(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5270529号
(P5270529)
(45)発行日　平成25年8月21日(2013.8.21)　(24)登録日　平成25年5月17日(2013.5.17)

(51)Int.Cl.　　F I
H03K 19/177 (2006.01)　　H03K 19/177
G06F 9/45 (2006.01)　　G06F 9/44 322F

請求項の数 29　(全 25 頁)

(21)出願番号　特願2009-500674 (P2009-500674)
(86)(22)出願日　平成19年3月19日 (2007.3.19)
(65)公表番号　特表2009-530924 (P2009-530924A)
(43)公表日　平成21年8月27日 (2009.8.27)
(86)国際出願番号　PCT/BE2007/000027
(87)国際公開番号　WO2007/106959
(87)国際公開日　平成19年9月27日 (2007.9.27)
審査請求日　平成21年9月28日 (2009.9.28)
(31)優先権主張番号　0605349.0
(32)優先日　平成18年3月17日 (2006.3.17)
(33)優先権主張国　英国 (GB)

(73)特許権者　591060898
アイメック
ＩＭＥＣ
ベルギー、ベー－３００１ルーヴァン、カペルドリーフ７５番
(74)代理人　100101454
弁理士　山田　卓二
(74)代理人　100081422
弁理士　田中　光雄
(74)代理人　100125874
弁理士　川端　純市
(72)発明者　アンドレアス・カンシュタイン
ドイツ連邦共和国デー－８０９９７ミュンヘン、レヒェルシュトラーセ５６番、ミューラー－ブッケ内

最終頁に続く

(54)【発明の名称】再構成可能なマルチ処理粗粒アレイ

(57)【特許請求の範囲】
【請求項１】
マルチ処理法で少なくとも２つの処理スレッドを同時に処理するように調整された粗粒再構成可能信号処理デバイスにおいて、
データに関してワードレベル若しくはサブワードレベルの動作を実行できる複数のファンクションユニットであって、少なくとも２つの異なるタイプのファンクションユニットを含む、複数のファンクションユニットと、
アプリケーションコードを格納するデータストレージと、
上記複数のファンクションユニットを相互接続するためのルーティングリソースであって、動的にスイッチされ得る複数の相互接続構成をサポートし、少なくとも一つの上記相互接続構成が上記複数のファンクションユニットを夫々所定のトポロジを備える少なくとも２つの非オーバーラップのパーティションの中に相互接続し、上記パーティションの各々若しくはパーティションの組み合わせが上記処理スレッドの夫々一つを処理するように構成されている、ルーティングリソースであり、上記アプリケーションコード内の所定のポイントで上記相互接続構成間で動的にスイッチするように調整されている、ルーティングリソースと、
上記粗粒再構成可能信号処理デバイス内に格納される複数のコンフィグレーションであって、動作を選択してルーティングリソースを制御することにより上記粗粒再構成可能信号処理デバイスの振る舞いを制御する、複数のコンフィグレーションと、
少なくとも２つの制御モジュールであって、個々の制御モジュールが制御のために上記

パーティションの一つに割り当てられている、少なくとも２つの制御モジュールと
を含む粗粒再構成可能信号処理デバイス。
【請求項２】
更に、複数のデータストレージを含み、
上記ルーティングリソースは、上記複数のファンクションユニットと上記複数のデータストレージを相互接続することを特徴とする請求項１に記載の粗粒再構成可能信号処理デバイス。
【請求項３】
上記アプリケーションコードは少なくとも２つの処理スレッドを含む処理を規定し、上記パーティションにより実行される
ことを特徴とする請求項１又は２に記載の粗粒再構成可能信号処理デバイス。
【請求項４】
上記ルーティングリソースが、稼働中のアプリケーションのデータ内容に依存して相互接続構成を動的にスイッチするように調整されている
ことを特徴とする請求項１乃至３のうちのいずれか一に記載の粗粒再構成可能信号処理デバイス。
【請求項５】
上記ルーティングリソースが、多重化及び／又は逆多重化回路を含む
ことを特徴とする請求項４に記載の粗粒再構成可能信号処理デバイス。
【請求項６】
クロックを有し、
上記多重化及び／又は逆多重化回路が、相互接続構成を動的にスイッチするための設定により構成されるように調整され、上記設定がクロック周期毎に変更し得る
ことを特徴とする請求項５に記載の粗粒再構成可能信号処理デバイス。
【請求項７】
更に、複数のファンクションユニット間で共有される少なくとも一つのグローバルストレージを含む
ことを特徴とする請求項１乃至６のうちのいずれか一に記載の粗粒再構成可能信号処理デバイス。
【請求項８】
上記相互接続構成の少なくとも別の一つが、上記複数のファンクションユニットをシングル制御モジュールの制御下にあるシングルパーティション内に相互接続する
ことを特徴とする請求項１乃至７のうちのいずれか一に記載の粗粒再構成可能信号処理デバイス。
【請求項９】
少なくとも２つの上記制御モジュールの少なくとも一つが、シングルパーティションを伴う相互接続構成で利用するグローバル制御ユニットの一部である
ことを特徴とする請求項８に記載の粗粒再構成可能信号処理デバイス。
【請求項１０】
シングルパーティションを伴う少なくとも一つの相互接続構成にて、上記制御モジュールの少なくとも一つが、少なくとも一つの他の制御モジュールに追随させることによって、全ての上記ファンクションユニットの制御信号を駆動する
ことを特徴とする請求項９に記載の粗粒再構成可能信号処理デバイス。
【請求項１１】
利用される上記制御モジュール内で、複数の非オーバーラップパーティションを伴う相互接続構成の上記パーティションに割り当てられる上記制御モジュールの少なくとも一部を、シングルパーティションを伴う相互接続構成にて、再利用するように調整されている
ことを特徴とする請求項１乃至１０のうちのいずれか一に記載の粗粒再構成可能信号処理デバイス。
【請求項１２】

請求項１乃至１１のうちのいずれか一に記載の粗粒再構成可能信号処理デバイスでアプリケーションを実行する方法であって、
最初の制御モジュールの制御下でシングル処理スレッドとして上記粗粒再構成可能信号処理デバイス上で上記アプリケーションを実行するステップと、
上記粗粒再構成可能信号処理デバイスを少なくとも２つの非オーバーラップパーティションを伴うデバイスに動的にスイッチするステップと、
上記アプリケーションの一部を少なくとも２つの処理スレッドに分割するステップと
を含み、
個々の処理スレッドは、上記パーティションの一つ上で独立の処理スレッドとして同時に実行され、
個々のパーティションは、独立の制御モジュールにより制御される
ことを特徴とする方法。
【請求項１３】
上記粗粒再構成可能信号処理デバイスを少なくとも２つのパーティションを伴うデバイスにスイッチするステップが、アプリケーションを決定するアプリケーションコード内の第１の命令により決定される
ことを特徴とする請求項１２に記載の方法。
【請求項１４】
上記第１の命令が、上記独立の処理スレッドの各々の上記命令の開始アドレスを含む
ことを特徴とする請求項１３に記載の方法。
【請求項１５】
更に、
上記粗粒再構成可能信号処理デバイスをシングルパーティションを伴うデバイスに動的にスイッチし戻すステップと、
上記独立の制御モジュールを同期化させるステップと、
上記アプリケーションの上記少なくとも２つのスレッドをシングル処理スレッドに結合するステップと
を含み、
上記シングル処理スレッドは、上記同期化された制御モジュールの制御下で上記シングルパーティション上で処理スレッドとして実行される
ことを特徴とする請求項１２乃至１４のうちのいずれか一に記載の方法。
【請求項１６】
上記粗粒再構成可能信号処理デバイスをシングルパーティションを伴うデバイスに動的にスイッチし戻すステップが、アプリケーションを決定するアプリケーションコード内の第２の命令により決定される
ことを特徴とする請求項１５に記載の方法。
【請求項１７】
上記第２の命令が、上記シングル処理スレッドとして実行される上記命令の開始アドレスを含む
ことを特徴とする請求項１６に記載の方法。
【請求項１８】
シングル処理スレッドとして上記アプリケーションを実行するとき、上記シングル制御モジュールが、上記独立の制御モジュールの少なくとも一つを再利用する
ことを特徴とする請求項１２乃至１７のうちのいずれか一に記載の方法。
【請求項１９】
シングルパーティションを伴う相互接続構成にて、上記独立の制御モジュールの一つが、他の制御モジュールに追随させることによって、全ての上記ファンクションユニットの制御信号を駆動する
ことを特徴とする請求項１２乃至１４のうちのいずれか一に記載の方法。
【請求項２０】

請求項１乃至１１のうちのいずれか一に記載の粗粒再構成可能信号処理デバイス上で実行されるコンパイルコードを取得するためにアプリケーションソースコードをコンパイルするための方法であって、
アプリケーションソースコードを入力するステップと、
上記アプリケーションソースコードからコンパイルコードを生成するステップと
を含み、
上記コンパイルコードを生成することが、コンパイルコード内に、マルチプル処理スレッドを同時に実行し且つ上記処理スレッドを同時に実行することを開始するように上記粗粒再構成可能信号処理デバイスを構成する第１の命令と、上記マルチプル処理スレッドの最後のものがその命令をデコードするときに、上記粗粒再構成可能信号処理デバイスが統合モードでの実行を継続するべく構成されるように、上記マルチプル処理スレッドの同時実行を終了させる第２の命令とを含む
ことを特徴とする方法。
【請求項２１】
上記粗粒再構成可能信号処理デバイスのアーキテクチャ記述を設けるステップを更に含み、
上記アーキテクチャ記述が、パーティションを形成するファンクションユニットの所定の相互接続構成の記述を含む
ことを特徴とする請求項２０に記載の方法。
【請求項２２】
アーキテクチャ記述を設けるステップが、パーティション毎に独立の制御モジュールを設けることを含む
ことを特徴とする請求項２１に記載の方法。
【請求項２３】
上記第１の命令が、上記マルチプル処理スレッドの各々の命令の開始アドレスを含む
ことを特徴とする請求項２０乃至２２のうちのいずれか一に記載の方法。
【請求項２４】
上記第２の命令が、上記マルチプル処理スレッドの実行の後に統合モードで実行される命令の開始アドレスを含む
ことを特徴とする請求項２０乃至２３のうちのいずれか一に記載の方法。
【請求項２５】
上記コンパイルコードを生成するステップが、
上記アプリケーションソースコードを分割し、これによりコード分割を生成するステップと、
どのモードで、及びどのパーティションで、コード分割が実行されるか分類するステップと、
上記コード分割の各々を独立してコンパイルするステップと、
上記コンパイルされたコード分割をシングル実行可能コードファイル内にリンクするステップと
を含む請求項２０乃至２４のうちのいずれか一に記載の方法。
【請求項２６】
アプリケーションを請求項１乃至１１のうちのいずれか一に記載の粗粒再構成可能信号処理デバイス上で実行されるように調整するための方法であって、
上記アプリケーションの種々の分割の探査を行うステップを含み、
上記探査を行うステップが、上記粗粒再構成可能信号処理デバイスの種々の相互接続構成を探査するため、上記粗粒再構成可能信号処理デバイスのアーキテクチャ記述のインスタンスを変更することを含む
方法。
【請求項２７】
上記粗粒再構成可能信号処理デバイスの相互接続構成を探査するステップが、シングル

