・オースティン・ジム　ブリッジャー　ドライブ・１２０１０

最終頁に続く

(54)【発明の名称】同期命令を含む処理システム

(57)【特許請求の範囲】
【請求項１】
複数のプロセッサであって、前記複数のプロセッサの各プロセッサは、複数のプロセッサポートおよび１つの同期アダプタを備え、前記同期アダプタが複数のアダプタポートを備える、プロセッサと；
複数のコントローラであって、前記複数のコントローラの各コントローラは、複数のコントローラポートを備え、前記複数のコントローラポートの各コントローラポートは、前記複数のプロセッサのうちの隣接するプロセッサのアダプタポートに接続している、コントローラと；
を備えるシステムであって、
前記複数のプロセッサの各プロセッサは、
選択的に、１つ以上のアダプタポートを介して同期信号を前記複数のコントローラのそれぞれの１つ以上のコントローラに送信し；
１つ以上のコントローラからの応答に応じてプログラム命令の実行を停止する；
ように設定され、
前記複数のコントローラの各コントローラは、
前記複数のプロセッサのそれぞれの１つ以上のプロセッサから１つ以上の同期信号を受信し；
前記受信した１つ以上の同期信号に応じて前記複数のプロセッサの前記それぞれの１つ以上のプロセッサの各々に応答を送信するように設定される、システム。

【請求項２】
前記複数のコントローラの各コントローラは、１つ以上の設定データビットを受信するように設定された設定ポートを備える、請求項１に記載のシステム。
【請求項３】
前記それぞれの１つ以上のプロセッサの各々に前記応答を送信するために、前記複数のコントローラの各コントローラは、さらに、前記１つ以上の設定データビットに応じて前記それぞれの１つ以上のプロセッサに前記応答を送信するように設定される、請求項２に記載のシステム。
【請求項４】
前記複数のコントローラの各コントローラは、レジスタを備える、請求項１に記載のシステム。
【請求項５】
前記複数のコントローラの各コントローラに対する前記１つ以上の設定ビットは、各コントローラの前記レジスタに格納される、請求項４に記載のシステム。
【請求項６】
マルチプロセッサシステムを動作させるための方法であって、前記マルチプロセッサシステムは、複数のプロセッサおよび複数の同期コントローラを備え、前記複数のプロセッサの各プロセッサは、同期アダプタを備え、
１つ以上のグループのプロセッサの各グループは、前記複数のプロセッサのサブセットを備え、
前記方法は：
前記１つ以上のグループのプロセッサの所与のグループを特定し、前記所与のグループの各プロセッサのソフトウェア内に少なくとも１つの場所を特定して、同期バリアを挿入すること；
前記所与のグループの各プロセッサの各同期アダプタに接続している、前記複数の同期コントローラのうちの１つの同期コントローラを特定すること；
前記特定した同期コントローラに接続している各同期アダプタに向かう方向に応じて、前記特定した同期コントローラに対する設定データを決定し、前記設定データは、前記特定した同期コントローラと、前記所与のグループの選択されたプロセッサの前記同期アダプタとの間で複数の同期信号を伝送することを可能にすること；および
前記所与のグループの各プロセッサに対して、前記ソフトウェア内で前記特定した場所の各々に同期バリアを挿入し、各同期バリアは一連の１つ以上の同期命令を含み、各同期命令は１つ以上の引数を含み、前記１つ以上の引数の各引数は、前記所与のグループの各プロセッサの前記同期アダプタから、前記特定した同期コントローラの１つ以上の方向のうちの所与の１つを指定すること；
前記挿入した同期命令の所与の１つを実行している前記所与のグループの各プロセッサに応答して、前記所与のグループの各プロセッサの実行を停止することと；
前記所与のグループが前記挿入した同期命令の１つ１つを全プロセッサが実行したかどうかという判断に応答して、前記所与のグループの各プロセッサの実行を再開することとを含む、方法。
【請求項７】
前記判断した設定データは、複数の設定データビットを備え、前記複数の設定データビットのそれぞれの組み合わせに対応する各データパターンは、前記特定した同期コントローラに接続しているプロセッサの前記所与のグループの所与のプロセッサの前記同期アダプタへ向かう方向に一致する、請求項６に記載の方法。
【請求項８】
設定データを前記マルチプロセッサシステムにロードすることは、前記複数の設定データビットを前記特定した同期コントローラのレジスタに格納することを含む、請求項７に記載の方法。
【請求項９】

前記所与のグループの各プロセッサの実行を再開することは、前記特定した同期コントローラによって停止信号をデアサートすることを含む、請求項６に記載の方法。
【請求項１０】
前記所与のグループの各プロセッサの実行を停止することは、前記所与のグループの各プロセッサの前記同期アダプタによって同期リクエスト信号をアサートすることを含む、請求項６に記載の方法。
【請求項１１】
同期コントローラであって：
複数の設定ビットを格納するように設定されたレジスタであって、前記複数の設定ビットは、複数のデータパターンを符号化し、前記複数のデータパターンの各データパターンは、接続した方向の複数のサブセットのうちの所与の１つに一致する、レジスタと；
１つ以上の論理回路であって、前記１つ以上の論理回路の各論理回路は：
１つ以上の同期リクエスト信号を受信し；
前記受信した１つ以上の同期リクエストおよび前記複数のデータパターンの対応するデータパターンに応じて、少なくとも１つのプロセッサに対する停止信号を生成し、前記少なくとも１つのプロセッサは、前記接続した方向の複数のサブセットのうちの対応する１つの所与の方向で同期コントローラに接続される
ように設定される、論理回路とを備える、同期コントローラ。
【請求項１２】
１つ以上の入力の各入力は、ラッチを含む、請求項１１に記載の同期コントローラ。
【請求項１３】
前記受信した１つ以上の同期リクエストおよび前記複数のデータパターンの前記対応するデータパターンに応じて前記停止信号を生成するために、各論理回路は、さらに、接続した方向の前記複数のサブセットのうちの対応する１つの方向に対応する１つ以上の停止信号を生成するように設定される、請求項１１に記載の同期コントローラ。
【請求項１４】
前記受信した同期リクエストおよび前記複数のデータパターンの前記対応するデータパターンに応じて前記停止信号を生成するために、各論理回路は、さらに、前記受信した同期信号のうちの選択した１つのアサートに応答して前記停止信号を生成するように設定され、前記受信した同期信号のうちの前記選択した１つは、前記複数のデータパターンの前記対応するデータパターンの方向に対応する、請求項１１に記載の同期コントローラ。
【請求項１５】
前記受信した同期リクエストおよび前記複数のデータパターンの前記対応するデータパターンに応じて前記停止信号を生成するために、各論理回路は、さらに、前記受信した同期信号のうちの残りの同期信号がアサートされていないという判断に応答して前記停止信号を生成するように設定される、請求項１４に記載の同期コントローラ。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、コンピュータシステムおよびデジタル信号プロセッサ（ＤＳＰ）に関し、さらに詳細には、マルチプロセッサシステムに関する。
【背景技術】
【０００２】
本明細書は、電子システム、特に、コンピュータ、デジタル信号プロセッサ（ＤＳＰ）などのデジタル電子システム、およびさらに大きいシステムに組み込まれたこれらのシステムに関する。さらに詳細には、本概念は、デジタル電子システム内の信号ネットワークに関し、特にマルチプロセッサアレイ（ＭＰＡ）内の同期信号ネットワークに関する。ＭＰＡは、処理要素（ＰＥ）のグループと、支持メモリ（ＳＭ）と、ＰＥとメモリとの間で帯域幅の広いデータ通信を支持する第一相互接続ネットワーク（ＰＩＮ）とで設定される。

【０００３】
上記ＰＥは、入力データおよび出力データをバッファリングするためのレジスタ、命令処理ユニット（ＩＰＵ）、およびデータに対して算術関数および論理関数を実施するための論理回路／回路のほか、システムの他の部分と通信するための多数のスイッチおよびポートを有する。ＩＰＵは、メモリから命令を取得し、それを復号化し、適切な制御信号を設定してデータをプロセッサから出し入れして算術関数および論理関数を実施する。
【０００４】
コンピュータのメモリおよびＤＳＰは、最上部に高速メモリがある階層で、階層が一段下がるにつれて低速になるが容量が多くなっていく階層内に組織される。ＭＰＡでは、階層の最上部にある支持メモリが各ＰＥに隣接して設置される。各支持メモリは、命令のみまたはデータのみを保持するように特殊化されてよい。特定のＰＥに対する支持メモリがそのＰＥに専用のものであってもよいし、他のＰＥと共有されてもよい。
【０００５】
ＭＰＡが最初に回路基板上でデジタル集積回路（ＩＣ）のアレイとして作製され、各ＩＣが１つのプロセッサを備え、回路基板がプロセッサどうしを相互接続するデータ通信リンクを提供していた。製造寸法がさらに小さい相補型金属酸化膜半導体（ＣＭＯＳ）トランジスタ回路に基づいた超大規模集積（ＶＬＳＩ）技術が進化し続け、シリコンＩＣチップ１つあたりの論理回路とメモリ回路との密度が大幅に増大した。今日、１つのＩＣチップ上でＭＰＡは、１００個以上のプロセッサおよびその支持メモリおよび相互接続ネットワークを備えて作製されている。これらのＭＰＡチップを回路基板上でさらに相互接続させてさらに大きいシステムを作製できる。
【０００６】
ＭＰＡに適したＰＥは、単にＭＰＡチップ１つあたりのＰＥ数が多いために汎用プロセッサ（ＧＰＰ）よりもエネルギー効率が高いことがあり、余分なエネルギーは余分な廃熱になり、その熱を取り除くにはチップのパッケージ費用および操作費用が加算される。
【先行技術文献】
【特許文献】
【０００７】
【特許文献１】米国特許第７，４１５，５９４号
【特許文献２】米国特許出願第１３／２７４，１３８号
【発明の概要】
【０００８】
マルチプロセッサアレイの様々な実施形態を開示する。広義には、複数のプロセッサおよび複数のコントローラが散在するように一緒に接続している回路および方法が構想される。各プロセッサは、複数のプロセッサポートおよび１つの同期アダプタを備え、同期アダプタは、複数のアダプタポートを備えている。各アダプタポートは、複数のコントローラのうちの１つのコントローラポートに接続され、各コントローラは、複数のコントローラポート、および１つの設定ポートを備えている。各プロセッサは、１つ以上のアダプタポートを介して同期信号をそれぞれの１つ以上のコントローラに送信するように設定され、さらに、１つ以上のコントローラから受信する応答に応じてプログラム命令の実行を停止するように設定される。
【０００９】
さらに別の実施形態では、各コントローラは、設定ポートを備えていてよい。設定ポートは、１つ以上の設定データビットを受信するように設定されてよい。
【００１０】
もう１つの非限定的な実施形態では、各コントローラは、さらに、１つ以上の設定ビットに応じて応答を送信するように設定されてよい。
【図面の簡単な説明】
【００１１】
【図１】コンピューティングシステムの一実施形態を示すブロック図である。