制御モジュールの制御下にあるシングルパーティションを有する相互接続構成と、独立の制御モジュールの制御下に各々がある少なくとも２つのパーティションを有する相互接続構成との間を、動的にスイッチすることを探査することを含む
ことを特徴とする請求項２６に記載の方法。
【請求項２８】
請求項１乃至１１のうちのいずれか一に記載の粗粒再構成可能信号処理デバイス上で稼動するときに、請求項１２乃至１９のうちのいずれか一に記載の方法を実行するための、コンピュータプログラム。
【請求項２９】
請求項２８に記載のコンピュータプログラムを格納する機械読み取り可能テータ格納デバイス。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、マルチプロセス若しくはマルチスレッド方式で少なくとも２つのスレッドを同時に処理するのに適用される信号処理デバイス、その信号処理デバイスでアプリケーションを実行する方法、その信号処理デバイスで実行可能なコンパイルされたコードを取得するためにアプリケーションソースコードをコンパイルする方法、アプリケーションを調整してその信号処理デバイスで実行させる方法、その信号処理デバイスでアプリケーションを実行する方法のいずれかを実行するコンピュータプログラムプロダクト、そのコンピュータプログラムプロダクトを格納する機械読み取り可能データストレージデバイス、及び、ローカルエリア若しくはワイドエリアの遠隔通信ネットワークにおけるそのコンピュータプログラムプロダクトの転送に、関する。
【背景技術】
【０００２】
今日、標準的埋込式システムは、実行時間時のビデオエンコード／デコードなどのタスクを実施する高いパフォーマンスを要求する。その標準的埋込式システムは、軽量バッテリを利用して何時間も更には何日間も稼動できるように、あまりエネルギを消費しないものであるべきである。それは、一つのデバイス内に多重アプリケーション若しくはスタンダードを十分に統合できるような可撓性を備えるべきである。それは、実質的には複雑さが増大するにも拘わらず、短い商品化時間で設計され確認されねばならない。設計者は奮闘してこれらの挑戦に応じるのであるが、アーキテクチャと設計方法論の両方の革新が要求される。
【非特許文献１】Ｇ．Ｍ．Ａｍｄａｈｌによる“Ｖａｌｉｄｉｔｙ ｏｆ ｔｈｅ ｓｉｎｇｌｅ ｐｒｏｃｅｓｓｏｒ ａｐｐｒｏａｃｈ ｔｏ ａｃｈｉｅｖｅ ｌａｒｇｅ－ｓｃａｌｅ ｃｏｍｐｕｔｉｎｇ ｃａｐａｂｉｌｉｔｉｅｓ”，Ｐｒｏｃ．ＡＦＩＰＳ Ｓｐｒｉｎｇ Ｊｏｉｎｔ Ｃｏｍｐｕｔｅｒ Ｃｏｎｆ．３０，１９６７ Ｐａｇｅ（ｓ）：４８３－４８５
【非特許文献２】Ｉｗａｔａらによる“Ｅｘｐｌｏｉｔｉｎｇ Ｃｏａｒｓｅ－Ｇｒａｉｎ Ｐａｒａｌｌｅｌｉｓｍ ｉｎ ｔｈｅ ＭＰＥＧ－２ Ａｌｇｏｒｉｔｈｍ”，Ｓｔａｎｄｆｏｒｄ Ｕｎｉｖｅｒｓｉｔｙ Ｃｏｍｐｕｔｅｒ Ｓｙｓｔｅｍｓ Ｌａｂ Ｔｅｃｈｎｉｃａｌ Ｒｅｐｏｒｔ ＣＳＬ－ＴＲ－９８－７７１，Ｓｅｐｔｅｍｂｅｒ １９９８
【発明の開示】
【発明が解決しようとする課題】
【０００３】
粗粒再構成可能（Ｃｏａｒｓｅ－ｇｒａｉｎｅｄ ｒｅｃｏｎｆｉｇｕｒａｂｌｅ）アーキテクチャ（ＣＧＲＡ）は、上記挑戦に応じる潜在的な候補者として出現している。近年、多数のデザインが提案されている。これらのアーキテクチャは、数十から数百のファンクションユニット（ＦＵ）を含むことがしばしばであり、該ファンクションユニットは

、通常のＦＰＧＡで見られるビットレベルのオペレーションの代わりにワードレベルのオペレーションを実行できる。この粗粒により、ＦＰＧＡと比べて、遅延、エリア、電力、及び構成が、大きく減少する。一方で、従来の“粗粒の”プログラム可能プロセッサと比較すると、それらの大規模な計算機資源によりそれらは高度な並行処理及び効率を達成できる。しかしながら、主としてそのような複雑なアークテクチャのプログラミングの困難性のために、現存のＣＧＲＡは未だ広範には採用されていない。

【課題を解決するための手段】

【０００４】

第１の形態では、本発明は、マルチ処理法で少なくとも２つの処理スレッドを同時に処理するように調整された信号処理デバイスに関する。信号処理デバイスは、データに関してワードレベル若しくはサブワードレベルの動作を実行できる複数のファンクションユニットと、上記複数のファンクションユニットを相互接続するためのルーティングリソースであって、動的にスイッチされ得る複数の相互接続構成をサポートし、少なくとも一つの上記相互接続構成が上記複数のファンクションユニットを夫々所定のトポロジを備える少なくとも２つの非オーバーラップの処理ユニットの中に相互接続し、上記処理ユニットの各々が上記処理スレッドの夫々一つを処理するように構成されている、ルーティングリソースとを含む。上記相互接続構成の他方は、上記複数のファンクションユニットをシングル処理ユニットの中に相互接続できる。信号処理デバイスは、少なくとも２つの制御モジュールであって、個々の制御モジュールが制御のために上記処理ユニットの一つに割り当てられている、少なくとも２つの制御モジュールを、更に含む。ワードレベル若しくはサブワードレベルの動作により、非ビットレベルの動作を意味している。

【０００５】

上記ファンクションユニットが、少なくとも一つのファンクションユニットを含む所定の／静的なグルーピングでグループ化され、そのグルーピングの各々が処理ユニットを規定するということは、本発明の一つの形態である。

【０００６】

制御モジュールは、命令フェッチユニット及び制御ユニットを含んでもよい。上記制御モジュールは、それらに割り当てられた処理ユニット内部でワードレベル若しくはサブワードレベル（非ビットレベル）動作を制御するように調整される。

【０００７】

本発明の実施形態では、上記制御モジュールは、プログラムカウンタ上で動作（インクリメント、変更）を行い得る。相応のデバッギングをサポートしてもよい。

【０００８】

本発明の実施形態では、複数のデータストレージが設けられ、上記ルーティングリソースは、上記複数のファンクションユニットと上記複数のデータストレージを相互接続する。データストレージはレジスタでもよい。データストレージはレジスタでもよい。データストレージは、上記ファンクションユニット間で共有されてもよい。本発明の実施形態では、一つのデータストレージが個々の処理ユニットのために設けられてもよい。

【０００９】

本発明の実施形態に係る信号処理デバイスは、アプリケーションコードが格納されるデータストレージを含んでもよく、上記アプリケーションコードは少なくとも２つの処理スレッドを含む処理を規定し、上記処理ユニットにより実行される。上記ルーティングリソースは、上記アプリケーションコード内の所定のポイントで上記相互接続構成間で動的にスイッチするように調整されていてもよい。

【００１０】

本発明の実施形態に係る信号処理デバイスでは、上記ルーティングリソースが、稼働中のアプリケーションのデータ内容に依存して相互接続構成を動的にスイッチするように調整されていてもよい。そのデータ内容は、例えば、スレッドの処理ユニットファンクションがマップされ得るパラメータファイル記述、若しくは、上記ファンクションユニットの一つのデータストレージ内の一若しくはそれ以上のビットのデータであってもよい。上記

ルーティングリソースが、多重化及び／又は逆多重化回路を含んでもよい。信号処理デバイスは、クロックを有し、上記多重化及び／又は逆多重化回路が、相互接続構成を動的にスイッチするための適切な設定により構成されるように調整され、上記設定がクロック周期毎に変更し得るようにしてもよい。

【００１１】

本発明の実施形態に係る信号処理デバイスは、更に、複数のファンクションユニット間で共有される少なくとも一つのグローバルストレージを含んでもよい。

【００１２】

本発明の実施形態に係る信号処理デバイスは、少なくとも２つの異なるタイプのファンクションユニットを含んでもよい。

【００１３】

本発明の実施形態に係る信号処理デバイスでは、上記相互接続構成の少なくとも別の一つが、上記複数のファンクションユニットをシングル制御モジュールの制御下にあるシングル処理ユニット内に相互接続してもよい。

【００１４】

本発明の実施形態に係る信号処理デバイスでは、少なくとも２つの上記制御モジュールの少なくとも一つが、シングル処理ユニットを伴う相互接続構成で利用するグローバル制御ユニットの一部であってもよい。シングル処理ユニットを伴う少なくとも一つの相互接続構成にて、上記制御モジュールの少なくとも一つが、少なくとも一つの他の制御モジュールに追随させることによって、全ての上記ファンクションユニットの制御信号を駆動してもよい。

【００１５】

本発明の実施形態に係る信号処理デバイスは、利用される上記制御モジュール内で複数の非オーバーラップ処理ユニットを伴う相互接続構成の上記処理ユニットに割り当てられる上記制御モジュールの少なくとも一部を、シングル処理ユニットを伴う相互接続構成にて、再利用するように調整されていてもよい。

【００１６】

第２の形態では、本発明は、上述のような信号処理デバイスで少なくとも一つのアプリケーションを実行する方法に関する。アプリケーションは、シングル制御モジュールの制御下において、シングル処理スレッドとして信号処理デバイス上で通常実行される。

【００１７】

シングルスレッドアプローチとマルチスレッドアプローチの間でのスイッチングが適用される、少なくとも一つのアプリケーションを実行するための方法を示すことは、本発明の一つの形態であり、そのアプリケーションの一部は部分分割され、各々の部分は、予め定められた処理ユニットの一つの上で、独立の処理スレッドとして実行される。本発明の実施形態に係る方法は、上記信号処理デバイスを少なくとも２つの非オーバーラップ処理ユニットを伴うデバイスに動的にスイッチするステップと、上記アプリケーションの一部を少なくとも２つの処理スレッドに分割するステップとを含み、個々の処理スレッドは、上記処理ユニットの一つ上で独立の処理スレッドとして同時に実行され、個々の処理ユニットは、独立の制御モジュールにより制御される。

【００１８】

アプリケーションを実行するこのシングル若しくはマルチスレッドスイッチングは、信号処理デバイスの構成機能、特に統合モードでそれを操作する機能によりサポートされ、デバイスの全てのファンクションユニットは、制御の一つのスレッドで及び分割モードで動作し、シングル処理ユニット内部の全てのファンクションユニットは、制御の一つのスレッドで動作し、処理ユニット自身は制御の種々のスレッドで同時に動作する。

【００１９】

つまりそれ故に、信号処理デバイスは、複数のパーティションを含み、夫々は処理スレッドを独立して稼動できる。

【００２０】

その結果、個々の処理ユニット内部で、命令フローは、例えばコード内の分岐のために、別の処理ユニット内の命令フローから独立して、変化し得る。

【００２１】

複数のファンクションユニットを伴う信号処理デバイスを利用することにより、命令レベルの並行処理が可能であり、これらのファンクションユニットを構成して特徴的な処理ユニットにグルーピングすることにより、スレッドレベルの並行処理が可能である。グルーピングは動的に変更され得るので、マルチコアアプローチによるよりも、より可撓性を得られ得る。

【００２２】

上記機能は、上記２つ若しくはそれ以上の制御モジュールを設けることで実現される。上記制御モジュールの各々は制御のシングルスレッドを実行できる。

【００２３】

本発明の実施形態によると、上記信号処理デバイスを少なくとも２つの処理ユニットを伴うデバイスにスイッチするステップが、アプリケーションを決定するアプリケーションコード内の第１の命令により決定されてもよい。上記第１の命令が、上記独立の処理スレッドの各々の上記命令の開始アドレスを含んでもよい。開始アドレスは、命令が見出される位置の指標である。それは位置への直接言及でも位置へのポインタでもよい。ここで位置とは、例えば、レジスタ内部やデータストレージ内部である。

【００２４】

本発明の実施形態に係る方法は、更に、上記信号処理デバイスをシングル処理ユニットを伴うデバイスに動的にスイッチし戻すステップと、上記独立の制御モジュールを同期化させるステップと、上記アプリケーションの上記少なくとも２つのスレッドをシングル処理ユニットに結合するステップとを含み、上記シングル処理スレッドは、上記同期化された制御モジュールの制御下で上記シングル処理ユニット上で処理スレッドとして実行される。上記信号処理デバイスをシングル処理ユニットを伴うデバイスに動的にスイッチし戻すステップが、アプリケーションを決定するアプリケーションコード内の第２の命令により決定されてもよい。上記第２の命令が、上記シングル処理スレッドとして実行される上記命令の開始アドレスを含んでもよい。

【００２５】

本発明の実施形態によると、シングル処理スレッドとして上記アプリケーションを実行するとき、上記シングル制御モジュールが、上記独立の制御モジュールの少なくとも一つを再利用してもよい。

【００２６】

本発明の実施形態によると、シングル処理ユニットを伴う相互接続構成にて、上記独立の制御モジュールの一つが、他の制御モジュールに追随させることによって、全ての上記ファンクションユニットの制御信号を駆動してもよい。

【００２７】

第１の構成のパーティション間でのスイッチングにおいて、パーティションの幾つかはシングルスレッドとして共に実行され、他のパーティションは別のスレッド及び第２の構成のパーティションとして共に実行される方法にまで、このシングル若しくはマルチスレッドスイッチング方法が容易に拡張可能であることは、明白である。

【００２８】

この汎用のスイッチング方法は、デバイスの静的な所定の構成によりサポートされる、デバイスの動的なランタイムの再構成と、言い換えることができる。

【００２９】

上記信号処理デバイス上で少なくとも一つのアプリケーションを実行するための処理中の信号処理デバイスを動的に再構成するために、更に方法が提示され、上記信号処理デバイスは、データに関してワードレベル若しくはサブワードレベル（非ビットレベル）動作を実行することができる複数のファンクションユニットを含み、上記ファンクションユニットは、一つ又はそれ以上非オーバーラップの処理ユニットにグループ化され、ルーティ

ングリソースは上記ファンクションユニットを相互接続し、上記アプリケーションは、少なくとも一部が少なくとも部分的に同時に実行可能である複数のスレッドとして設けられ、
上記方法は、
上記処理ユニットに対して一つ若しくはそれ以上のスレッドの第１の割り当てを行うことにより、上記計算の信号処理デバイスを構成するステップと、
上記構成後、上記一つ若しくはそれ以上のスレッドを同時に実行するステップであって、上記の実行されるスレッドの各々が上記第１の割り当てに係る上記処理ユニットの一つ若しくはそれ以上のものの上で実行されるステップと、
上記実行を終了するステップと、
上記処理ユニットに対して一つ若しくはそれ以上のスレッドの第２の、別の割り当てを行うことにより、上記信号処理デバイスを構成するステップと、
上記構成後、上記一つ若しくはそれ以上のスレッドを同時に実行するステップであって、上記の実行されるスレッドの各々が上記第２の割り当てに係る上記処理ユニットの一つ若しくはそれ以上のものの上で実行されるステップと
を含む。

【００３０】
大きいオーバーヘッドを回避するため、シングルスレッド処理のために用いられている制御モジュールは、個々の処理ユニットのために利用可能である、制御モジュールの少なくとも一部（更には全部）を再利用してもよい。

【００３１】
一つの実施形態では、この再利用は、パーティションの制御モジュールを同期化することにより実現され得る。つまり、個々の上記要素が同じ入力を利用しそれらの各々の出力を割り当てられたパーティションに分配するということである。

【００３２】
例えば、上記ファンクションユニットの間に多重化及び／又は逆多重化回路を設けることで、それらは柔軟に接続され得るということも注意すべきである。上記重化及び／又は逆多重化回路を設定することにより動的な再構成が実現され得る。この設定は周期から周期へと変更し得る。

【００３３】
本発明は更に、上述の信号処理デバイス上で実行されるコンパイルコードを取得するためにアプリケーションソースコードをコンパイルするための方法であって、特に、コードを分割するソースコードレベルに含まれる命令、及び、実行モード（例えば、統合及び分割モード）のスイッチングのため、コンパイルコード内に自動的に含まれる命令に対するものである。

【００３４】
この形態では、本発明は、上述の信号処理デバイス上で実行されるコンパイルコードを取得するためにアプリケーションソースコードをコンパイルするための方法に関する。方法は、アプリケーションソースコードを入力するステップと、上記アプリケーションソースコードからコンパイルコードを生成するステップとを含む。上記コンパイルコードを生成することが、コンパイルコード内に、マルチプル処理スレッドを同時に実行し且つ上記処理スレッドを同時に実行することを開始するように上記信号処理デバイスを構成する第１の命令と、上記マルチプル処理スレッドの最後のものがその命令をデコードするときに、上記信号処理デバイスが統合モードでの実行を継続するべく構成されるように、上記マルチプル処理スレッドの同時実行を終了させる第２の命令とを含む。この故に、コード自身により（動的に）構成は為され得る。

【００３５】
本発明の更なる形態では、上記信号処理デバイスのアーキテクチャ記述が設けられ、ファンクションユニットのグルーピングの記述を含む。実際、そのような信号処理デバイスは、一般的なテンプレートのインスタンスとして通常生成される。発明を記載すると、一

般的なテンプレートは、一つ又はそれ以上の処理ユニットを形成してグループ毎に制御モジュールを設けるように、ファンクションユニットをグループ化し得ることを含む。

【００３６】

本発明の実施形態では、方法は、上記信号処理デバイスのアーキテクチャ記述を設けるステップを更に含み、上記アーキテクチャ記述が、処理ユニットを形成するファンクションユニットの所定の相互接続構成の記述を含む。アーキテクチャ記述を設けるステップが、処理ユニット毎に独立の制御モジュールを設けることを含み得る。

【００３７】

本発明の更なる形態は、コンパイル方法を設けることであり、アプリケーションソースコード及び上述のアーキテクチャ記述を入力するステップと、コンパイルコードを生成するステップとを含み、マルチプル処理スレッドを実行し且つ上記処理スレッドを実行することを開始するように上記信号処理デバイスを構成する（例えば、分岐を意味する）第１の命令と、上記マルチプルスレッドの実行を終了させる（例えば、結合を意味する）第２の命令とを含む。特に、上記第２の命令は、上記スレッドの最後のものがその命令をデコードするときに、上記信号処理デバイスが統合モードで継続するべく構成される、というものである。

【００３８】

本発明の実施形態では、上記第１の命令が、上記マルチプル処理スレッドの各々の命令の開始アドレスを含んでもよい。本発明の実施形態では、上記第２の命令が、上記マルチプル処理スレッドの実行の後に統合モードで実行される命令の開始アドレスを含んでもよい。

【００３９】

本発明の実施形態では、上記コンパイルコードを生成するステップが、上記アプリケーションソースコードを分割し、これによりコード分割を生成するステップと、どのモードで、及びどの処理ユニットで、コード分割が実行されるか分類するステップと、上記コード分割の各々を独立してコンパイルするステップと、上記コンパイルされたコード分割をシングル実行可能コードファイル内にリンクするステップとを含む。

【００４０】

本発明の更に別の形態は、コンパイル方法を設けることであって、該方法は、アプリケーションソースコードと上述のアーキテクチャ記述を入力するステップと、コードを分割しコードがどのように（統合／分割モード）どこで（どの処理要素で）実行され、コードの各々の独立したコンパイルを分離しコンパイルされたコードを実行可能なシングルにリンクするのかを、分類するステップとを含む。

【００４１】

本発明は更に環境の調整に関し、アプリケーションに対して、種々の分割の探査が行われ、上記環境の調整は、上記信号処理デバイスの種々の構成を探査するため、上記信号処理デバイスのアーキテクチャ記述のインスタンスを変更し得ることも含む。

【発明の効果】

【００４２】

本発明の特定の実施形態は、添付の請求項に設定される。独立請求項からの特徴は、従属請求項の特徴と、更に、適宜の他の独立請求項の特徴と組み合わせることができ、請求項に明白に記載されるものに限るものではない。

【００４３】

本発明の上記の及び他の特性、特徴及び利点は、例示に過ぎないが本発明の原理を示す添付の図面と関連させることで、以下の詳細な説明から明白となる。この記載は、例示のためのものであり、本発明の範囲を限定することなく為されている。以下に引用される符号は添付の図面を参照する。

【発明を実施するための最良の形態】

【００４４】

本発明は、図面を参照しつつ特定の実施形態によって説明されているが、本発明はそれ

らに限定されるものではない。記載されている図面は概要に過ぎず限定的なものではない。図面では、例示の目的のため、幾つかの要素のサイズは誇張されており尺度によっては描かれていない。寸法及び相対的寸法は、本発明を実施するための現実の縮小に対応していない。

【００４５】

更に、明細書や請求項における、第１の、第２の、第３の、などの用語は、類似の要素間で区別をするのに利用しており、一連の若しくは登場順の順序を記載するのに利用しているというものでもない。当然のことながら、ここで利用する用語は適切な状況下では相互に交換可能であり、明細書等に記載の発明の実施形態は明細書等に記載の若しくは例示の順序とは異なる順序で、動作可能である。

【００４６】

更に、明細書及び図面内の、頂部、底部、を超えて、の下で、等の用語は、記載のために利用するものであり、必ずしも相対的位置のために利用するものではない。当然のことながら、かように利用する用語は適切な状況下では相互に交換可能であり、明細書等に記載の発明の実施形態は明細書等に記載の若しくは例示の向きとは異なる向きで、動作可能である。