【図２Ａ】一実施形態のコンピューティングシステムのソフトウェアとハードウェアとの階層を示すブロック図である。
【図２Ｂ】テストおよび開発システムを示すブロック図である。
【図３】マルチプロセッサ集積回路の一実施形態を示すブロック図である。
【図４】マルチプロセッサアレイの一実施形態を示すブロック図である。
【図５】同期コントローラの一実施形態を示すブロック図である。
【図６】同期ネットワークのもう１つの実施形態を示すブロック図である。
【図７】同期アダプタの一実施形態を示すブロック図である。
【図８】同期コントローラの一実施形態を示すブロック図である。
【図９】同期の連鎖を示すブロック図である。
【図１０】マルチプロセッサアレイを動作させる方法の一実施形態を描いたフローチャートである。
【図１１】マルチプロセッサアレイを動作させるもう１つの方法の一実施形態を描いたフローチャートである。
【図１２】マルチプロセッサアレイの２つの処理要素の間の同期を示すブロック図である。
【図１３】マルチプロセッサシステムの処理要素を同期させる方法の一実施形態を描いたフローチャートである。
【図１４】マルチプロセッサシステムの３つの処理要素の間の同期を示すブロック図である。
【図１５】マルチプロセッサシステム内の同期コントローラを動作させる方法の一実施形態を描いたフローチャートである。
【図１６】マルチプロセッサシステム内の２つのプロセッサ群を同期させる方法の一実施形態を描いたフローチャートである。
【図１７】マルチプロセッサシステム内の２セットのプロセッサ同期させる方法の一実施形態を描いたフローチャートである。
【図１８】マルチプロセッサシステム内の２セットのプロセッサ同期させる方法の一実施形態を描いたフローチャートである。
【図１９】マルチプロセッサシステム用のソフトウェアを設計する方法の一実施形態を描いたフローチャートである。

【００１２】

本開示は、様々な修正および代替形態が可能であるが、本開示の特定の実施形態を例として図面に示し、本明細書ではこれについて詳述していく。しかしながら、図面およびそれに対する詳細な説明は、図示した特定の形態に本開示を限定する意図はなく、逆にその意図は、付属の特許請求の範囲に規定される本開示の精神および範囲内に収まるあらゆる修正、均等物、代替物を含むことである点を理解すべきである。本明細書で使用する見出しは、編成のみを目的とし、本明細書の範囲を限定するために使用しているのではない。本明細書全体にわたって使用しているように、「ｍａｙ（～であってよい）」という単語は、強制の意味（すなわち、ｍｕｓｔ（～しなければならない）の意味）ではなく、容認の意味（すなわち、その可能性があるという意味）で使用している。同じように、「ｉｎｃｌｕｄｅ（含む）」、「ｉｎｃｌｕｄｉｎｇ（含む）」および「ｉｎｃｌｕｄｅｓ（含む）」という単語は、それを含んでいるという意味であって、限定するものではない。

【００１３】

様々なユニット、回路、またはその他のコンポーネントが、１つまたは複数のタスクを実行する「ように設定された（ｃｏｎｆｉｇｕｒｅｄ　ｔｏ）」と記載されていることがある。このような文脈では、「～するように設定された（ｃｏｎｆｉｇｕｒｅｄ　ｔｏ）」というのは、動作中にその１つまたは複数のタスクを実行する「回路を有する」ことを全体的に意味する構造の広義の詳述である。このように、ユニット／回路／コンポーネントは、そのユニット／回路／コンポーネントがその時点でオンになっていないとしても、タスクを実行するように設定できるものである。一般に、「～するように設定された（ｃ

ｏｎｆｉｇｕｒｅｄ　ｔｏ）」に対応する構造を形成する回路は、ハードウェア回路を含んでいてよい。同じように、様々なユニット／回路／コンポーネントが１つまたは複数のタスクを実行すると便宜上明細書内に記載していることがある。このような記載は、「～するように設定された（ｃｏｎｆｉｇｕｒｅｄ　ｔｏ）」という句を含むものと解釈すべきである。１つ以上のタスクを実行するように設定されたユニット／回路／コンポーネントの詳述は、米国特許法第１１２条第６段落にあるそのようなユニット／回路／コンポーネントに対する解釈を採用しないことを明確に意図している。さらに全体的には、どの要素の詳述も、「ｍｅａｎｓ　ｆｏｒ（～を意味する）」または「ｓｔｅｐ　ｆｏｒ（～のステップである）」という言葉が明記されていない限り、米国特許法第１１２条第６段落にあるそのような要素に対する解釈を採用しないことを明確に意図している。

【発明を実施するための形態】

【００１４】

「Ｐｒｏｃｅｓｓｉｎｇ　Ｓｙｓｔｅｍ　Ｗｉｔｈ　Ｉｎｔｅｒｓｐｅｒｓｅｄ　Ｓｔａｌｌ　Ｐｒｏｐａｇａｔｉｎｇ　Ｐｒｏｃｅｓｓｏｒｓ　Ａｎｄ　Ｃｏｍｍｕｎｉｃａｔｉｏｎ　Ｅｌｅｍｅｎｔｓ」と題し、発明者がＭｉｃｈａｅｌ　Ｂ．Ｄｏｅｒｒ、Ｗｉｌｌｉａｍ　Ｈ．Ｈａｌｌｉｄｙ、およびＤａｖｉｄ　Ａ．Ｇｉｂｓｏｎ、Ｃｒａｉｇ　Ｍ．Ｃｈａｓｅである米国特許第７，４１５，５９４号を参照することにより、本明細書に完全に記載されているかのようにその全容を本明細書に援用する。

【００１５】

２０１１年１０月１４日に出願され、「Ｄｉｓａｂｌｉｎｇ　Ｃｏｍｍｕｎｉｃａｔｉｏｎ　ｉｎ　ａ　Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒ　Ｓｙｓｔｅｍ」と題し、発明者がＭｉｃｈａｅｌ　Ｂ．Ｄｏｅｒｒ、Ｃａｒｌ　Ｓ．Ｄｏｂｂｓ、Ｍｉｃｈａｅｌ　Ｂ．Ｓｏｌｋａ、Ｍｉｃｈａｅｌ　Ｒ　Ｔｒｏｃｉｎｏ、およびＤａｖｉｄ　Ａ．Ｇｉｂｓｏｎである米国特許出願第１３／２７４，１３８号を参照することにより、本明細書に完全に記載されているかのようにその全容を本明細書に援用する。

【００１６】

用語

ハードウェア設定プログラム－例えば集積回路などのハードウェアをプログラムするか設定するために使用できるバイナリイメージにコンパイルできるソーステキストからなるプログラム。

【００１７】

コンピュータシステム－様々な種類のコンピューティングシステムまたは処理システムのうちのいずれかであり、パーソナルコンピュータシステム（ＰＣ）、メインフレームコンピュータシステム、ワークステーション、ネットワーク機器、インターネット機器、携帯情報端末（ＰＤＡ）、グリッドコンピューティングシステム、もしくはその他のデバイス、あるいはデバイスを組み合わせたものなどである。一般に、「コンピュータシステム」という用語は、記憶媒体からの命令を実行する少なくとも１つのプロセッサを有する任意のデバイス（またはデバイスを組み合わせたもの）を含ませるために広義に規定できる。

【００１８】

自動的－コンピュータシステム（例えばコンピュータシステムが実行するソフトウェア）またはデバイス（例えば回路、プログラム可能なハードウェア要素、ＡＳＩＣなど）によって実施されるアクションまたは動作のことであり、そのアクションまたは動作を直接指定したり実施したりするユーザ入力がないこと。そのため、「自動的」という用語は、ユーザが手動で実施または指定する動作で、ユーザが動作を直接実施するための入力を提供するものとは対照的である。自動的な手順であれば、ユーザが提供する入力で開始できるが、「手動で」実施されるその後のアクションはユーザが指定するものではなく、すなわち「手動で」実施されず、ユーザは実施するための各アクションを指定する。例えば、各フィールドを選択して電子フォームに書き込み、（例えば情報をタイピングしたり、チェックボックスを選択したり、無線選択をしたりして）入力して情報を指定しているユー

ザは、コンピュータシステムがユーザの行為に応答して更新しなければならないとしても、手動でフォームに書き込んでいる。フォームは、コンピュータシステムによって自動的に書き込まれてよく、この場合、コンピュータシステム（例えばコンピュータシステムで実行するソフトウェア）は、フォームのフィールドを分析し、フィールドへの回答を指定する何らかのユーザ入力なしにフォームに書き込む。前述したように、ユーザは、フォームの自動書き込みを呼び出すことができるが、その時点のフォームの書き込みに関わってはいない（例えばユーザは、フィールドへの回答を手動で指定してはおらず、むしろフィールドは自動的に完成されている）。本明細書では、ユーザが取った行為に応答して自動的に実施された動作の様々な例を提供する。

【００１９】

概要

並列処理が可能なコンピュータシステムは、複数のデータ処理要素（ＰＥ）、支持メモリ（ＳＭ）ユニット、および帯域幅の広い相互接続ネットワーク（ＩＮ）で設定されて、個々のＰＥ、ＳＭ、およびＩ／Ｏポートシステムどうしの間でデータを移動させることができる。そのようなシステムの第一ＩＮ（またはＰＩＮ）は、帯域幅が広く平均配信時間（待ち時間）が短くなるように最適化されてよい。しかしながら、ＰＩＮは、確実に配信するようには最適化できない（メッセージは、「ビジー」信号になる他のメッセージをブロックできる）。その結果、ＰＩＮは、ＰＥのグループに対してタスクを同期させるのには適していないことがある。いくつかの実施形態では、同期化のためにコンピュータシステムに追加のネットワークを追加することができる。このようなネットワークにより、メッセージを確実に配信できるが、このネットワークにより、コンピュータシステムに対して複雑さ、電力消費、または物理的なサイズが加わることがある。図面に示し、以下に説明した実施形態は、並列処理要素をコンピュータシステム内で同期させると同時に、コンピュータシステムの複雑さ、電力消費、または物理的なサイズに対する影響を最小にする技術を提供できる。

【００２０】

並列処理

旧式のマイクロプロセッサおよびデジタル信号プロセッサ（ＤＳＰ）は、一度に１つのタスクを実行でき、これは一般に、以下のように実行スレッドと呼ばれる。プロセッサのＩＰＵユニットからわかるように、実行スレッドは命令ストリームである。いくつかの実施形態では、命令ストリームに応答して結果の単一のストリームが生成される。この実行方式は一般に、Ｓｉｎｇｌｅ Ｉｎｓｔｒｕｃｔｉｏｎ Ｓｉｎｇｌｅ Ｄａｔａ（単一命令単一データ、ＳＩＳＤ）と呼ばれる。他の実施形態では、複数の算術論理ユニット（ＡＬＵ）を用いて、結果の複数のストリームを可能にすることができる。この実行方式は通常、Ｓｉｎｇｌｅ Ｉｎｓｔｒｕｃｔｉｏｎ Ｍｕｌｔｉｐｌｅ Ｄａｔａ（単一命令複数データ、ＳＩＭＤ）と呼ばれる。さらに大きいマイクロプロセッサおよびＤＳＰは、ＳＩＭＤの能力を有することがあり、様々な実施形態では、そのような並列処理を利用して性能を高めるために、ソフトウェアを用いてよい。例えば、ＳＩＭＤを使用することによって、映像圧縮およびトランスコーディング、コンピュータビジョン、音声認識、および暗号化を加速できる。