【００４７】

注目すべきことは、請求項で利用される“含む”という用語は、それ以降に掲げられる手段に対して限定するように解釈されるべきではないということである。それは他の要素やステップを排除するものではない。それは、記載の通り、規定の特徴、整数、ステップ若しくは要素の存在を特定するように解釈されるべきであり、一つ又はそれ以上の別の特徴、整数、ステップ若しくは要素、又はそれらの組み合わせの存在若しくは付加を排除するものではない。従って、手段Ａ及びＢを含むデバイスという表記の範囲は、要素Ａ及びＢのみから構成されるデバイスに限定されるべきではない。

【００４８】

同様に注目すべきは、請求項でまた利用される“結合された”という用語は、直接的な結合のみに限定して解釈されるべきではないということである。従って、“デバイスＢに結合するデバイスＡ”という表記の範囲は、デバイスＡのアウトプットがデバイスＢのインプットに直接に結合するようなデバイスやシステムに限定されるべきではない。Ａｎｏアウトプットとのインプットの間に、他のデバイスや手段を含むパスであってもよいパスが存在することを意味する。

【００４９】

粗粒再構成可能アーキテクチャ（ＣＧＲＡ）、ＡＤＲＥＳ（動的再構成可能埋込システム）は周知であり、ベルギー国ルヴェン市のアンテルユニヴェシテール・ミクロ－エレクトロニカ・サントリュム・ヴェー・ゼッド・ドゥブルヴェにより製造されている。ＡＤＲＥＳは、現存のＣＧＲＡの問題を対処するものである。ＡＤＲＥＳアーキテクチャを参照しつつ本発明を説明する。しかしながら、これは限定を意図するものではない。本発明は他の適切な粗粒アレイアーキテクチャにも利用され得る。

【００５０】

ＡＤＲＥＳアーキテクチャは、データ経路連結の粗粒再構成可能マトリクスである。ＡＤＲＥＳアーキテクチャは、超長命令語（ＶＬＩＷ）デジタル信号プロセッサ（ＤＳＰ）を２－Ｄ粗粒異種再構成可能アレイ（ＣＧＡ）と組み合わせる、電力効率の良い可撓性のあるアーキテクチャテンプレートであり、ＶＬＩＷのデータ経路から延在するものである。ＶＬＩＷアーキテクチャは、サイクル毎に、単一の大きい“命令ワード”即ち“パケット”内にパックされる多重命令を実行し、簡素な規則正しい命令セットを利用する。ＶＬＩＷ　ＤＳＰは、命令レベル並行処理（ＩＬＰ）を活用してコントロールフローコードを効率よく実行する。多数のファンクションユニットを含むアレイは、高程度のループレベル並行処理（ＬＬＰ）を活用して、データフローループを加速する。アーキテクチャテンプレートにより、設計者はファンクションユニットの相互接続、タイプ及び数を特定できる。

【００５１】
このようにＡＤＲＥＳテンプレートは、同一の物理リソース上に２つのファンクションモードを設けることにより、超長命令ワード（ＶＬＩＷ）プロセッサ１１と粗粒アレイ１２を堅固に連結できる。これは、高パフォーマンス、低通信オーバーヘッド及びプログラミングの容易性などの、利点をもたらす。例えば、Ｃなどのプログラミング言語で書かれたアプリケーションは、ＡＤＲＥＳインスタントに即座にマップされ得る。ＡＤＲＥＳは、具体的アーキテクチャに代わるテンプレートである。アーキテクチャ探査により、よりよいアーキテクチャを発見すること若しくはドメイン特定アーキテクチャを設計することが可能になる。

【００５２】
ＡＤＲＥＳアレイは、具体的インスタントに代わる可撓性のあるテンプレートである。アーキテクチャ記述言語は、様々なＡＤＲＥＳインスタンスを記述するように開発された。スクリプトベースの技術により、設計者は、ターゲットのアーキテクチャに関する通信トポロジ、サポートされるオペレーションセット、リソースアロケーション及びタイミングのための、種々の値を記述することにより、容易に種々のインスタンスを生成できる。再ターゲット可能なシミュレータ及びコンパイラと共に、このツールーチェーンにより、アプリケーションドメイン特定プロセッサをアーキテクチャ探査し開発できる。テンプレートを利用してＡＤＲＥＳインスタンスが規定されるので、ＶＬＩＷ幅、アレイサイズ、相互接続トポロジ等は、利用のケースによって変動し得る。

【００５３】
ＡＤＲＥＳテンプレートは、計算機、記憶装置及びルータなどのリソースを含む、多数の基本的要素を含む。計算機のリソースは、制御信号により選択されるワードレベルオペレーションのセットを実行できるファンクションユニット（ＦＵｓ）１３である。レジスタファイル（ＲＦｓ）１４及びメモリブロック１５などのデータ記憶装置は、中間データを格納するのに利用され得る。ルータ・リソース１６は、ワイヤ、マルチプレクサ及びバスを含む。このようにＡＤＲＥＳインスタンスは、ファンクションユニット１３、レジスタ１５、レジスタファイル１４、及び、ファンクションユニット１３やレジスタファイル１４を連結するバスやマルチプレクサなどのルータ・リソース１６を含む。

【００５４】
基本的に、計算機のリソース（ＦＵｓ）１３と記憶装置のリソース（例えば、ＲＦｓ）はルータ・リソース１６により或るトポロジで連結され、ＡＤＲＥＳアレイのインスタンスを形成する。ＡＤＲＥＳ全体は、図１の点線で示されるように、ＶＬＩＷプロセッサ１１と再構成可能アレイ１２との、２つのファンクションモデルを有する。これらの２つのファンクションモデル１１、１２は、プロセッサ／共同プロセッサモデルのおかげでそれらの実行がオーバーラップすることが決して無いので、物理リソースをシェアできる。プロセッサは、ＶＬＩＷモードでもＣＧＡモードでも動作する。グローバルデータレジスタファイルＲＦ’１５は両方のモードで利用され、両方のモードの間のデータインタフェースとして機能し、統合コンパイルフローを有効にする。

【００５５】
ＶＬＩＷプロセッサ１１は、典型的なＶＬＩＷアーキテクチャ内として、複数のＦＵｓ１３と少なくとも一つのマルチポートレジスタファイルＲＦ’１５を含むが、この場合ＶＬＩＷプロセッサ１１は再構成可能アレイ１２の第１の行としても利用される。この第１の行の複数ＦＵｓ１３は、利用可能のポート数に拠るがメモリヒエラルキ１０に接続する。統一アーキテクチャのメモリへのデータアクセスは、これらのＦＵｓで利用可能なロード／ストア動作を介して為される。

【００５６】
コンパイラによりＡＤＲＥＳアーキテクチャのためのアプリケーションをコンパイルするとき、ＣＧＡ１２のためのループがモジュロースケジュールされ、残余のコードがＶＬＩＷ１１のためにコンパイルされる。ランタイムにてＶＬＩＷモードとＣＧＡモードとの間でアーキテクチャをシームレスでスイッチすることにより、静的に区分されスケジュー

ルされるアプリケーションが、多数の、クロックサイクル毎の命令（ＩＰＣ）数で、ＡＤＲＥＳインスタンス上で稼動し得る。

【００５７】

ループ内部の制御フローを除去するために、ＦＵｓ１３は叙述動作をサポートする。ＦＵｓ１３の結果は、分散型ＲＦｓ１４、即ち特定のファンクションユニット１３専用のＲＦｓ１４などのデータストレージに書き込まれ得るが、そのＲＦｓ１４は小さいものであり、複数のファンクションユニット１３間で共有される少なくとも一つのグローバルデータストレージであるレジスタファイルＲＦ’１５などの共有データストレージよりもポート数が少ない。ＦＵｓ１３の結果は、若しくは別のＦＵｓ１３にルート付けされ得る。タイミングを保証するために、ＦＵｓ１３のアウトプットは、アウトプットレジスタによりバッファされ得る。マルチプレクサ３２は、ＦＵｓ１３を少なくとも２つの非オーバーラップの処理ユニットの中に相互接続するルータ・リソースの一部である。それらは異なるソースからのデータをルート付けするのに利用される。コンフィグレーションＲＡＭ３１（図１及び図３参照）は、ローカルに少しのコンフィグレーションを格納するが、それらは一周期ずつに基づいてロードされ得る。ローカルのコンフィグレーションＲＡＭ３１が十分に大きくないならば、上記コンフィグレーションは追加の遅延を犠牲にしてメモリヒエラルキ１０からロードされ得る。マイクロプロセッサ内の命令と同様に、上記コンフィグレーションは、オペレーションを選択してマルチプレクサを制御することにより、基本的な要素の振る舞いを制御する。上述のような詳細なデータパスの例は、図３に示される。

【００５８】

本発明に係る実施形態は、高度並行データ処理アーキテクチャ、例えば、ＡＤＲＥＳ、若しくは粗粒再構成可能アレイを、マルチ－スレッド／プロセス・デバイスにまで拡張する。上述のように、ＡＤＲＥＳインスタンスは、ファンクションユニット１３、レジスタやレジスタファイル１４等のデータストレージ、及び、ファンクションユニット１３やレジスタファイル１４を接続するバスやマルチプレクサ等の接続レジスタ１６を、含む。ＡＤＲＥＳは、必要であれば全周期で、アレイのあらゆるエレメントを独立して構成することにより、ＭＩＭＤ（マルチプルインストラクションマルチプルデータ）プログラミングモードをサポートする。更に、ファンクションユニット１３は、データ経路の幅を利用するＳＩＭＤ（シングルインストラクションマルチプルデータ）処理をサポートし得る。コードの適切な部分から超長命令レベル並行処理（ＩＰＬ）を抽出するのに、特別なプログラミングアプローチが用いられる。ＡＤＲＥＳは、実行しているファンクションユニットが先ず無い従来のＶＬＩＷ（超長命令ワード）も実装する。入手可能な命令レベル並行処理（ＩＬＰ）が先ず無いが、従来のプログラミングモデルで十分である、コードのために、利用され得る。

【００５９】

本発明に係る実施形態のために、ＡＤＲＥＳアレイは、スレッド－レベル並行処理を可能にするために、パーティションに下位分割される。あらゆるパーティション若しくはパーティションの組み合わせは、ＶＬＩＷモードで及びアレイモードで実行し得る。このマルチ－スレッドは、多重ＡＤＲＥＳインスタンスを作成することによって達成され得るが、新規のパーティションアプローチにより、２つ以上の結合したパーティション上でスレッドを稼動することもできる。このことは、再構成可能性の更なる寸法を本質的に提示するものである。

【００６０】

ＡＤＲＥＳインスタンスに係るファンクションユニット１３及びレジスタファイル１４のトポロジ、連結性、及び特性は、設計時にアーキテクチャテンプレートにより規定される。マルチ－スレッドのＡＤＲＥＳに対して、テンプレートは、パーティションに関する情報を含み、更に、個々のパーティション若しくはパーティションの組み合わせがそれ自身有効なＡＤＲＥＳインスタンスであることを保証するように、拡張される。アーキテクチャテンプレートは、コンパイラのために、特定のＡＤＲＥＳインスタンスの全ての様相