【００２１】

ＳＩＭＤ命令によって、命令の効果的な処理が可能になることがある。しかしながら、その効果は、データがＡＬＵに供給される速度によって異なることがある。通常、１つのＳＩＭＤ命令から生じるデータ数は、２から８までの範囲であり、ＳＩＭＤごとに生じるデータ数は増大するため、各データ項目に含まれるビット数は通常減少する。ＳＩＭＤ命令ごとのデータ項目数を増大させる努力をすると、次のような様々な問題が生じることがある：複数のデータ項目が共通のメモリから同時にＡＬＵに供給され、ＡＬＵの結果を格納すると同時にメモリに戻り、動作中に余分な電力消費が起こり、余分なＡＬＵはアイドル状態だが電源が入って準備が整っているときに余計な漏れ電力を消費するなどだが、これに限定されない。

【００２２】
より最近のマイクロプロセッサおよびＤＳＰは、複数のＩＰＵならびにＡＬＵを用いて複数のスレッドを同時に実行できる。何を反復するかや特殊化されるかどうかは、それぞれのマイクロプロセッサ／ＤＳＰの設計タイプによって大きく異なる。それぞれのスレッドには独立した命令ストリームがあってよいため、この並列動作方式は、ｍｕｌｔｉ－ｉｎｓｔｒｕｃｔｉｏｎ ｍｕｌｔｉ－ｄａｔａ（マルチ命令マルチデータ、ＭＩＭＤ）と呼ばれる。プロセッサごとの通常のスレッド数は２～４だが、プロセッサは１６スレッド以上の能力があるように設計されている。プロセッサごとのスレッド数を増大させる努力をすると、命令ストリームが共通のメモリから複数のＩＰＵへ同時に流れ、余分なレジスタに対して余分な漏れ電力を消費するという問題が起こるが、これらは、複数のデータストリームが対応するＡＬＵに供給される問題に加えて起こることである。

【００２３】
マルチプロセッシング
マルチプロセッサシステムによってプログラマは、大型のタスクを、並列に実行できる複数の小さなタスクに分割できる。並列実行を利用して、時間を短縮して大型のタスクを完了するか、あるいは（プロセッサのクロック周波数を低減して）電力消費を削減することができる。クロック周波数が低減すれば、電源の電圧も低下してエネルギーを節約できる。

【００２４】
マルチプロセッサシステムは、様々なコンピュータシステムのうちの１つに用いることができる。コンピューティングシステムの一実施形態を図１に示している。図示した実施形態では、コンピューティングシステム１００は、デスクトップコンピュータ、ラップトップコンピュータ、タブレットコンピュータ、スマートフォン、またはその他の任意の適切なシステムであってよい。コンピューティングシステム１００は、様々な実施形態では、例えば組み込みシステム１１０のような１つ以上の組み込みシステムを備えていてよい。いくつかの実施形態では、組み込みシステム１００は、例えばマルチプロセッサＩＣ１２０のような１つ以上の集積回路（ＩＣ）を備えていてよい。組み込みシステム１１０には１つのＩＣしか描かれていないが、他の実施形態では、異なる数のＩＣを用いてよく、そのそれぞれが異なる機能を実施するように設定されてよい。

【００２５】
図２Ａには、一実施形態のソフトウェアとハードウェアとの階層を描いたブロック図を示している。この階層の下はコンピューティングシステム２４０である。いくつかの実施形態では、コンピューティングシステム２４０は、コンピューティングシステム１００に相当するものであってよい。コンピューティングシステム２４０は、様々な実施形態では、デスクトップワークステーションであってよく、他の実施形態では、コンピューティングシステム２４０は、ラップトップコンピュータまたはその他のモバイルデバイスであってよく、ディスプレイ、ハードディスクドライブ、ネットワークインターフェースデバイスのようなコンポーネント、およびその他の任意の適切なコンポーネントを備えていてよい。

【００２６】
階層の次のレベルは、オペレーティングシステム２５０である。様々な実施形態では、オペレーティングシステム２５０は、例えばＷｉｎｄｏｗｓ（登録商標）、Ｌｉｎｕｘ（登録商標）、Ｕｎｉｘ（登録商標）などの様々なタイプのオペレーティングシステムのうちの１つであってよい。オペレーティングシステム２５０のようなオペレーティングシステムは、様々な実施形態では、コンピューティングシステム２４０のハードウェアにアクセスするためのアプリケーションまたはユーザプログラムに必要なコマンドおよびプログラム命令を提供できる。

【００２７】
上記のように、オペレーティングシステム２５０は、他のプログラムのためにハードウェアリソースへのアクセスを提供できる。図示した実施形態では、このようなプログラム

は、設計ツール一式２１０、ならびにプロジェクトデータベース２２０Ａおよび２２０Ｂを備えている。いくつかの実施形態では、設計ツール一式２１０は、ユーザがハードウェアリソースをコンピューティングシステム２４０内で設定できるように設定されていてよい。以下にさらに詳細に説明するように、このような設定は、制御ビットをマルチプロセッサ内の１つ以上の制御レジスタに格納することを含んでいてよい。制御ビットは、様々な実施形態では、マルチプロセッサの処理要素どうしの間の情報のルーティングを制御できる。いくつかの実施形態では、制御ビットは、マルチプロセッサの処理要素どうしの間の同期も制御できる。

【００２８】

テストおよび開発システムの一実施形態を図２Ｂに示している。図示した実施形態では、マルチプロセッサＩＣ２７０は、開発システム２５０に含まれている。開発システム２５０は、テストベンチ２６０と接続している。様々な実施形態では、テストベンチ２６０は、テスト機材、ラップトップコンピュータ、およびマルチプロセッサＩＣ２７０のテストおよび開発を補佐するその他の任意の適切な機材を備えていてよい。

【００２９】

動作中、開発システム２５０を使用して、所与のソフトウェアアプリケーションと一緒に使用するためにどのようにマルチプロセッサ２７０を設定するかを決定できる。いくつかの実施形態では、設定は、１つ以上の処理要素をどのようにマルチプロセッサ２７０内で同期させて、並列処理中に個々の処理要素にデータを交換させるかを決定することを含んでよい。

【００３０】

図２Ｂに示した実施形態は単なる一例であることに注意されたい。他の実施形態では、異なる数のマルチプロセッサＩＣおよび異なるテスト機材を用いてよい。

【００３１】

マルチプロセッサＩＣの一実施形態を図３に示している。図示した実施形態では、マルチプロセッサＩＣ３００は、プロセッサアレイ３１０を備えている。マルチプロセッサＩＣ３００は、他の実施形態では、他の回路および機能ブロック（図示せず）も備えていてよい。例えば、マルチプロセッサＩＣ３００は、発振器、位相ロックループ（ＰＬＬ）、内部で電源を生成し調節する回路などを備えていてよいアナログ／混合信号ブロックを備えていてよい。

【００３２】

動作中、メモリまたはハードディスクドライブまたはその他の適切な媒体に格納されたプログラム命令をプロセッサアレイ３１０で実行できる。いくつかの実施形態では、プロセッサアレイ３１０内の個々の処理要素（ＰＥ）を、特定のプログラム命令を実行するように設定できる。以下にさらに詳細に説明するように、プログラム命令の実行は、様々な実施形態では、同期を利用してＰＥどうしの間で調整できる。

【００３３】

図４には、マルチプロセッサアレイ（ＭＰＡ）の一例を示している。いくつかの実施形態では、ＭＰＡ４００は、図３に示したマルチプロセッサＩＣ３００のプロセッサアレイ３１０に相当するものであってよい。図示した実施形態では、ＭＰＡ４００は、複数の処理要素（ＰＥ）および複数の支持メモリ（ＳＭ）および１つの相互接続ネットワーク（ＩＮ）を備えている。ＩＮは、スイッチノードとリンクとで設定される。ルータとも呼ばれるスイッチノードをリンクと共に使用して、ＰＥどうしの間およびＰＥとＭＰＡのＩ／Ｏポートとの間に通信経路を形成できる。しかしながら、各ＰＥでは、通信されるどのような情報もＳＭにバッファリングされてよい。図示した実施形態では、ＳＭは、データメモリルータ（ＤＭＲ）と呼ばれる通信経路ルータと組み合わされている。本明細書で使用しているように、ＰＥをＰＥノードと呼ぶこともあり、ＤＭＲをＤＭＲノードと呼ぶこともある。本明細書ではＤＭＲを「設定可能な通信素子（ｃｏｎｆｉｇｕｒａｂｌｅ　ｃｏｍｍｕｎｉｃａｔｉｏｎ　ｅｌｅｍｅｎｔ）、またはＣＣＥ」と呼ぶこともある。

【００３４】

図４に示したようなＤＭＲどうしの間のリンクは、直線のメッシュを形成する。しかしながら、他の実施形態では、他の多くの接続図式が可能であり、構想される。図４に示したＭＰＡ接続図式では、各ＰＥは４つの隣接するＤＭＲに接続しているのに対し、各ＤＭＲは４つの隣接するＰＥに接続するとともに４つの隣接するＤＭＲにも接続している。ＤＭＲ１つあたりにＤＭＲとＤＭＲとの６つのリンクを使用して３次元の直線メッシュを支持するか、あるいはＤＭＲ１つあたりに８つのリンクを使用して東西南北方向に加えて４つの対角線方向を支持するなど、より高次元のＩＮを支持するために他の接続図式も検討される。リンクは、物理的に最も近くにある隣接対象物に限定されない。

【００３５】

ＭＰＡとアプリケーションソフトウェアとを組み合わせたものは、様々な実施形態では、並列処理システム（ＰＰＳ）と呼ぶことがある。例えば、ＭＰＡをプログラムして、カメラからの生の映像データをバッファリングして分析してから、バッファコンテンツに対して映像データの圧縮を実施し、圧縮したデータを無線リンク上に伝送することができる。このアプリケーションソフトウェアとＭＰＡとを組み合わせたものを、例えば並列映像信号プロセッサと呼ぶことがある。

【００３６】

Ｉ／Ｏセルチップを含むＭＰＡ４００を、汎用マイクロプロセッサ、ＤＳＰ、ＦＰＧＡ、またはＡＳＩＣが現在使用されている様々なシステムおよびアプリケーションのいずれかに使用できる。例えば、図４に示した処理システムは、様々なタイプのコンピュータシステム、デジタル信号プロセッサ（ＤＳＰ）または計算を必要とする他のデバイスのいずれかで使用できる。

【００３７】

図４に示したＭＰＡは単なる一例であることに注意されたい。他の実施形態では、異なる数のＰＥおよびＰＥどうしの間の異なる接続機構を用いてよい。

【００３８】

同期

マルチプロセッシングにより、さらに多くのプログラム命令を同時に実行できる可能性があるが、マルチプロセッシングにより、効果的な通信および調整のために、小さいタスクを特定の境界で時間と空間の面で同期させる必要性が生じる可能性あがる。タスクがシステムクロック（クロックサイクルまたは省略して単に「サイクル」とも呼ばれる）の最小ティックで始まる場合、２つ以上のタスクが同期される。タスクの同期は、様々な実施形態では、サイクル数が少ない同期を実施するには困難なことがある。

【００３９】

上記に注意したように、ＰＩＮはメッセージの配信を確実にするものではない。そのため、ＰＩＮはＰＥの同期には適していない。代替策が、各ＰＥおよび共有メモリロケーションでソフトウェアバリアを用いることである。ＰＥがそのバリアに到達すると、ＰＥはロケーション値を上げた後、そのロケーション値が調整すべき予想ＰＥ数と一致するまでその値を定期的にポーリングする。しかしながら、このような技術を用いると、ほとんどのＰＥのアーキテクチャは、一連のＰＥがすべてバリアを超えて同じクロックサイクル内で他のＰＥと同期し続けるのを確実にすることはない。