を規定する。

【００６１】

図１は、３つのパーティション１７、１８、１９を伴う可能なＡＤＲＥＳテンプレートのための例を示す。これにより、例えば、１つ、２つ若しくは３つのスレッドが、スレッドを実行するパーティションの様々な組み合わせを利用して、並行に実行され得る。例えば、シングルスレッドは、（第１のパーティション１７として示される）全体の８×８のアレイで、若しくは（第３のパーティション１９として示される）１×２の下位パーティションで、若しくは（第２のパーティション１８として示される）４×４の下位パーティションで、実行され、アレイの残部（非利用部）は低電力モードのままであることが可能である。以下では、この例のパーティションは、夫々、第１のパーティション１７、第２のパーティション１８及び第３のパーティションとして示される。パーティション１７、１８、１９は異なる寸法となるよう選択され、スレッド内で利用可能な並行処理の程度にアーキテクチャをより適合させる。このことは、異種ファンクションユニットを利用して、更に異種データパス幅と組み合わせ可能であり、特定のファンクションのためのパーティションを最適化する。

【００６２】

本発明に係る実施形態は、ＡＤＲＥＳのプログラミングを活用し拡張する。コンパイラは、例えば、パラメータファイル内のデータに基づいて、及び、ファンクション名の接頭辞や固有の即ち特別な命令のようなコード内の或る構成に基づいて、ＶＬＩＷモードのための及びアレイモードのためのコードを生成する。同様に、スプリットモードオペレーション、即ち、個々の処理ユニットが処理ユニットに割り当てられたコントロールモジュールの制御下にある、複数の非オーバーラップの処理ユニット上で、複数のスレッドが並行して稼動しているオペレーションのモードは、例えば、パラメータファイル内にセッティングし、分割し結合するパーティションのための特別の命令を利用することにより、示され得るものである。スレッドの最初と後続のファンクションがどのパーティションに対してマップされるかを、パラメータファイル内のエントリが記述するので、コンパイラはどのアーキテクチャテンプレートを利用するか分かっている。

【００６３】

拡張コンパイラは、コード内の並行スレッドを自動的に識別し、パフォーマンス及び電力消費のために改良され若しくは最適化もされたマッピングのための利用可能なパーティションを探査し得る。

【００６４】

一方で、アレイを分割する命令、即ちパーティションをサブパーティションの中に挿入することで、コード内にスレッドを規定することはプログラマの仕事である。命令の引数は、個々の並行スレッド内の第１の関数に対する照会、例えばポインタを少なくとも含む。このように分割命令のメカニズムは、アレイ若しくはパーティションの状態を変更すること、及び、サブルーチンコールに類似するものを介してスレッドを誘発することである。リターンアドレスをセーブすることの他に、命令は分割動作のためのパーティションのレジスタ１５のセットアップもする、即ち、全ての新しいスレッドのためのスタックポインタを初期化する。アレイのデータレジスタファイルは、アレイの潜在的なパーティション間でクラスタが共有されないように、クラスタ化レジスタファイルとして実装されるべきである。

【００６５】

どのパーティション１７、１８、１９もそれ自身の制御信号のセット、即ち、ＶＬＩＷ及びアレイモードのためのプログラムカウンタ２１、２２、２３及びモードと別ステータスのフラグ２４、２５を有する。更に、どのパーティション１７、１８、１９も、これら信号を駆動するそれ自身の制御モジュール２６、２７、２８を有する。統合モード、即ち、単一処理ユニットとの相互接続構成で稼動するとき、一つの制御モジュール２６は、図２に示すように、他の制御モジュール２７、２８に追随させて、相互接続構成内部で全てのパーティションの制御信号を駆動する。制御モジュール２６、２７、２８は同じモジュ

ールの多重インスタンス化であってもよい。同期して実行するためにプログラムカウンタ２１、２２、２３が一つの制御モジュールから次の制御モジュールへと押されるのであってもよい。制御モジュール２６、２７、２８を同期化するための他の実装も可能である。一つの形態は、現存の実装から制御モジュールを再利用できることであり、その場合殆ど必要な拡張は無い。分割が実行されると、個別の制御モジュール２６、２７、２８は、分割命令内のファンクションポインタに規定されるように、個々のスレッド内の最初の命令を実行することを開始する。分割モード内の個々の制御モジュール２６、２７、２８は、その夫々の信号を駆動する。プログラマの観点から、パーティションは個別のＡＤＲＥＳインスタンスのように動作し、ＶＬＩＷモードとＣＧＡモードの間で独立してスイッチできる。

【００６６】

しかしながら、プログラマは、全てのスレッドの間で共有されるデータメモリを使うことができる。更に、拡張コンパイラは、この場合マルチスレッドのためのメモリアロケーションのタスクによって、プログラマをサポートできる。スレッド間でのデータの同期及び共有のために、拡張アーキテクチャは、セマフォレスや他のマルチ処理の基本要素を実効的に実装するための特別な命令を設ける。

【００６７】

共有アーキテクチャ要素は、命令メモリである。これはプログラミングモデルに直接に影響を与えるものではなく、コードパーティションをリンクするとき、並行スレッドのコードをパックするためにリンカ若しくはリンクモジュールがパーティションを承知していることを要求するに過ぎない。どのパーティションも、夫々の制御モジュール２６、２７、２８に直接接続する独立の命令フェッチユニット２９ａ、２９ｂ、２９ｃを有する。統合モードでは、制御モジュール２６、２７、２８はプログラムカウンタ２１、２２、２３を介して同期される。この場合、命令ユニット２９ａ、２９ｂ、２９ｃは、統合パーティションのための命令のセットの一部をフェッチし、その結果実行は同じフローを追う。分割モードでは、個々のコントローラ２６、２７、２８はそれ自身のスレッドの実行フローを追い、個々の命令ユニット２９ａ、２９ｂ、２９ｃは、夫々のサブパーティションのための命令のセットをフェッチする。制御モジュール２６、２７、２８と命令フェッチユニット２９ａ、２９ｂ、２９ｃとの間のつながりは図２に示される。

【００６８】

スレッドを結合するために、今のスレッドを終わらせるコードの中に、特別な結合命令が挿入される。夫々のパーティションが自動的に低電力モードに入れられ得る。分割命令から起動された最後のスレッドが終了すると、組み合ったばかりのパーティションで実行が継続し、次の命令は分割を追随する。リカバリルーチンのために、スレッドにモニタさせ、必要時には別のスレッドを中断する特別なメカニズムが設けられる。

【００６９】

概略、本発明のこの実施形態は、既に高度に並行で再構成可能なアーキテクチャを拡張し、更なる並行処理の寸法及び再構成可能性を備えるものとする。上記実施形態は、簡潔なプログラミングモデルを保持しつつも、現存のアーキテクチャ及びツール、特にコンパイラを利用する。マルチスレッドの拡張により、粗粒アレイのユーザは、複雑なデータ－レベルの並行処理と共に、新たなアプリケーション内に見出される適用アルゴリズムに変数を実効的に実装する、ファンクション－レベルの並行処理を利用できる。本発明のプログラム可能性と簡潔性は、鍵となる分化のファクタである。

【００７０】

例として、本発明の実施形態に係るマルチスレッドアーキテクチャのデモンストレーションのために、ＭＰＥＧ２デコーダが利用される。大抵のＭＰＥＧ２デコーダカーネルは、クロック毎の命令数（ＩＰＣ）が８から４３の範囲で、ＣＧＡ上にスケジュールされ得る。しかしながら、ＣＧＡのサイズが増加しても、モジュロ－スケジュールされたカーネルＩＰＣのうちには十分に拡大しないものがある、ということが見出されている。積極的なアーキテクチャには、クロックサイクル毎に６４命令を実行するポテンシャルを有する

ものもあるが、このレベルの並行処理を活用できるアプリケーションは殆ど無く、平均ＩＰＣがより低くなるという結果となる。このことは２つの理由により生じる。（１）カーネルの固有のＩＬＰは低く、ループ展開でも実効的には増加し得ず、若しくは、コードが複雑であり例えばメモリーポートなどのリソース制約によりそれ程多くのユニット上に実効的にスケジュールできない。（２）ＶＬＩＷモードで逐次コードを実行する際ＣＧＡが遊んでいる。より多くの逐次コードが実行される程、アプリケーションの平均ＩＰＣの達成は低くなり、更にはＣＧＡの利用が少なくなる。結論として、ＡＤＲＥＳアークテクチャが非常に拡張可能であっても、多数のアプリケーションからより多くの並行処理を取り出す課題に直面する。それは、より小さいＡＤＲＥＳアレイで実行されるのがより適切である。このことは、非特許文献１にて、Ｇ．Ｍ．Ａｍｄａｈｌにより記載されるように、Ａｍｄａｈｌの法則として周知である。

【００７１】

プログラミング時に適切に再構成され変換されるならば、同じアプリケーション内のマルチカーネルは、アプリケーション設計者により実効的に並行処理され得る。低－ＬＬＰカーネルはプロファイリングを介して静的に識別され得、個々のカーネルに対するＡＤＲＥＳアレイサイズの最適な選択が評価可能であり、大きいＡＤＲＥＳアレイが、可能であればスレッドに並行処理化される個々のカーネルに適合する複数の小スケールのＡＤＲＥＳサブアレイに、分割可能である。アプリケーションが実行されるとき、複数の低－ＬＬＰカーネルを並行して実行するために、大きいＡＤＲＥＳアレイは複数のより小さいサブ－アレイに分割され得る。同様に、高－ＬＬＰカーネルが実行されるとき、サブアレイは大きいＡＤＲＥＳアレイに統合され得る。そのようなマルチスレッドＡＤＲＥＳ（ＭＴ－ＡＤＲＥＳ）は非常に可撓性があり、アプリケーションのＬＬＰが探査し難いとき大きいスケールのＡＤＲＥＳアレイの過度な利用を増加し得る。

【００７２】

以下では、マッチングコンパイルツールと共に、単一スレッドアーキテクチャの頂部に実装されるＭＰＥＧ２デコーダ上に、例示の試験的なデュアルスレッドが、示される。この試験を介して、マルチスレッドがＡＤＲＥＳアーキテクチャに適していることが証明された。

【００７３】

拡張性のあるパーティションベースのアプローチが、ＡＤＲＥＳ等の粗粒再構成可能アーキテクチャに対して提案されている。ＡＤＲＥＳアーキテクチャ上の豊かなリソースにより、大きい粗粒再構成可能アレイを２つ以上のサブアレイに分割することができ、該サブアレイの各々は規模縮小の粗粒再構成可能アーキテクチャとして見ることができ、図４に示すように更に下位階層状に分割できる。本発明の実施形態に係る分割技術により、汎用プロセッサで利用される動的な異常実行の制御ロジックのコスト無しに、スレッド間でＨＷリソースを動的に共有できる。

【００７４】

個々のスレッドは、それ自身のリソース要求を有する。高度のＩＬＰを有するスレッドはより多くの計算機リソースを要求し、従ってより大きいパーティション上でそれを実行することは、ＡＤＲＥＳアレイをより実効的に利用することになり、その逆も同様である。広域で最適なアプリケーション設計は、プログラマがアプリケーションの個々の部分のＩＰＣを承知し個々のスレッドに対する実効的なアレイパーティションを見出せることを、要求する。