【００４０】

様々な実施形態では、ＰＥのアーキテクチャは、例えば１から２０サイクルまでのような広いサイクル範囲内でタスクを同期させることができる可能性があるが、特定の瞬間の現在のサイクル数は、プログラマの正常な制御を超えたいくつかの要因、例えば、他のアクティブスレッドの状態、ＳＭで現在処理されているデータのロケーション、キャッシング、割り込み処理などによって異なる。

【００４１】

いくつかの実施形態では、同期ネットワーク（ｓｙｎｃｈｒｏｎｉｚａｔｉｏｎ ｎｅｔｗｏｒｋ）（本明細書では「同期ネットワーク（ｓｙｎｃ ｎｅｔｗｏｒｋ）」とも呼ぶ）を用いて、ＭＰＡの１つ以上のＰＥどうしの間でタスクを同期させることができる。

マルチプロセッサシステムの第一相互接続ネットワーク（ＰＩＮ）は、リンクとノードとで設定されてよく、その場合のノードは、リンクならびに点在する処理要素（ＰＥ）および支持メモリ（ＳＭ）に接続するためのポートを有するが、同期ネットワークを、一連の同期コントローラと、一連のＰＥアダプタおよびその間の接続部と、各ＰＥにおける新たな命令とで設定してよい。

【００４２】

同期ネットワークを備えたＭＰＡの一実施形態を図５に示している。図示した実施形態では、単一の同期コントローラ（Ｃ）をＰＩＮの各ノードに対して使用する。１つのシステムにある同期コントローラはすべて同じであってよい。各同期コントローラは、複数のポートを有してよく、その各々が隣接するＰＥに結合し、１つのポートが設定のためのものであってよい。いくつかの実施形態では、同期コントローラに結合している一連の隣接するＰＥは、最も近いＰＩＮノードが結合している一連のＰＥと同じであってよい。設定ポートは、ＳＭロケーション、ＰＥレジスタ、プログラミング／デバッグするための第二ネットワーク、または設定ポートに設定データを供給するその他の任意の手段に結合していてよい。同期コントローラにあるＰＥポートは、インバウンド用の同期信号およびアウトバウンド用の同期停止信号を有する。

【００４３】

いくつかの実施形態では、各ＰＥは、複数のポートを有する同期アダプタ（Ａ）を用いることができ、各ポートは、同期コントローラに結合するほか、それ自体のＰＥにも結合する。いくつかの実施形態では、同期アダプタをＰＥに組み込むことができ、他の実施形態では、同期アダプタは別のエンティティであってよい。同期コントローラに結合している様々なポートは、ＰＥから見た方向、例えば結合が４つの場合はＮＥ、ＳＥ、ＳＷ、およびＮＷというコンパスの方向で区別することができる。アダプタにある各ポートは、アウトバウンド用の同期信号およびインバウンド用の同期停止信号を有する。さらに高次元のＩＮを支持するために、同期コントローラとアダプタとの間の他の接続図式も検討される。同期コントローラと同期アダプタとの間のリンクは、物理的に最も近い隣接物に限定される必要はない。

【００４４】

図５に示した同期ネットワークは単なる一例であることに注意されたい。他の実施形態では、異なる数の同期コントローラおよびアダプタ、ならびに同期コントローラとアダプタとの間の異なる接続が可能であり、構想される。

【００４５】

いくつかの実施形態では、ＰＥどうしの間の同期のソフトウェアによる制御を用いてよい。このような場合、専用の「同期」命令をＰＥ命令セットに含めてよい。様々な実施形態では、このような命令に対するアセンブリ言語形式は、

同期＜ｄｉｒｅｃｔｉｏｎ　ｌｉｓｔ＞

であってよい。

【００４６】

＜ｄｉｒｅｃｔｉｏｎ　ｌｉｓｔ＞フィールドは、信号を送信（アサート）し、その後同期停止信号をデアサートするのを待つための同期コントローラ方向のリスト（すなわち１つ以上）を指摘できる。ＡＰＥは、次の命令を実行する前にリストに一致するすべての同期停止信号がデアサートされるまで待つことができる。

【００４７】

いくつかの実施形態では、同期コントローラがこのＰＥを１つ以上の他のＰＥと同期させるように設定されていない場合、同期停止信号はデアサートされたままになり、ＰＥは同期停止信号を待つことができない。同期コントローラは、同期設定レジスタ内に「マスクされて」いるＰＥからの同期を無視できるとともに、同期停止をこのＰＥにアサートできない。様々な実施形態では、同期設定レジスタは、ＭＰＡが実行する所与のアプリケーションに特有の設定情報を格納できる。

【００４８】

同期コントローラが所与のＰＥを１つ以上の他のマスクされていないＰＥと同期するように設定され、かつそのマスクされていないＰＥがそのそれぞれの同期信号をまだアサートしていない場合は、同期コントローラは、同期停止信号をアサートし返すことで応答できる。その場合、所与のＰＥは、同期コントローラがマスクされていないＰＥの同期信号をすべて受信し、マスクされていない同期停止信号をすべてデアサートするまで待つことができる。

【００４９】

様々な実施形態では、同期コントローラをＤＭＲの一部として備えてよく、他の実施形態では、各同期コントローラをＭＰＡ内のスタンドアローン型ユニットとして実施してよいことに注意されたい。

【００５０】

図６には、同期ネットワークの一部の一実施形態を示している。図示した実施形態では、同期ネットワーク６００は、同期アダプタ６０１、６０３、および６０５、ならびに同期コントローラ６０２、６０４、および６０６を備えている。それぞれの同期コントローラと同期アダプタとの間の接続部は、２つのワイヤを備えている。１つのワイヤは、同期リクエスト（ｓｙｎｃ＿ｒｅｑｕｅｓｔ、ＳＲ）信号の通信に使用されてよく、もう１つのワイヤは、同期停止（ｓｙｎｃ＿ｓｔａｌｌ、ＳＳ）信号の通信に使用されてよい。いくつかの実施形態では、ＳＲ信号は同期アダプタから同期コントローラに送信されてよく、ＳＳは同期コントローラから同期アダプタに送信されてよい。

【００５１】

所与の同期コントローラと所与の同期アダプタとの間の接続部に備わる２つのワイヤは、４つの状態を符号化できる。第１の状態では、ＳＲとＳＳ信号の両方が、非アクティブを指す論理０レベルであってよい。ＳＲ信号は論理０レベルであってよく、ＳＳ信号は第２の状態で論理１レベルであってよく、これは同期バリアがアクティブであるが、まだリクエストされていないことを指す。第３の状態では、ＳＲ信号およびＳＳ信号は両方とも論理１値であってよく、これは同期バリアがアクティブでリクエストされているがまだ完了していないことを指す。第４の状態では、ＳＲリクエスト信号は論理１値であってよく、ＳＳ信号は、同期バリアが完了したことを指す論理０値であってよい。

【００５２】

図６に示した実施形態は単なる一例であることに注意されたい。他の実施形態では、異なる数の同期アダプタおよび同期コントローラ、ならびに異なる設定の同期コントローラおよび同期アダプタを用いてよい。

【００５３】

同期アダプタの一実施形態を図７に示している。図示した実施形態では、同期アダプタ７００は、接続したＯＲゲート７１０を備えている。同期アダプタ７００は、様々な実施形態では、ＰＥ内に含まれていてよく、他の実施形態では、同期アダプタは、ＭＰＡ内の別のエンティティであってよい。同期アダプタ７００はＯＲゲートを備えているが、他の実施形態では、他の論理ゲートおよび論理ゲートの他の設定を用いてよい。

【００５４】

動作中、同期アダプタ７００は、ＰＥの命令を取得し復号化するユニットと隣接する同期コントローラとの間の通信を仲介できる。ＰＥの命令を取得し復号化するユニットから受信したＳＲ信号は、同期アダプタ７００を介して隣接する同期コントローラまで移動できる。いくつかの実施形態では、ＳＲ信号は、前述した命令のようなソフトウェア命令の関数であってよい。ＳＳ信号はそれぞれの同期コントローラから戻り、ＯＲゲート７１０によって論理的に組み合わされる。その結果生じる信号は、ＰＥの命令を取得し復号化するユニットを停止するために使用されてよい。いくつかの実施形態では、次のＰＥクロックサイクルに対する命令の取得を遅らせるには、隣接する同期コントローラのうちの１つから送信される単一のアクティブなＳＳ信号で十分である可能性がある。

【００５５】

図７に示した同期アダプタは単なる一例であることに注意されたい。他の実施形態では

、異なる数の論理ゲートならびに異なる数のＳＲ信号およびＳＳ信号が可能であり、構想される。

【００５６】

図８には、同期コントローラの一実施形態を示している。図示した実施形態では、同期コントローラ８００は、論理ゲート８０１から８０４、マスクレジスタ８０５を備えている。いくつかの実施形態では、マスクレジスタは、同期コントローラ８００の外部に位置していてもよいし、メモリ内にマッピングされた場所であってもよい。本明細書に記載したようなレジスタは、１つ以上のデータビットを格納するように設定された格納回路の特定の実施形態であってよい。いくつかの実施形態では、レジスタは、ラッチ、フリップフロップなどの１つ以上のデータ格納セルを備えていてよい。レジスタ８０５は、対応するＰＥの各「方向」に対応するマスクビットを備えていてよい。マスクレジスタ８０５には４方向しか描かれていないが、他の実施形態では、異なる数の「方向」が可能であり、構想される点に注意されたい。

【００５７】

動作中、同期コントローラ８００は、例えば図７に示した同期アダプタ７００のような隣接する同期アダプタからＳＲ信号８０６を受信する。すると、論理ゲート８０１から８０４は、受信したＳＲ信号を組み合わせてＳＳ信号８０７を生成することができる。いくつかの実施形態では、ＳＳ信号８０７の生成にマスクレジスタ８０５からの設定ビットを用いてもよい。例えば、マスクビットが論理０であれば、対応する方向からの入力は無視してもよく、その方向に対応するＳＳ信号は、対応するＰＥが停止していないことを指す論理０レベルに設定されてよい。

【００５８】

マスクビットが論理１レベルの場合、対応する方向に対するＳＳ信号の状態は、その方向からのＳＲ信号および対応する論理ゲート内の対応するＯＲゲートの出力によって異なっていてよい。マスクビットが論理０レベルの場合、対応するＳＳ信号および対応するＳＲ信号の状態は、ＳＳ信号の状態に影響を及ぼさない。

【００５９】

２つ以上のマスクビットが論理１レベルの場合、対応するＳＳ信号は論理０レベルになり、これは、様々な実施形態では、ＳＲ信号が論理０レベルの場合に「停止しない」条件を指すことがある。ＳＳ信号に対応するＳＲ信号が論理１レベルであり、少なくとも１つの他のＳＲ信号が論理１レベルであれば、ＳＳ信号は、「停止」条件を指してよい論理１レベルになる。

【００６０】

「低い」、「低い論理レベル」または「論理０レベル」とは、アースでの電圧またはアース近くの電圧のことであり、「高い」、「高い論理レベル」または「論理１レベル」とは、ｎチャネル型ＭＯＳＦＥＴをオンにし、ｐチャネル型ＭＯＳＦＥＴをオフにするのに十分な大きさの電圧レベルのことである点に注意されたい。他の実施形態では、異なる技術では「低い」および「高い」に対して異なる電圧レベルになることがある。図８に描いた同期コントローラの実施形態は単なる一例であることにさらに注意されたい。他の実施形態では、異なる論理ゲートおよび論理ゲートの異なる設定を用いてよい。

【００６１】

どのような実際のプロセッサアレイであっても、ＤＭＲに接続されるＰＥには有限数ｎがあり、これは１クロックサイクルでＤＭＲが同期できるＰＥの最大数である点に注意されたい。いくつかの実施形態では、この数は４であってよいが、他の実施形態では、異なる数を用いてよい。ＰＥ１つあたりにｔ個の実行スレッドがある場合、各同期コントローラポートおよび各アダプタポートで同期信号の数および同期停止信号の数にｔを乗算することによって、ｔ×ｎ個のスレッドを単一のＤＭＲと同期させるとができる。ｎ個よりも多いＰＥを同期させるためには、異なる技術を用いてよい。

【００６２】

大規模なＰＥグループ全体に対して慎重に構築した一連の同期コマンドをプログラミン

グすることで、どのような数のＰＥでも同期させることができるが、１つのクロックサイクルで即座に同期させることはできない。このプログラミング技術は、グループ内の最も外側にあるＰＥから、そのグループの中心にあるいくつかのＤＭＲまで停止バリアが行き渡るように調整し、その後、中心のＤＭＲから最も外側のＰＥまでリリース波が伝播するように調整する技術である。いくつかの実施形態では、同期を微調整するために非動作命令（一般に「ｎｏｐｓ」と呼ばれる）を追加してよい。

【００６３】

図９には、１つのラインに配置された６つのＰＥの同期を示す一例を示している。ＰＥはどのような形状に配置されてもよいが、明瞭化のために例として１つのラインを選択した点に注意されたい。

【００６４】

ＰＥのラインに対して、ＤＭＲを使用して図９に示したＰＥの対を同期させることができる。ＤＭＲのＮＥポートおよびＮＷポートにある両方のＰＥが同期信号をＤＭＲに発信した場合に限り、同期停止から一対のＰＥを解放するように所与のＤＭＲを設定できる。他のＰＥが割り込むのを防ぐため、５つのＤＭＲすべてをそのＳＷ方向およびＳＥ方向で同期ポートをマスクする（無視する）ように設定する（これは、ＳＷ、ＳＥ方向で他のＰＥへのＤＭＲの接続がないこと以外は図示していない）。ＤＭＲのこの設定は、各ＤＭＲにある同期設定レジスタに設定ビットを格納することによって、同期命令よりも前に行われる。

【００６５】

図９に描いた６つのＰＥを同期させるのに必要な同期プログラミングも図９に示している。各ＰＥの下に一連の命令を列挙している。各ＰＥは、対応する命令を上から下に向かって実行する。破線で示したコネクタラインは、異なるＰＥに対してどの同期命令が対になって共通のＤＭＲ（対になっている両ＰＥに接続しているＤＭＲ）を介して一緒に動作するのかを示している。各ＰＥは、異なるクロックサイクルで第１の同期命令に到達してよいが、対になっているＰＥと同じクロックサイクルでその同期命令を終了するようハードウェアによって強制される。この特定の連鎖の例ではＰＥが対になっているが、前述したＤＭＲの制約に従って、必要に応じて３つ以上のＰＥを単一のサイクルで同期させることができる。

【００６６】

図９に示したＰＥのプログラミングを検証した上で、中心で交差している２つのチェーンに注意されたい。第１のチェーンは、ＰＥ００およびＰＥ０１に対する第１の対の同期命令からなる第１のリンク、ＰＥ０１とＰＥ０２との間の第１の対の同期命令からなる第２のリンク、ＰＥ０２とＰＥ０３との間の唯一の対の同期命令からなる第３のリンク、ＰＥ０３とＰＥ０４との間の第２の対の同期命令からなる第４のリンク、およびＰＥ０４とＰＥ０５との間の第２の対の同期命令からなる第５のリンクを有する。第２のチェーンは、同様に形成されるが、第１のチェーンに対するミラー像のように形成される。つまり、ＰＥ０５との間の第１の対の同期命令からなるリンクで始まり、ＰＥ００とＰＥ０１との間の第２の対の同期命令からなる第５のリンクで終わる。

【００６７】

そのため、ＰＥ０１は第１の同期ＳＷ命令で停止から解放されると、第１の同期ＳＥ命令に捕捉され、これは、ＰＥ０２の第１の同期ＳＷ命令と対になることによってチェーンの次のリンクになる。ＰＥ０２の列についても同様に、ＰＥ０２は第１の同期ＳＷ命令から解放されると、第１の同期ＳＥ命令に捕捉され、これは、ＰＥ０２の第１の同期ＳＷ命令と対になることによってチェーンの次のリンクになる。これ以降も同じように続く。

【００６８】

各ＰＥがプログラムされたタスクを有し、そのそれぞれのタスクが別々に、かつ場合によっては任意の時間をかけてよいと仮定すると、ＰＥは、もう１つの反復に向けて準備するためにデータを交換することになる。このデータ交換に対して準備するために、ＰＥは図９の同期プログラミングと同期されてよい。

【００６９】
各ＰＥはそのタスクを終了すると、図９にあるそのＰＥの対応する列で第１の同期命令を実行する。チェーン内の同期命令にヒットするのが第１のＰＥであれば、待機する。さらに多くのＰＥが各チェーンの上半分に到達してリンクを解放した場合、各ＰＥは、下半分のチェーンにある次の同期命令に進み、待機する。最終的に両チェーンの上半分が解放され、中央のＤＭＲ０３の端から端までのリンクが解放される。この時点で全ＰＥが下半分のチェーンで待機しているため、これらのチェーンは、１サイクルあたり１リンクの割合で高速に連続してＰＥを解放する。
【００７０】
図示したように、同期終了の波が外側へ向かう間、内側のＰＥが、最も外側のＰＥが解放されるのを待つようにする必要があることがある。これは、終了が起こる同期命令の後にｎｏｐｓを追加することによってプログラムできる。ＰＥがｎｏｐ命令を１つ実行すると、１クロックサイクル分を待機する。各ＰＥに対してプログラム内で使用するｎｏｐｓの数は、外側へ向かう同期の実行がそれぞれ正確に１クロックサイクル分かかることを把握した上で計算される。外側へ向かう同期の実行はそれぞれ１サイクルかかり、ｎｏｐｓはいずれも１サイクルかかり、停止する可能性のあるコードを実行しているＰＥはないため、すべてのＰＥを同期させる形で解放できる。図示したプログラミングは、すべてのＰＥが命令の連続を正確に同じクロックサイクルで終了するようにする。
【００７１】
この技術は、数千のＰＥを含むＭＰＡにもスケーラブルである。例えば、２０２５個のＰＥからなる正方向のアレイであれば、エッジの長さはＰＥ４５個分、つまりＰＥからＰＥまでの中継点が４４個であり、対角線のマンハッタン（階段状の）距離の中継点は８８個である。同期チェーンは、領域全体を覆うように放射状のファン模様に設定できる。角から角まで通っているチェーンは、中継点８８個分の長さであり、これらの中継点が２０２５個のＰＥすべてを同期するための最悪の場合の時間間隔を決定し、この場合は８８クロックサイクルである。ＰＥが停止する動作中は、同期命令に遭遇すると、最終的にこれはアレイの中央まで通信される。停止は中央から波の形で解放され、この波は放射状に外側に向かって伝播する。Ｎｏ－ｏｐｓ（非動作）は、中央により近いＰＥを波が角に到達するまで遅らせるために必要になることがある。その場合、全ＰＥは、連続する命令を同じクロックサイクルで開始できる。
【００７２】
同期命令は、配信された支持メモリを含むプロセッサグループを調整して並列プログラムを実行するために使用できる。並列プログラムには多くの様々な種類がある。
【００７３】
この命令によりプログラマは、大規模なＭＰＡにある複数のプロセッサをロックステップの実行に組み入れることができる可能性がある。これによって、このようなアレイの並列処理の利便性を、特にリアルタイムで処理するタスクや、収縮モードでの動作で大いに高めることができる。収縮モードとは、鼓動サイクルの長さが１以上であってよい鼓動を有するようにＭＰＡがプログラムされるプログラミング方法である。各鼓動においていくつかのデータ項目が各ＰＥに受信され、いくつかの動作が実施され、その結果が隣接するＰＥに出力される。収縮モードは、ＰＥ１つあたりにほとんどメモリを必要とせず、いくつか例を挙げると、行列数学、フィルタリング、および画像処理アプリケーションに適用されてきた。一次元の収縮モードは、通常パイプラインと呼ばれる。これらおよびその他の並列プログラミング方法によりプログラマは、大規模な単一タスクの計算力を多くの小さなタスクに分割しやすくなる。
【００７４】
単一命令複数データ（ＳＩＭＤ）の能力を含む従来のコンピュータ／ＤＳＰシステムでは、並列処理は、１つのＰＥサイクルにあるＡＬＵに対して利用可能にできるデータ項目数によって制限される。通常この数には一定の最大値があり、マイクロプロセッサの場合は通常２～８であり、スーパーコンピュータでは最大でおそらく１２８である。ＭＩＭＤ

の並列処理が可能な従来のマルチプロセッサシステムは、その複数の命令ストリーム中の同じ命令を用いてプログラムできる；しかしながら、これ単独では、関連するＰＥが始動したりロックステップの実行に残ったりすることは確実にはならない。

【００７５】

同期命令の連鎖を用いることによって、大規模なＰＥグループの開始を、いくつかの実施形態では、同じサイクルで開始でき、同じ命令を同時に実行するＡＬＵの数に対するＳＩＭＤの制限を克服する。同じタスクの複数のコピーを実行することで、複数のＰＥがロックステップにとどまることが可能だが、確実ではない。ＰＥは、データ値、データメモリまたは通信リソースに対する干渉が原因で起こる待機状態、割り込み、ブレークポイントなどに対してサイクルのカウントが依存していない場合に限り、ロックステップにとどまることができる。長期間にわたってロックステップの実行を達成するには慎重なプログラミングが必要になることがある。しかしながら、これがうまくいかない場合は、上記の方法によって複数のスレッドを定期的に再度同期させることができる。

【００７６】

ハードウェア（およびそれに伴い電力）の影響は、追加された同期命令の能力に対しては極めて低い。必要でないＰＥは閉じてよい。そのため、電力は、アルゴリズムが必要な時のみに使用され、ＳＩＭＤ命令を実施させるのに必要だが常時使用するわけではないオーバーヘッド電力の一部ではない。

【００７７】

図１０には、図５に描いたアレイＭＰＡ５００のようなマルチプロセッサアレイを動作させる方法の一実施形態を示している。この方法はブロック１００１から始まる。次に、マルチプロセッサシステムに対して設定およびソフトウェアを設計できる（ブロック１００２）。いくつかの実施形態では、ソフトウェアは、様々なアプリケーションプログラムのうちのいずれか１つであってよく、その個々のプログラム命令は、ＭＰＡ内にある個々のＰＥで作動可能なものであってよい。設定は、いくつかの実施形態では、ＰＥの時間面での共通点でデータ交換を可能にするための同期の命令およびセッティングを含んでいてよい。

【００７８】

ソフトウェアおよび設定の設計が完了すると、マルチプロセッサシステムを設定できる（ブロック１００３）。いくつかの実施形態では、マルチプロセッサシステムの設定は、例えば図８に示したレジスタ８０５のような設定レジスタに情報を格納することを含んでいてよい。他の実施形態では、設定データは、マルチプロセッサシステム内に備わっている１つ以上のメモリに格納されてよい。