【００７５】

あるアプリケーションの個々の部分によりどれだけ多くのリソースが要求されるかを見出す最も容易な方法は、コードをプロファイルすることである。プログラマは単一スレッドのアプリケーションから開始し、大きい単一スレッド再構成粗粒アレイ上でそれをプロファイルする。プロファイルの結果から、低ＩＰＣを伴い、他のカーネルに従属しないカーネルは、スレッドのための高プライオリティの候補として識別される。リソース要求に依拠し、スレッドに依存して、プログラマは、アプリケーションの実行の間、どのように

いつ再構成可能粗粒アレイがパーティションに分割されるべきかについて、静的に計画する。スレッドが十分に構成されると、全体のアレイは実効的に利用され得る。

【００７６】

アーキテクチャ設計形態

【００７７】

ＡＤＲＥＳ上のＦＵアレイは異種である、これは複数の様々なＦＵｓ１３がアレイ内に存在するという意味である。分割を強制するアレイ上には専用メモリユニット、特別の演算ユニット及び制御／分岐ユニットが存在する。アレイを分割するとき、あるパーティション上で実行されるプログラムがスケジュールされ得ることが保証されなければならない。スレッド内に呼び出されるどの命令もアレイパーティション内のファンクションユニットの少なくとも一つによりサポートされることを、このことは要求している。適格なパーティションは、通常、分岐動作を為し得る少なくとも一つのＶＬＩＷ　ＦＵ、記憶動作を為し得る一つのＦＵ、必要ならば複数の演算ユニット、及び、一般的動作を処理できる複数のＦＵを有する。

【００７８】

ＡＤＲＥＳアーキテクチャでは、ＶＬＩＷレジスタファイル（ＲＦ’）１５は、容易には分割され得ないリソースである。ＡＤＲＥＳアーキテクチャは、クラスタ化レジスタファイルを使用し得る。ＲＦバンクが複数のスレッド間で共有されることを禁じられるならば、ＲＦクラスタはＶＬＩＷ／ＣＧＡにより分割可能であり、スレッドコンパイルは非常に平易化され得る。シングルレジスタが利用される場合、レジスタ割り当てスキームは、強制レジスタ割り当てをサポートするように改められなければならない。

【００７９】

ＡＤＲＥＳアーキテクチャは、超広メモリ帯域幅を有してもよい。バンクコンフリクトを減少するために上記アーキテクチャに適用されるマルチバンクメモリは、静的データアロケーションスキームを適切に対処することが判明している。ＡＤＲＥＳでは、メモリ及びアルゴリズムコアは、キューを伴うクロスバーと適合し得る。そのようなメモリインタフェースは、スクラッチパッドスタイルのメモリ提示を全てのロード／ストアユニットに対して用意し、従ってマルチバンクメモリは共有される同期メモリとして利用され得る。

【００８０】

共有されるメモリの他に、レジスタベースのセマフォレス若しくはパイプなどの他の専用同期基本要素も、ＡＤＲＥＳテンプレートに適用され得る。これらの基本要素は、異なるスレッドパーティションに属するファンクションユニットのペアの間で接続され得る。同期命令は、固有のものとして或るファンクションユニットに加えられ得る。

【００８１】

単一－スレッドＡＤＲＥＳアーキテクチャでは、プログラムカウンタ及び動的再構成カウンタは、有限状態機械（ＦＳＭ）により制御され得る。マルチスレッドＡＤＲＥＳを実装するならば、拡張可能制御メカニズムが、階層的に分割されたアレイに適合するように利用され得る。

【００８２】

図５に示すように、ＦＳＭタイプコントローラは重複されてもよく、コントローラは階層状に構成されてもよい。このマルチスレッドコントローラでは、個々のパーティションはＦＳＭコントローラ５０により依然制御されるが、制御パスは、マージャ５１及びバイパサ５２と称される２つのユニットにより拡張される。マージャ５１及びバイパサ５２は、プログラム実行の間管理しやすい階層マスター－スレーブ制御を形成する。マージャパスは、フローの変更の情報をパーティションのマスタコントローラに通信するのに利用され、一方、バイパサは、現下のＰＣ若しくは構成メモリアドレスをマスタからパーティション内の全てのスレーブに伝搬する。

【００８３】

このような制御メカニズムを有することの原理は、以下の通りである。デュアルスレッドのために２つの半分に分割可能であり、夫々の半分がそれ自身のコントローラを有する

ＡＤＲＥＳアーキテクチャを想定する。できるだけ多くのコントローラを再利用するために、プログラムがデュアルスレッドモードで稼動しているとき個々のコントローラはＡＤＲＥＳの分割部（パーティション）を制御するが、プログラムが単一スレッドモードで稼動するときはコントローラの一つがＡＤＲＥＳ全体の全制御をすることも好ましい。コントローラの一つを割り当ててＡＤＲＥＳ全体を制御することにより、マスタが生成される。ＡＤＲＥＳが単一スレッドモードで稼動しているとき、マスタコントローラは更にスレーブパーティションから信号を受け取り、グローバル制御信号を生成するためにそれをマスタパーティション信号とマージする。同時に、スレーブパーティションはローカルコントローラから生成された信号をバイパスし、マスタパーティションから生成されたグローバル制御信号を追随すべきである。ＡＤＲＥＳがデュアルスレッドモードで稼動しているとき、マスタ及びスレーブコントローラは他のパーティションからくる制御信号を完全に無視しローカル信号にのみ応答する。このストラテジは、更なる分割に対処するように容易に拡張され得る。

【００８４】

マルチスレッド方法論

【００８５】

スレッドされたアプリケーションがコンパイルされ得る前に、アプリケーションは再構成されるべきである。図６に示されるように、アプリケーションは複数のスレッドファイル６１、６２、６３、６４分割され得るが、それらの各々は、特定のパーティションで、例えばアプリケーションがＣでプログラミングされていると想定するとＣ－ファイルで、実行されるべきスレッドを記載する。スレッド間で共有されるデータは、全てのスレッドファイルに含まれるグローバルファイル内で規定され、同期メカニズムにより保護される。そのような再構成は地道な努力を要するが、プログラマが様々なスレッド／パーティションの組み合わせで試行し実効的な、例えば最適なリソースの予算を立てることを、より容易なものにする。図６に示される実施形態では、タスク１は統合モードで最初に実行される。タスク１の実行後、ＡＤＲＥＳアーキテクチャは、タスク２、タスク３及びタスク４を並行して実行する３つの並行処理ユニットに分割される。タスク２、３及び５を実行した後、ＡＤＲＥＳアーキテクチャは、再び統合モードに戻され、タスク４実行する。

【００８６】

マルチスレッドアーキテクチャディスクリプション、例えばＡＤＲＥＳアークテクチャディスクリプションは、は、図７に示すように、パーティションディスクリプションにより拡張する。業務用のＦＰＧＡ上のエリア制約された配置及びルート付けと同様に、スレッドがＡＤＲＥＳパーティション上でスケジュールされると、命令の配置及びルート付けはパーティションの記載により制約される。個々のスレッドの生成されたアセンブリコードは、独立してアセンブルプロセスを介して進み、最終のコンパイルのステップでリンクされる。

【００８７】

シミュレータ７０はアーキテクチャディスクリプション７１を読み取り、アプリケーションシミュレーションが開始する前にアーキテクチャシミュレーションモデルを生成する。図５に示すように、個々のパーティションはそれ自身のコントローラ５０を有し、従ってコントローラのシミュレーションモデルの生成は同様にパーティションディスクリプションに依存する。更に、制御信号分布もパーティション依存し、従ってシミュレーションモデル生成の間に参照されるべきパーティションディスクリプションを要求する。

【００８８】

本発明の実施形態に係るマルチスレッド方法論では、副次的な別の実用上の問題を対処する必要がある。最もコストのかかる問題は、ＡＤＲＥＳの異なるパーティションは概念状異なるインスタンスであり、従って特定のパーティションのためにコンパルされるファンクションは、他のどのパーティションででも実行され得ない、ということである。ファンクションが一つ以上のスレッドによりコールされるとき、このファンクションの多重パーティション－特定バイナリは様々なコーラーのために命令メモリ内に格納されねばなら

ない。次に、多重スタックがデータメモリ内に割り当てられる必要がある。

【００８９】

スレッドによりＡＤＲＥＳがより小さいパーティションに分割する度に、一時的データを格納するために新しいスタックが形成されるべきである。現下、新しいスタックがどこに形成されるべきかを決定する最良の解決策は、プロファイリングに基づくものであり、スレッドスタックはコンパイル時に割り当てられる。最終的に、新しいスレッドが形成される毎に、特別目的のレジスタの新しいセットが初期化される必要がある。スレッドが稼動を開始した直後に、スタックポイント、戻りレジスタなどを適切に初期化するには、複数のクロックサイクルが必要である。

【００９０】

実験

【００９１】

本発明の実施形態に係るマルチスレッド方法論をサポートするのにどの特性が望ましいかを理解し、その実現可能性を証明するために、十分に理解されたベンチマークであるＭＰＥＧ２デコーダに基づいて、実験を行った。実行可能なスレッドアプリケーションを生成し、スレッドのため命令／データメモリを分割し、サイクルに正確なアーキテクチャシミュレーションを更新し、本発明の実施形態に係るシミュレータによりＭＰＥＧ２デコーダの実行を首尾よくシミュレートする、という全体のプロセスを経過することが、目的である。全体のプロセスを経過することにより、スレッドのためのコンパイル、及び、ＭＴ－ＡＤＲＥＳのシミュレーション／ＲＴＬモデル生成をいかにして自動化するかに関する豊富な知識を取得できる。

【００９２】

概念実証の実験は、ＭＰＥＧ２デコーダ上でデュアルスレッドに達する。ＭＰＥＧ２デコーダは、非特許文献２に記載されるように、複数の粒度で並行処理化され得るのであり、従って、それは実験するのに適当なアプリケーションである。離散逆コサイン変換（ＩＤＣＴ）及び動き補正（ＭＣ）が２つの並行スレッドとして選択され、図８に示すようにＭＰＥＧ２デコーダを再構成した。デコーダは８×４アレイ８０で実行を開始し、可変長デコード（ＶＬＤ）及び逆量子化（ＩＱ）を実行し、スレッドモード（スプリットモード）にスイッチする。スレッド実行が開始すると、８×４アレイ８０は２つの４×４ＡＤＲＥＳアレイ８１、８２に分割し、スレッドの実行を継続する。両方のスレッドが終了すると、２つの４×４アレイ８１、８２は統合し８×４アレイ８０において統合モードで追加ブロックファンクションを実行する。ＭＰＥＧ２プログラムは図８に示すように再構成され、固有のものとして“分割”命令（フォーク命令）及び“統合”命令（結合命令）を追加した。これらの命令８３、８４は、一般にそれら自身何も行わないが、ＭＰＥＧ２バイナリコードでスレッドモードが変化すべきところをマークするためにのみ利用される。これらのマークは、スレッドモードプログラム実行を可能にするために／不能にするために、ランタイムにて分割制御ユニットにより利用される。