【００７９】

マルチプロセッサアレイの設定が完了した状態で、事前に設計したソフトウェアをマルチプロセッサシステムにロードできる（ブロック１００４）。いくつかの実施形態では、ソフトウェアは、マルチプロセッサシステム内に備わっている１つ以上のメモリにロードされてよい。ソフトウェアは、他の実施形態では、例えばハードディスクドライブ、ＣＤなどのコンピュータアクセス可能媒体、またはその他の任意の適切な記憶媒体に格納されてよい。

【００８０】

ソフトウェアがマルチプロセッサシステムにロードされると、マルチプロセッサの個々のＰＥが開始される（ブロック１００５）。いくつかの実施形態では、各ＰＥは、ロードされたソフトウェア内に含まれる特定の命令セットを実行できる。各ＰＥは、様々な実施形態では、命令の実行を停止して、マルチプロセッサシステム内の他のＰＥによって実行されている命令の完了を保留できる。ＰＥが命令を実行している状態で、本方法はブロック１００６で終了できる。

【００８１】

図１０に描いた方法は単なる一例であることに注意されたい。他の実施形態では、異なる動作および異なる順序の動作を用いてよい。

【００８２】
マルチプロセッサシステムを動作させる方法のもう１つの実施形態を図１１に示している。本方法はブロック１１０１から始まる。次に、マルチプロセッサシステムを動作させる複数のセットを設計できる（ブロック１１０２）。いくつかの実施形態では、各セットは、設定データおよびソフトウェアアプリケーションを含んでいてよい。設定データは、様々な実施形態では、そのセットに含まれる特定のソフトウェアアプリケーションに唯一のものであってよい。
【００８３】
複数のセットを規定した状態で、複数のセットの第１のセットの設定データに基づいてマルチプロセッサシステムを設定でき、その後、対応するソフトウェアアプリケーションを実行できる（ブロック１１０３）。いくつかの実施形態では、マルチプロセッサシステム内の様々なＰＥが、ソフトウェアアプリケーションの一部である様々な命令を実行してよい。
【００８４】
第１のセットからソフトウェアアプリケーションが実行されると、複数のセットの次のセットから得たデータでマルチプロセッサシステムを設定できる（ブロック１１０４）。新たに設定したマルチプロセッサシステムを用いて対応するソフトウェアアプリケーションを作動できる。いくつかの実施形態では、対応するソフトウェアアプリケーションのタスクを実行しているプロセッサは、停止させられてよく、本方法を前述したブロック１１０３から進めてよい。本方法は、ブロック１１０５で終了できる。
【００８５】
図１１に示した方法は単なる一例であることに注意されたい。他の実施形態では、異なる動作および異なる順序の動作が可能であり、構想される。
【００８６】
図１２には、マルチプロセッサシステムの２つのＰＥを同期させる一実施形態を描いたブロック図を示している。図示した実施形態では、処理要素Ｐ１およびＰ２は、それぞれ方向Ｄ１およびＤ２を介して同期コントローラＣ１に接続している。ＰＥのＰ１およびＰ２によって処理されている各スレッドは、同期命令を含んでいる。いくつかの実施形態では、同期命令は、上記にさらに詳細に記載したような方法を含んでいてよい。
【００８７】
動作中、マルチプロセッサシステムを設定し、ソフトウェアアプリケーションをロードして実行できる。ＰＥのＰ１およびＰ２がそれぞれの指定タスクを実行している際に、同期命令に遭遇することがある。同期命令に遭遇する第１のＰＥは、他のＰＥがそのそれぞれの同期命令に遭遇するまで実行を停止できる。その時、２つのＰＥはデータを交換でき、その後そのＰＥのそれぞれのスレッドの実行を再開できる。
【００８８】
図１２に示した実施形態は単なる一例であることに注意されたい。他の実施形態では、異なる数のＰＥおよび異なる数の同期コントローラを用いてよい。
【００８９】
マルチプロセッサシステム内でＰＥを同期させる方法の一実施形態を示すフローチャートを図１３に描いている。図１２のブロック図および図１３のフローチャートを合わせて参照すると、本方法はブロック１３０１から始まる。次に、同期バリアよりも前にあるソフトウェア命令をプロセッサＰ１およびＰ２によって実行できる（ブロック１３０２）。
【００９０】
次に、プロセッサＰ１は、同期命令に遭遇でき、その同期命令に応答して、同期リクエストをコントローラＣ１に送信できる（ブロック１３０３）。いくつかの実施形態では、マルチプロセッサシステム内の様々なプロセッサが実行する命令スレッドどうしの間の同期をそれぞれが要求する場所をソフトウェア内で特定できる。
【００９１】
プロセッサＰ１が同期命令を送信すると、プロセッサＰ１は、コントローラＣ１が生成

した停止信号に応答する実行を停止できる（ブロック１３０４）。図１２には単一のコントローラを示しているが、他の実施形態では、１つのプロセッサを複数のコントローラに接続してよく、複数のコントローラのいずれか１つから得た停止信号で、さらに他のプログラム命令の実行を停止できる。様々な実施形態では、停止信号は、コントローラＣ１がプロセッサＰ１とプロセッサＰ２との両方から同期リクエストを受信した時点でデアセートされてよい。コントローラＣ１が両方のプロセッサから同期リクエストを受信したとき、両プロセッサは同期バリアに到達していて、「同期している」と通知される。コントローラＣ１は、様々な実施形態では、図１５に関して以下に記載する方法と同様に、停止信号をいつデアサートするかを決定できる。

【００９２】

停止信号のデアサートによって、プロセッサＰ１は、同期バリアの後にソフトウェアの実行を再開できる（ブロック１３０５）。その時点で、本方法はブロック１３０６で終了できる。図１３に示した方法には同期リクエストのみを描いているが、他の実施形態では、所与のプロセッサの命令スレッドに複数の同期命令を挿入してよい。追加の同期命令によって、いくつかの実施形態では、マルチプロセッサシステム内のさらに多数のプロセッサを同期させることが可能なことがある。

【００９３】

図１３に示した方法は単なる一例であることに注意されたい。他の実施形態では、異なる動作および異なる順序の動作を用いてよい。

【００９４】

図１４には、マルチプロセッサシステムの３つのＰＥを同期させる一実施形態を描いたブロック図を示している。図示した実施形態では、処理要素Ｐ１は、方向Ｄ１を通って同期コントローラＣ１に接続し、処理要素は、方向Ｄ２およびＤ４を通って同期コントローラＣ１およびＣ２にそれぞれ接続し、処理要素Ｐ３は、方向Ｄ３を通って同期コントローラＣ２に接続している。

【００９５】

設計段階では、３つのＰＥを動作させるための設定が設計される。いくつかの実施形態では、この設定は、同期コントローラＣ１およびＣ２内のマスクレジスタにロードされるデータビットを含んでいてよい。データビットは、前述した方向Ｄ１からＤ４に応じて決定されてよい。例えば、方向Ｄ１およびＤ２以外の全方向をマスクするようにＣ１のレジスタを設定できる。設計は、関連ソフトウェアの命令スレッドのどこに同期命令を挿入するかを決定することを含んでいてもよい。データビットおよび同期命令は、一緒にバリアを形成する、すなわち、全スレッドがその地点に到達するまで３つの処理要素の各々が待機する時間に１つの場所を形成することができる。

【００９６】

次に、設定データおよび関連ソフトウェアは、マルチプロセッサシステムにロードされてよい。ロードされると、Ｐ１、Ｐ２、およびＰ３の各々に対して標的となった命令（スレッドとも呼ばれる）を実行できる。Ｐ１が同期命令に遭遇したとき、Ｐ１は、Ｐ２がＤ２方向で同期命令に遭遇するまでそのスレッドの実行を停止できる。同じように、Ｐ３が同期命令に遭遇したとき、Ｐ３は、Ｐ２がＤ４方向で同期命令に遭遇するまでそのスレッドの実行を停止できる。

【００９７】

Ｐ１およびＰ３に対する同期命令がＰ２に対するどの同期命令よりも先に到着した場合、同期命令がＤ２方向で到着することによって、Ｐ１を次のクロックサイクルのＤ１方向でその第２の同期命令まで進めることができる。次のクロックサイクルでも、Ｐ２がＤ４方向で同期命令に遭遇すると、これによってＰ３をＤ３方向でその第２の同期命令まで進めることができる。第３のクロックサイクルでは、Ｐ２がＤ２方向で同期命令に到達することによって、１つのサイクルでＰ１と再度同期させることができる。いくつかの実施形態では、Ｐ３に対してはｎｏ－ｏｐ命令が望ましいことがある。

【００９８】

Ｐ１がＰ２の後にバリアに到達した場合、Ｐ２は、Ｐ１が到達するまでその同期（Ｄ２）命令で待機する。Ｐ３がＰ２の後にバリアに到達すれば、Ｐ２は、Ｐ３がバリアに到達するまでその同期（Ｄ４）命令で待機する。

【００９９】

図１４に示した実施形態は単なる一例であることに注意されたい。他の実施形態では、異なる数のＰＥおよびコントローラを用いてよい。

【０１００】

図１５には、例えば図８の同期コントローラのような同期コントローラを動作させる方法の一実施形態を描いたフローチャートを示している。本方法はブロック１５０１から始まる。次に、設定を決定できる（ブロック１５０２）。いくつかの実施形態では、設定データは、例えば図８に示したレジスタ８０５のようなレジスタに含まれていてよい。このようなレジスタに格納されている設定データビットは、様々な実施形態では、同期情報の受信をどの方向から許可するのかを決定するために復号化されてよい。それぞれの方向は、隣接するプロセッサのうちの対応する１つに一致していてよい。

【０１０１】

次に、全方向からアサートされた同期リクエストを設定と比較することができる（ブロック１５０３）。いくつかの実施形態では、この比較は、例えば図８に示した論理回路８０１のような論理回路を用いて実施できる。次に同期停止信号は、受信した同期リクエストおよび設定に応じてアサートされるかデアサートされてよい（ブロック１５０４）。いくつかの実施形態では、図８の論理回路８０１のような論理回路が、１つ以上の同期リクエストと設定情報とを論理的に合わせて、所与の停止信号をいつアサートまたはデアサートすべきかを判断できる。いくつかの実施形態では、２つ以上の同期停止信号を任意の所与の時間にアサートしデアサートできる点に注意されたい。次に本方法はブロック１５０５で終了できる。図１５に示したフローチャートには、そこに描いた動作の単一の適用例を示しているが、様々な実施形態では、図１５の方法をマルチプロセッサシステムの動作中に常時実施してよい。

【０１０２】

図１５に示した方法は単なる一例であることに注意されたい。他の実施形態では、異なる数の同期リクエストおよび設定データビットが可能であり、構想される。

【０１０３】

マルチプロセッサシステム内の２つのグループのプロセッサを同期させる方法の一実施形態を描いたフローチャートを図１６に示している。本方法はブロック１６０１から始まる。次に、マルチプロセッサシステムの第１のグループのプロセッサの同期を開始できる（ブロック１６０２）。いくつかの実施形態では、第１のグループのプロセッサの同期は、図１３および図１５に記載した方法に関して前述した同期に似た動作を含んでいてよい。第１のグループは、様々な実施形態では、マルチプロセッサシステムの１つ以上のプロセッサを備えていてよい。いくつかの実施形態では、マルチプロセッサシステムのプロセッサの第１のグループの同期は、複数のクロックサイクルを完了するよう要求できる。

【０１０４】

プロセッサの第２のグループの同期も開始できる（ブロック１６０３）。第１のグループの同期と同じく、プロセッサの第２のグループの同期は、図１３および図１５に記載した方法に関して前述した同期に似た動作を含んでいてよい。第２のグループは、様々な実施形態では、第１のグループに含まれるプロセッサを排除するマルチプロセッサシステムの１つ以上のプロセッサを含んでいてよい。いくつかの実施形態では、１つ以上のプロセッサは、第１のグループと第２のグループとの間で共有されてよい。第１のグループが同期していると、第２のグループの同期は、複数のクロックサイクルが完了することを要求できる。

【０１０５】

次に本方法は、第１のグループおよび第２のグループの同期動作の状態に依存してよい（ブロック１６０４）。同期動作の一方または両方ともが完了していないとき、２つのグ

ループのプロセッサによるその先の実行が停止したままになる（ブロック１６０５）。両方の同期動作が完了すると、第１のグループのプロセッサは、プログラム命令の実行を再開できる（ブロック１６０６）。第２のグループのプロセッサは、プログラム命令の実行も再開できる（ブロック１６０７）。両グループのプロセッサが実行を再開すると、本方法はブロック１６０８を終了できる。実行を再開する２つの動作は、連続する形で実施されるように描かれている点に注意されたい。他の実施形態では、これらの動作は、並列または逆の順序で実施されてもよい。代替実施形態でその他の動作およびその他の順序の動作を用いてもよい。

【０１０６】

図１７には、マルチプロセッサシステムにある２セットのプロセッサを同期させる方法の一実施形態を描いたフローチャートを示している。本方法はブロック１７０１から始まる。次に、マルチプロセッサシステムの第１セットのプロセッサを同期させることができる（ブロック１７０２）。いくつかの実施形態では、同期は、図１３および図１５に記載した方法に関して前述した同期に似た動作を含んでいてよい。第１セットのプロセッサに含まれる各プロセッサは、様々な実施形態では、例えば図８に示した同期コントローラのような共通の同期コントローラに接続されてよい。

【０１０７】

マルチプロセッサシステムの第１セットのプロセッサが同期すると、次にマルチプロセッサシステムの第２セットのプロセッサを同期させることができる（ブロック１７０３）。様々な実施形態では、第２セットの各プロセッサは、共通の同期コントローラに接続されてよい。第１セットのプロセッサを同期させた状態で、第２セットのプロセッサの同期は、図１３および図１５に記載した方法に関して前述した同期に似た動作を含んでいてよい。

【０１０８】

第２セットのプロセッサの同期が完了すると、第１セットのプロセッサから１つのプロセッサを第２セットのプロセッサの１つのプロセッサと同期させることができる（ブロック１７０４）。いくつかの実施形態では、第１セットのプロセッサからのプロセッサ、および第２セットのプロセッサからのプロセッサは、共通の同期コントローラに接続されてよい。

【０１０９】

第１セットからのプロセッサおよび第２セットからのプロセッサが同期すると、第１セットのプロセッサを再同期させることができる（ブロック１７０５）。次に第２セットのプロセッサを再同期させることができる（ブロック１７０６）。２つの再同期動作は、連続する形で実施するように示しているが、他の実施形態では、動作は並列または逆の順序で実施してよい点に注意されたい。本方法はブロック１７０７で終了できる。図１７に描いた方法を用いることによって、様々な実施形態では、マルチプロセッサシステムのどのような数のプロセッサも同期させることができる。

【０１１０】

図１７のフローチャートに描いた方法は単なる一例であることに注意されたい。他の実施形態では、異なる動作および異なる順序の動作が可能であり、構想される。

【０１１１】

マルチプロセッサシステムの２セットのプロセッサを同期させる方法のもう１つの実施形態を図１８のフローチャートに示している。本方法はブロック１８０１から始まる。次に、第１セットのプロセッサのマルチプロセッサシステムを同期させることができる（ブロック１８０２）。いくつかの実施形態では、同期は、図１３および図１５に記載した方法に関して前述した同期に似た動作を含んでいてよい。第１セットのプロセッサに含まれる各プロセッサは、様々な実施形態では、例えば図８に示した同期コントローラのような共通の同期コントローラに接続されてよい。

【０１１２】

マルチプロセッサシステムの第１セットのプロセッサが同期すると、次にマルチプロセ

ッサシステムの第２セットのプロセッサを同期させることができる（ブロック１８０３）。様々な実施形態では、第２セットの各プロセッサは、共通の同期コントローラに接続されてよく、第２セットのプロセッサに含まれる少なくとも１つのプロセッサも第１セットのプロセッサに含まれる。第１セットのプロセッサを同期させた状態で、第２セットのプロセッサの同期は、図１３および図１５に記載した方法に関して前述した同期に似た動作を含んでいてよい。

【０１１３】

マルチプロセッサシステムの第２セットのプロセッサの同期が完了すると、次に第１セットのプロセッサを再同期させることができる（ブロック１８０４）。いくつかの実施形態では、第１セットおよび第２セットのプロセッサに少なくとも１つのプロセッサを含むことによって、第１と第２の両方のセットにある全プロセッサを同期させることが可能になる。第１セットのプロセッサの再同期が完了すると、本方法はブロック１８０５で終了できる。図１８に描いた方法を用いることによって、様々な実施形態では、マルチプロセッサシステムのどのような数のプロセッサも同期させることができる。

【０１１４】

図１８に示したフローチャートは単なる一例であることに注意されたい。他の実施形態では、代替動作を用いてよい。

【０１１５】

マルチプロセッサシステムに対してソフトウェアを設計する方法の一実施形態を描いたフローチャートを図１９に示している。本方法はブロック１９０１から始まる。次にソフトウェアを、例えば図３に示したマルチプロセッサＩＣ３００のようなマルチプロセッサシステムに対して設計できる（ブロック１９０２）。ソフトウェアは、様々な実施形態では、プロセッサのアレイ上にマッピングできるグラフィックスや映像データ、またはその他の任意の適切なアプリケーションを処理するためのアプリケーションを含んでいてよい。いくつかの実施形態では、他の命令スレッドとの同期を要求する個々のプロセッサの命令スレッド内での場所を特定できる。前述したもののような同期命令は、命令スレッドに挿入して同期バリアを形成できるとともに、マルチプロセッサシステム内にある１つ以上のプロセッサの間での同期を可能にすることができる。

【０１１６】

同期命令が挿入されると、設定データを設計できる（ブロック１９０３）。いくつかの実施形態では、設定データは、同期コントローラが、この同期コントローラに接続している１つ以上のプロセッサから同期リクエストを受け入れることができる設定データビットを含んでいてよい。所与の１セットの設定ビットの各ビットは、いくつかの実施形態では、接続しているプロセッサへ向かう対応する方向を表していてよいが、他の実施形態では、所与の１セットの設定ビットは、同期を入力するために許された方向を決定するために復号化されてよい。

【０１１７】

次に、設定データをマルチプロセッサシステムにロードできる（ブロック１９０５）。いくつかの実施形態では、設定データの一部を、例えば図８に示したレジスタ８０５のようなコントローラ内のレジスタにロードできる。設定データは、他の実施形態では、マルチプロセッサシステム内の１つ以上のメモリにロードできる。

【０１１８】

次に、設計したソフトウェアをマルチプロセッサシステムにロードできる（ブロック１９０５）。いくつかの実施形態では、ソフトウェアをマルチプロセッサシステム内にある１つ以上の共有メモリにロードできる。ソフトウェアは、他の実施形態では、小分けでき、ソフトウェアの個々のパーツをマルチプロセッサシステム内のローカルメモリにロードできる。このようなローカルメモリそれぞれをマルチプロセッサシステム内の対応するプロセッサに接続できる。

【０１１９】

設定データおよびソフトウェアがマルチプロセッサシステムにロードされると、ソフト

ウェアを実行できる（ブロック１９０６）。実行中、各プロセッサは、挿入された同期命令を実行して、命令スレッド内に前もって特定された場所で、プロセッサの様々なサブセット、または様々なプロセッサにそれ自体の動作を同期させることができる。ソフトウェアの実行が完了すると、本方法はブロック１９０７で終了できる。

【０１２０】

図１９のフローチャートに示した動作のいくつかは、連続する形で実施されているように描かれている点に注意されたい。他の実施形態では、１つ以上の動作を並列に実施できる。

【０１２１】

本発明のシステムおよび方法を好適な実施形態に結びつけて記載してきたが、本発明は本明細書に記載した特定の形態に限定されることを意図しているわけではなく、逆にそのような代替案、修正、および均等物は、付属の特許請求の範囲に規定した本発明の精神および範囲内に正当に含まれてよいため、これを範囲に含めることを意図している。