【００９３】

本発明に係るデュアルスレッドコンパイルフローが図９に示される。パーティションベースのスケジューリングの欠如により、スケジューリングへのインプットとして２つのアーキテクチャを利用することが余儀なくされる。８×４アーキテクチャ９０は、右及び左の半分が全く同一になるように慎重に設計される。このアーキテクチャは全体のＭＰＥＧ２バイナリの実行プラットフォームである。４×４アークテクチャも必要とされ、該アークテクチャは８×４アレイの半分のいずれにも互換性のある助力アークテクチャである。このアークテクチャは８×４アーキテクチャ９０のハーフアレイパーティションディスクリプションとして利用される。これら２つのアークテクチャ９０、９１を適所に配置し、シングルスレッドファイル９２、例えばＣ－ファイルが、８×４アークテクチャ及び４×４アークテクチャの夫々でのスレッドと同様に、コンパイルされる。リンカ９５によりリンクする後者は、プログラムの様々な部分からのバイナリをシームレスでステッチする。

【００９４】

スレッドされたＭＰＥＧ２のメモリ分割が図１０に示される。命令フェッチ（ＩＦ）、データフェッチ（ＤＦ）及びコンフィグレーション－ワードフェッチ（ＣＷ）は、デュアルスレッドのために二重化される。フェッチユニットペアは、シングルスレッドプログラム実行の間にステップロックされる。アーキテクチャがデュアルスレッドモードに入ると、フェッチユニットペアは２つのセットに分割するが、それらの各々はスレッドパーティション内でコントローラにより制御される。

【００９５】

リンクの間、命令メモリ１０１及びデータメモリ１０２はパーティションの中に分割される。命令メモリ１０１とコンフィグレーションメモリ１０３の両方は、３つのパーティションに分割される。これらの３つのパーティションペアは、図１０に示すように、シングルスレッドバイナリ、ＩＤＣＴバイナリ及びＭＣバイナリの命令及びコンフィグレーションを格納する。データメモリ１０２は４つのパーティションに分割される。最大のデータメモリパーティションは、共有されるグローバル静的データメモリである。シングルスレッド及びデュアルスレッドの両方のプログラムは、それらのデータを同じメモリパーティション１０５の中に格納する。データメモリの１０２の残りは、３つのスタックの中に分割される。ＩＤＣＴスレッドスタック１０６は、シングルスレッドプログラムスタック１０７の直ぐ上で拡張する。それらは同じ物理コントローラとスタックポインタを利用するからである。ＭＣスレッドのベーススタックアドレスは、リンク時に空きメモリ割り当てに対してオフセットする。プログラム実行がデュアルスレッドモードに入ると、ＭＣスタックポインタが複数のクロック周期を費やして適切に初期化される。

【００９６】

別の実施形態では、個々のスレッドがそれ自身のレジスタファイルを備えるように、クラスタ化レジスタファイルがアレイパーティション間でクラスタ化されてもよい。しかしながら、現段階での分割ベースのレジスタ割り当てアルゴリズムの欠如のために、パーティションアプローチはそれ程適切なものではない。シングルグローバルレジスタファイルを伴うＡＤＲＥＳアーキテクチャに関し実験を行い、レジスタファイル発行に一時的に適合する二重化ベースのアプローチに進む。図１１に示すように、シャドウレジスタファイル１１０がアーキテクチャ内に追加されてもよい。シングルスレッドプログラムが実行されていると、シャドウレジスタファイル１１０が最初のレジスタファイル１５でステップロックされる。プログラムがデュアルスレッドを開始すると、ＭＣスレッドはシャドウレジスタファイル１１０へアクセスし、アレイパーティション１１２及びシャドウレジスタファイル１５上で実行を継続する。プログラムが再開してシングルスレッドの実行に到ると、シャドウレジスタファイル１１０が再び隠される。ＭＰＥＧ２プログラムは、スレッド間で共有されるデータの全て、並びに住み込みの及び通いの変数の全てが、グローバルデータメモリを介して通過するように、僅かに修正される。

【００９７】

図５に示す拡張性のある制御コンセプトは、本発明の実施形態に係るシミュレーションモデル内で立証された。このスキームは或る規模まで拡張可能であり、制御ユニットシミュレーションモデル生成は自動化され得ることが、示された。

【００９８】

プログラムリンクの間に、“分割”及び“統合”命令が命令メモリ内でどこに格納されるかが、特定される。これらの命令の物理アドレスは、デュアルスレッドモードの開始ポイントと終了ポイントとをマークする。シミュレーションモデル生成の間、これらの命令のアドレスは、分割制御ユニット内の専用レジスタのセット内に格納される。プログラムが実行することを開始すると、個々のクロック周期にて、分割制御ユニットによりプログラムカウンタ（ＰＣ）値がチェックされる。プログラムカウンタが分割ポイントに達すると、分割制御ユニットは制御信号をマージャ及びバイパサに送信し、スレッドモードをイネーブルにする。プログラムがスレッドモードに移行する後、分割コントローラは、“統合”命令が格納されるところでＰＣ値に到達することにより両方のスレッドが結び付くことを、待つ。結び付く第１のスレッドは、他のスレッドが終了するまで、停止する。第２

のスレッドがついに結びつくと、分割コントローラはＡＤＲＥＳアレイをシングルスレッドモードに切り替えて戻し、アーキテクチャは８×４アレイモードを再開する。分割及び統合動作を行うことのオーバーヘッドは、主として、或る専用レジスタ上の複数の簿記命令を実行することから生じてくるのであるが、そのようなオーバーヘッドは無視できる。

【００９９】

アプリケーションがより複雑となり多重の分割／統合ポイントを有すると、現下のアプローチは処理するのにより困難なものとなり、よって、本発明の実施形態に係るアーキテクチャは、命令デコードにのみ依存して、“分割”及び“統合”命令を検出し得る。分割制御ユニットが除去され、その機能の一部が個々のパーティションのローカルコントローラ内に移されてもよい。

【０１００】

シミュレーション結果は、ＭＰＥＧ２が僅かに早い率で正確なイメージフレームを生成することを示す。表１は、スレッドと共に及びスレッド無しで、同じ８×４ＡＤＲＥＳインスタンスでデコードされた最初の５イメージフレームのクロックカウントを示す。

【０１０１】

【表１】

フレーム番号	シングルスレッドクロック周期カウント	デュアルスレッドクロック周期カウント	シングルスレッドデコード時間	デュアルスレッドデコード時間	スピードアップ
1	1874009	1802277			
2	2453279	2293927	579270	491650	15.1%
3	3113150	2874078	659871	580151	12.1%
4	3702269	3374421	589119	600343	15.1%
5	4278995	3861978	576726	487557	15.5%

【０１０２】

表１．同じアーキテクチャでのシングル及びデュアルスレッドのＭＰＥＧ２のクロック周期カウント

【０１０３】

クロック周期カウントのコラムは、イメージフレームがデコードされたときのオーバーオール実行時間のクロックカウントを示し、デコード時間のコラムは、デコードされる２つのフレーム間のクロックカウントを示す。以下の理由により、デュアルスレッドのＭＰＥＧ２はシングルスレッドのＭＰＥＧ２より約１２－１５％速い。

【０１０４】

ＩＤＣＴ及びＭＣアルゴリズムの両方は、高度なループレベルの並行処理を有し、従ってシングルスレッドの８×４アーキテクチャを最適に用いる。スレッドとして×４アーキテクチャ上にスケジュールされると、両方のアルゴリズムのＩＰＣは、半減されたアレイサイズのために半分に減少され、従って非スレッドとスレッドのＭＰＥＧ２の全体のＩＰＣは、略同じである。前に述べたように、ＡＤＲＥＳサイズがある程度まで増加すると、スケジューリングアルゴリズムは、アプリケーション内で並行処理を探査すること、及び、最適にＡＤＲＥＳアレイを利用することが、困難になる。ＡＤＲＥＳアレイのサイズを二倍にすること／四倍にすること、若しくは、スレッドのために低並行処理アルゴリズムを選択することは、結果としてよりスピードアップとなることは、明白である。

【０１０５】

周知のように、より小さいアーキテクチャ上へのモジュロスケジューリングの容易さから、僅かな性能向上が大抵達せられる。アプリケーションがより大きいＣＧＡ上でスケジュールされると、ルーティングの目的のために、多数の冗長な命令がカーネルの中に付加される。全体ＡＤＲＥＳの代わりに、ＩＤＣＴ及びＭＣカーネルが半分のＣＧＡパーティ

ション上にスケジュールされると、アプリケーションの全体のＩＰＣがそれ程改良されなくとも、配置及びルーティングのためのスケジューリングの間に付加される冗長命令の量は、大きく減少した。

【０１０６】

ＭＰＥＧ２デコードアルゴリズムでデュアルスレッド実験を行うことによって、ＭＴ－ＡＤＲＥＳアーキテクチャに関する豊富な知識が得られた。シミュレーションの結果は、ＭＰＥＧ２が１２－１５％のスピードアップを得たことを示す。これまでの結果は、スレッドアプローチは、ＡＤＲＥＳアーキテクチャにとって十分であり、実用上実行可能であり、或る程度まで拡張可能であることを、示している。これまでのところ、ＡＤＲＥＳに付加される唯一の特別なハードウエアコストは、第２の制御ユニットであるが、そのサイズは、３×３より大きいＡＤＲＥＳにとって無視し得るものである。

【０１０７】

好ましい実施形態、特別の構造及び構成を、本発明に係る装置に対して、本明細書で説明してきたが、当然のことながら、形状及び詳細における様々な変更若しくは修正が、本発明の範囲及び精神から乖離することなく、為され得る。

【図面の簡単な説明】

【０１０８】

【図１】本発明の実施形態と共に利用する粗粒アレイの実施形態の例を示す。

【図２】本発明の実施形態に係るコントロールモジュール及び命令フェッチユニットの再利用可能性及び拡張性のコンセプトを示す。

【図３】本発明の実施形態に係るファンクションユニットの詳細なデータパスを示す。

【図４】本発明の実施形態に係る拡張可能なパーティションベースのスレッドを示す。

【図５】本発明の実施形態に係る階層式マルチスレッドコントローラを示す。

【図６】本発明の実施形態に係るソースコード再編成を示す。

【図７】本発明の実施形態に係るマルチスレッドコンパイルツールチェーンを示す。

【図８】例として、ＭＰＥＧ２デコーダでのスレッドを示す。

【図９】試行的なデユアルスレッドコンパイルフローを示す。

【図１０】本発明の実施形態に係るデユアルスレッドメモリマネジメントを示す。

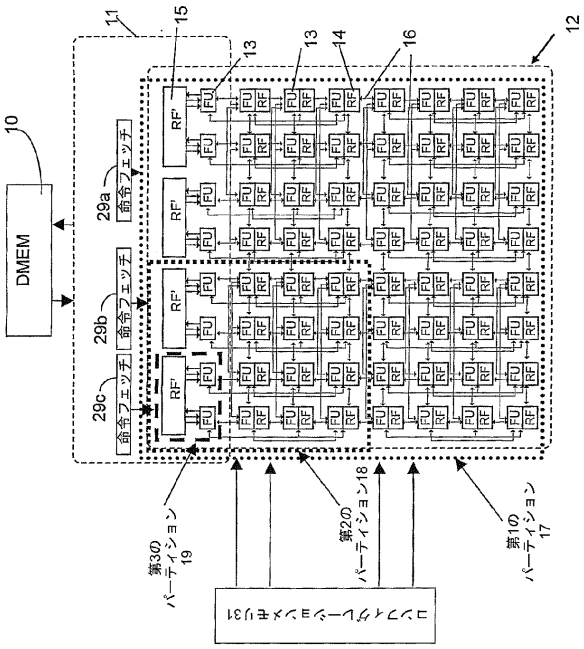
【図１１】本発明の実施形態によってセットアップされるシャドーレジスタファイルを示す。