【図１】

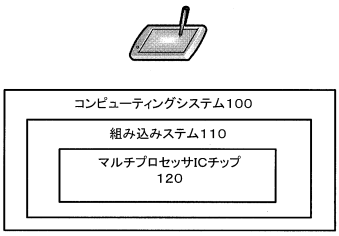

FIG. 1

【図２Ａ】

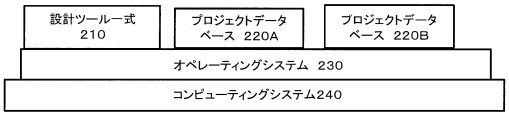

FIG. 2A

【図２Ｂ】

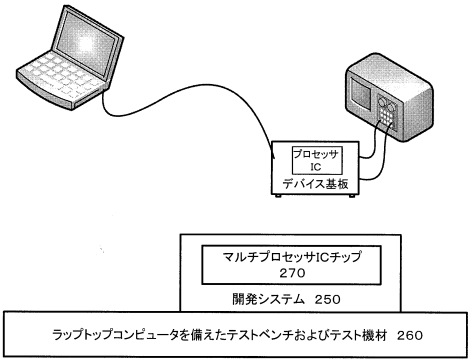

FIG. 2B

【図３】

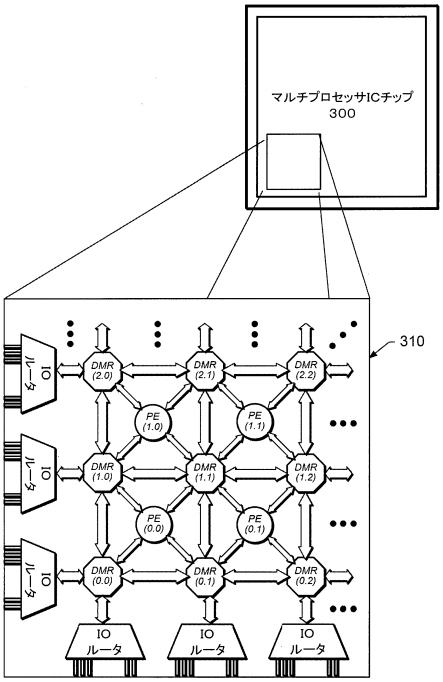

FIG. 3

【図４】

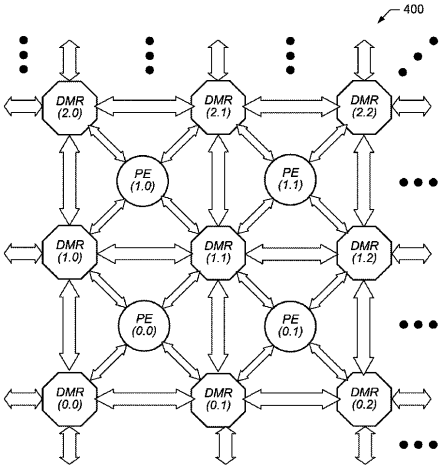

FIG. 4

【図５】

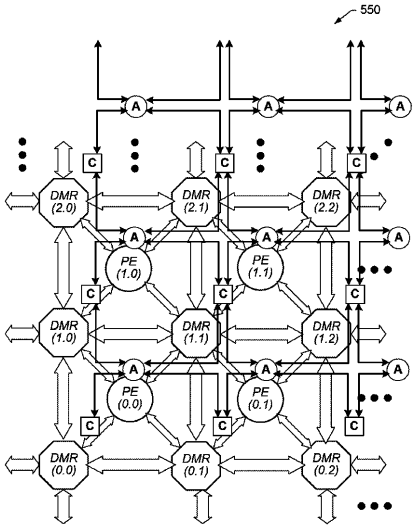

FIG. 5

【図６】

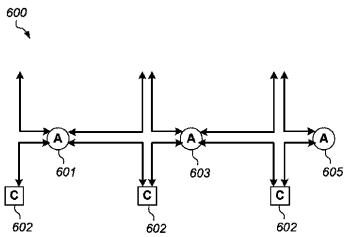

FIG. 6

【図７】

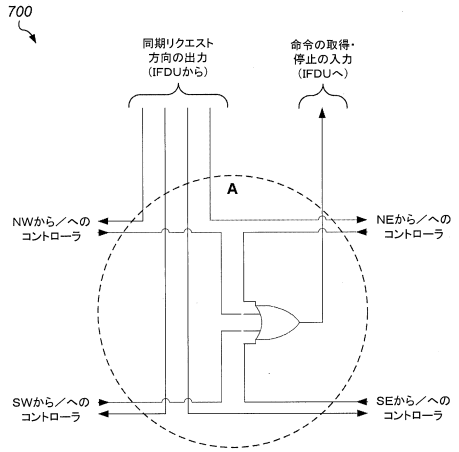

FIG. 7

【図８】

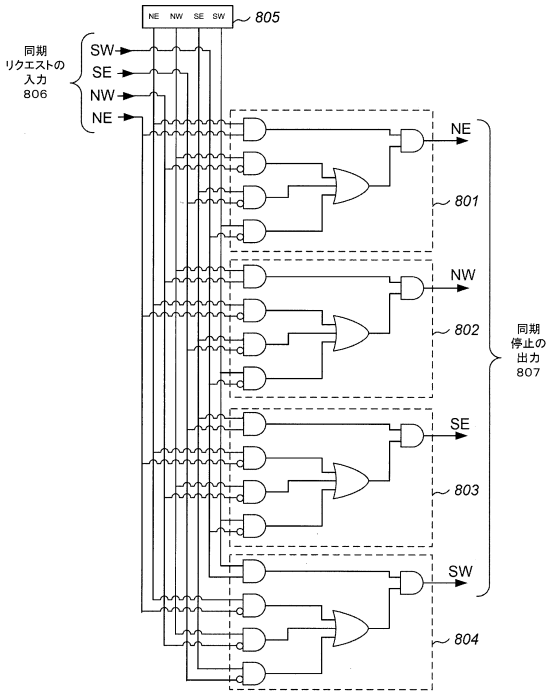

FIG. 8

【図９】

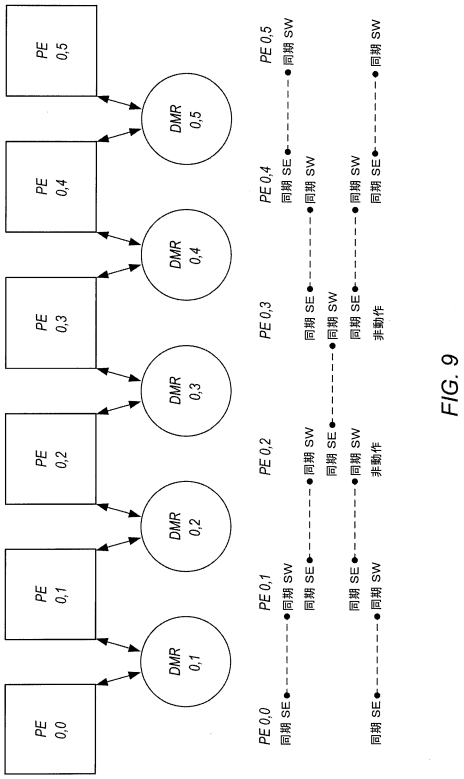

FIG. 9

【図１０】

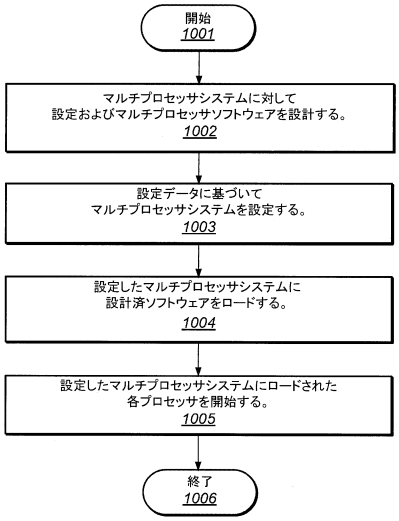

FIG. 10

【図１１】

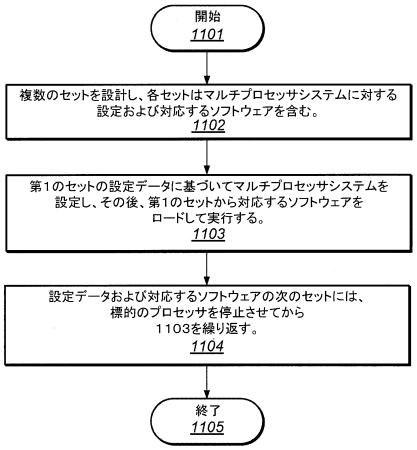

FIG. 11

【図１２】

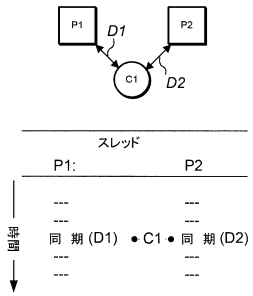

FIG. 12

【図１３】

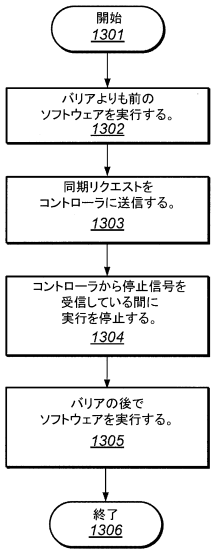

FIG. 13

【図１４】

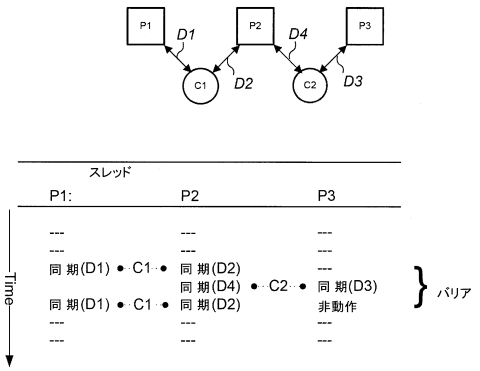

FIG. 14

【図１５】

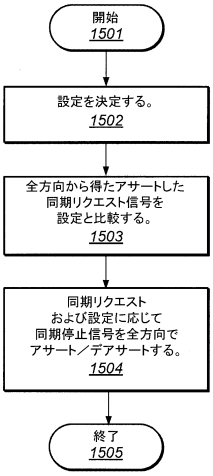

FIG. 15

【図１６】

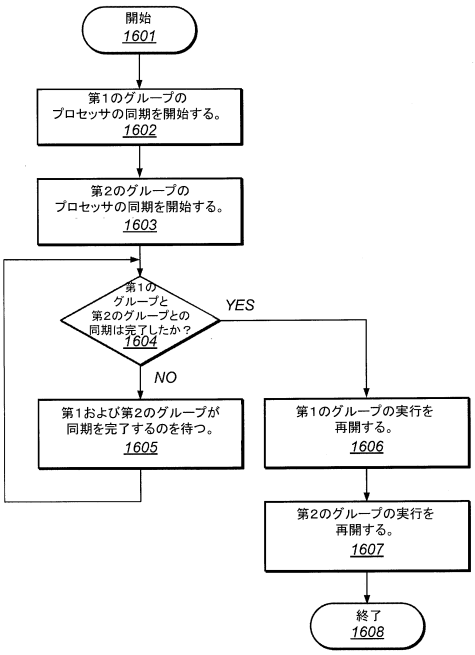

FIG. 16

【図１７】

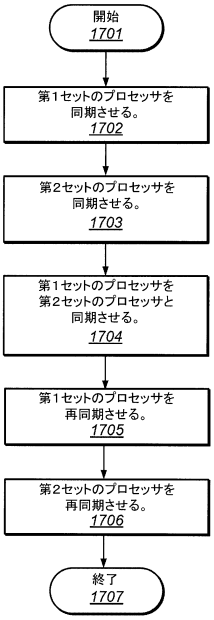

FIG. 17

【図１８】

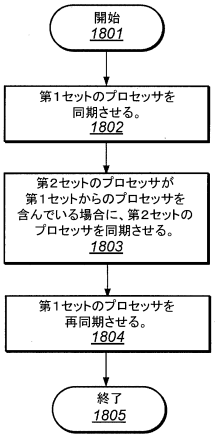

FIG. 18

【図１９】

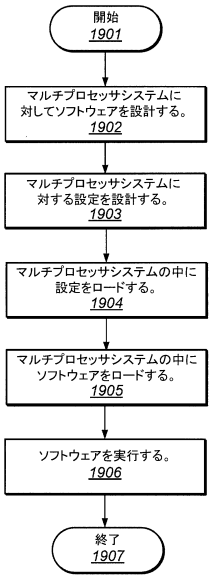
開始
1901
マルチプロセッサシステムに対してソフトウェアを設計する。
1902
マルチプロセッサシステムに対する設定を設計する。
1903
マルチプロセッサシステムの中に設定をロードする。
1904
マルチプロセッサシステムの中にソフトウェアをロードする。
1905
ソフトウェアを実行する。
1906
終了
1907

FIG. 19

フロントページの続き

(72)発明者　マリク，アフザル・エム
アメリカ合衆国・７８７２６・テキサス州・オースティン・ピークストン　ドライブ・１０２０５
(72)発明者　フォークナー，ケネス・アール
アメリカ合衆国・７８７３９・テキサス州・オースティン・ロスト　カヴァーン　コーヴ・３８０９
(72)発明者　ソルカ，マイケル・ビイ
アメリカ合衆国・７８７３１・テキサス州・オースティン・キャット　マウンテン　コーヴ・６２０９

審査官　漆原　孝治

(56)参考文献　米国特許出願公開第２０１２／０１３７１１９（ＵＳ，Ａ１）
特表２００５－５３１０８９（ＪＰ，Ａ）
特開平０２－２３８５５３（ＪＰ，Ａ）
特開平０２－２６４３５２（ＪＰ，Ａ）
特開平０７－２００４８６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１５／１７３
Ｇ０６Ｆ　　９／５２