【符号の説明】

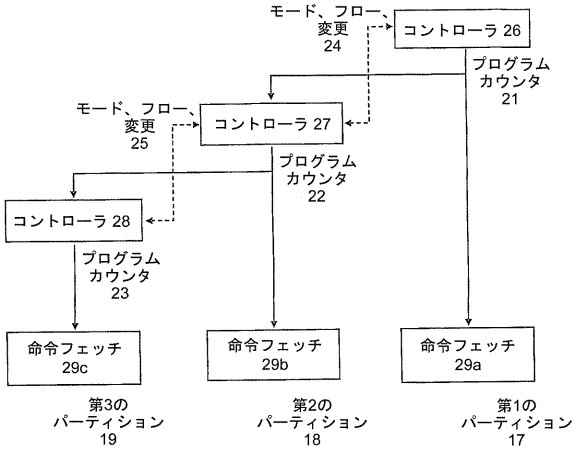
【０１０９】

１７・・・第１のパーティション、１８・・・第２のパーティション、１９・・・第３のパーティション、２１、２２、２３・・・プログラムカウンタ、２６、２７，２８・・・コントローラ、２９ａ、２９ｂ、２９ｃ・・・命令フェッチ、３１・・・コンフィグレーションメモリ。

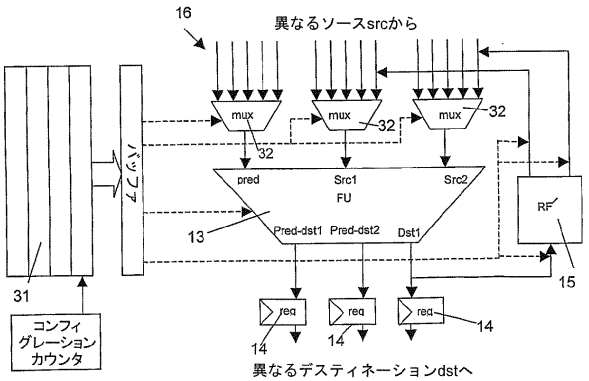
【図１】



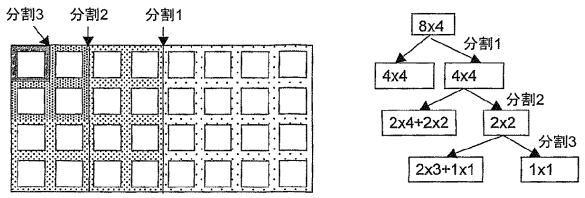
【図２】



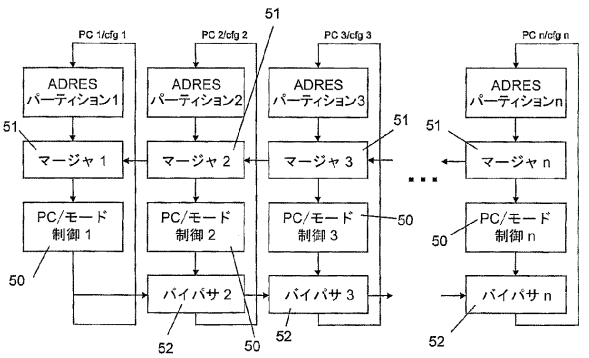
【図３】



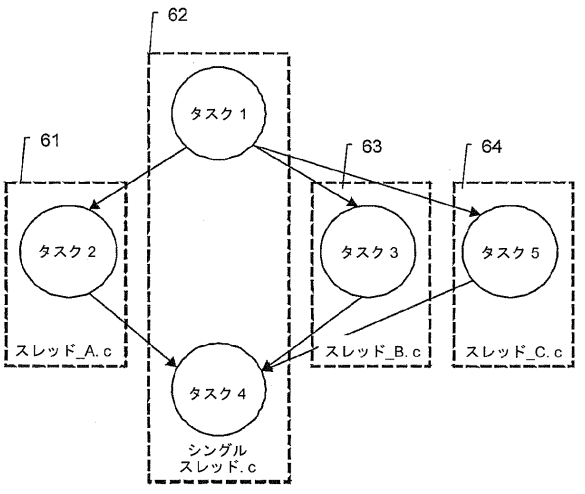
【図４】



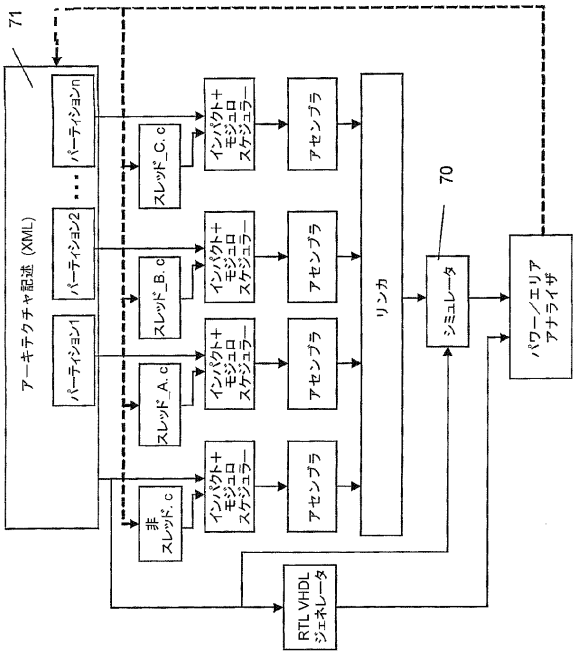
【図５】



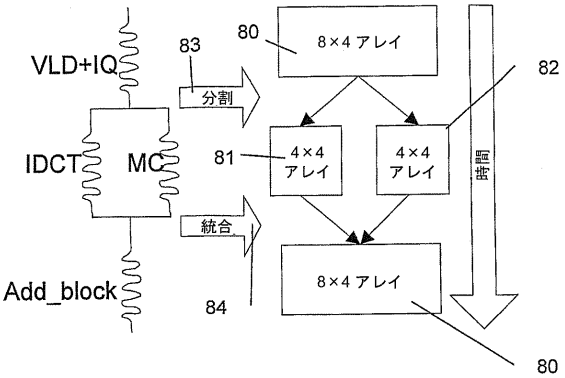
【図６】



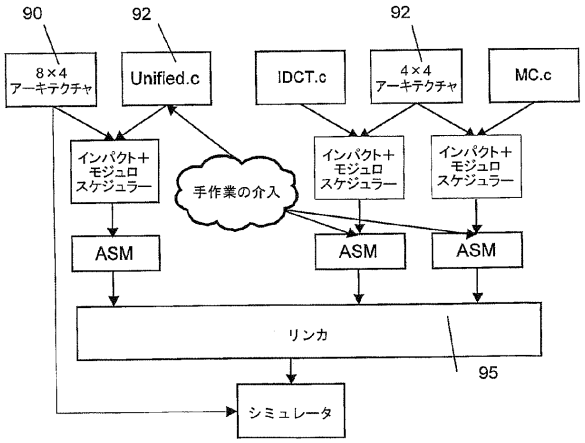
【図７】



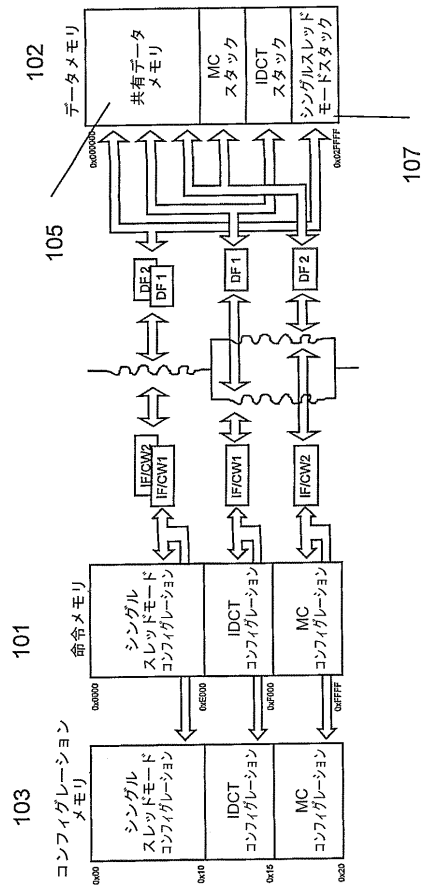
【図８】



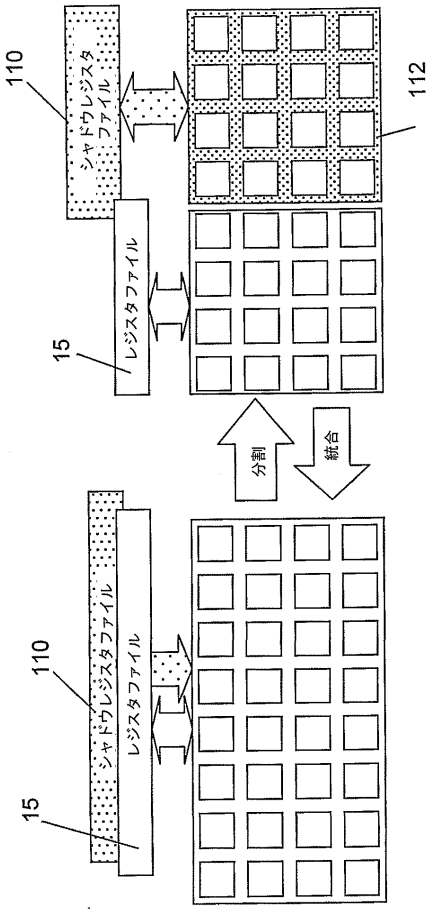
【図９】



【図１０】



【図１１】



フロントページの続き

(72)発明者　ムラデン・ベレコヴィク
ドイツ連邦共和国デー－３０９６６ヘミンゲン、ラウバイヒェンフェルト３２番

審査官　柳下　勝幸

(56)参考文献　国際公開第２００５／０２２３８０（WO，Ａ１）
特開２００６－０４０２５４（ＪＰ，Ａ）
特開２００３－０７６６６７（ＪＰ，Ａ）
特開２００１－２８２５３３（ＪＰ，Ａ）
Mladen Berekovic, Andreas Kanstein, Bingfeng Mei, apping MPEG Video Decoders on the ADRES Reconfigurable Array Processor for next generation multi-mode mobile terminals, Proeedings of GSPX 2006: TV to Mobile, Springer, ２００６年　3月, ＵＲＬ, http://130.161.38.15/publicationfiles/1307_801_GSPX_2006_Berekovic.pdf
Bingfeng Mei, Francisco-Javier Vereda, Bart Masschelein, MAPPING AN H.264/AVC DECODER ONTO THE ADRES RECONFIGURABLE ARCHITECTURE, International Conference on Field Programmable Logic and Applications, 2005, フィンランド, IEEE, ２００５年　8月２６日, p622-625

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０３Ｋ　１９／１７７
Ｇ０６Ｆ　９／４５