(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4560705号
(P4560705)
(45)発行日　平成22年10月13日(2010.10.13)　(24)登録日　平成22年8月6日(2010.8.6)

(51)Int.Cl.　F I
G06F 9/30 (2006.01)　G06F 9/30 370

請求項の数 3　(全 27 頁)

(21)出願番号　特願2003-291251(P2003-291251)
(22)出願日　平成15年8月11日(2003.8.11)
(62)分割の表示　特願2001-520597(P2001-520597)の分割
原出願日　平成12年8月30日(2000.8.30)
(65)公開番号　特開2004-5739(P2004-5739A)
(43)公開日　平成16年1月8日(2004.1.8)
審査請求日　平成19年8月30日(2007.8.30)
(31)優先権主張番号　特願平11-244137
(32)優先日　平成11年8月30日(1999.8.30)
(33)優先権主張国　日本国(JP)

(73)特許権者　000005496
富士ゼロックス株式会社
東京都港区赤坂九丁目７番３号
(74)代理人　100102934
弁理士　今井　彰
(72)発明者　佐藤　友美
茨城県つくば市東２丁目１８番地１０　ルーミつくば３１号２０２
審査官　三坂　敏夫

最終頁に続く

(54)【発明の名称】データ処理装置の制御方法

(57)【特許請求の範囲】
【請求項１】
再構成可能なデータ処理装置の制御方法であって、
前記データ処理装置は、演算または他のデータ処理を実行する複数の処理ユニットであって、それぞれ独自のコンフィグレーションメモリを備え、入力および／または出力インタフェースおよび処理内容が変更される複数の処理ユニットと、前記複数の処理ユニットの入力および／または出力インタフェースおよび処理内容を規定する制御ユニットとを含み、前記複数の処理ユニットの少なくとも一部を用いて少なくとも１つのデータフローが構成され、
前記制御ユニットが、前記複数の処理ユニットの少なくとも一部の処理ユニットの前記独自のコンフィグレーションメモリのデータをそれぞれ、特定のデータ処理の実行が決定される１または数クロック前に書き換える工程と、
前記制御ユニットが、前記少なくとも一部の処理ユニットに対し、前記少なくとも一部の処理ユニットが、共に、それぞれの入力および／または出力インタフェースおよび処理内容を、それぞれの前記独自のコンフィグレーションメモリに記憶されたデータにしたがって切り替える命令を出し、前記少なくとも一部の処理ユニットにより前記特定のデータ処理を実行するデータフローを再構成する工程とを有するデータ処理装置の制御方法。
【請求項２】
請求項１において、前記複数の処理ユニットにより複数のデータ処理をそれぞれ実行可能な複数のデータフローを構成可能であり、

前記再構成する工程では、他のデータ処理と並列して、前記特定のデータ処理を実行するデータフローを再構成する、データ処理装置の制御方法。

【請求項３】

演算または他のデータ処理を実行する複数の処理ユニットであって、それぞれ独自のコンフィグレーションメモリを備え、入力および／または出力インタフェースおよび処理内容が変更される複数の処理ユニットと、

前記複数の処理ユニットの入力および／または出力インタフェースおよび処理内容を規定する制御ユニットとを有し、前記複数の処理ユニットの少なくとも一部を用いて少なくとも１つのデータフローが構成される、再構成可能なデータ処理装置であって、

前記制御ユニットは、前記複数の処理ユニットの少なくとも一部の処理ユニットの前記独自のコンフィグレーションメモリのデータをそれぞれ、特定のデータ処理の実行が決定される１または数クロック前に書き換える機能と、

前記少なくとも一部の処理ユニットに対し、前記少なくとも一部の処理ユニットが、共に、それぞれの入力および／または出力インタフェースおよび処理内容を、それぞれの前記独自のコンフィグレーションメモリに記憶されたデータにしたがって切り替える命令を出し、前記少なくとも一部の処理ユニットにより前記特定のデータ処理を実行するデータフローを再構成する機能とを含む、データ処理装置。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、再構成可能なハードウェアを有するデータ処理装置の制御方法に関するものである。

【背景技術】

【０００２】

汎用的な処理および専用的なデジタルデータの処理を行う装置として、マイクロプロセッサー（ＭＰＵ）、デジタル・シグナルプロセッサー（ＤＳＰ）といった演算機能を内蔵したプロセッサ（データ処理装置あるいはＬＳＩ）が知られている。これらの性能向上に大きく貢献したアーキテクチャ要素として、パイプライン化技術、スーパー・パイプライン化技術、スーパー・スケーラ化技術、ＶＬＩＷ技術、特化型データパス（専用命令）追加を挙げることができる。さらに、分岐予測やレジスタバンク、キャッシュ技術等も挙げることができる。

【発明の開示】

【発明が解決しようとする課題】

【０００３】

ノン・パイプラインとパイプラインとの性能差は明確である。基本的に同一命令であれば、パイプラインの段数だけスループットは確実に向上する。たとえば、４段のパイプラインでは、スループットは４倍以上となることが期待でき、８段のパイプラインであれば、８倍という計算となり、スーパー・パイプライン技術は、更に性能を２倍以上向上させたことになる。実際は、プロセスの進歩に従い、クリティカルパスを細分化可能な為、動作周波数の上限を大きく改善するという効果も期待出来るので、その貢献度は更に大きいものとなって現われる。しかしながら、分岐命令のディレイ（ペナルティ）は解消されておらず、スーパー・パイプライン方式のマシンが成功するか否かは、メモリアクセスや分岐に対応した深い段数の遅延を、コンパイラによる命令スケジューリングなどによってどこまで処理できるかにかかっている。

【０００４】

次に、スーパー・スケーラ技術であるが、これは内部のデータパスを高性能化し、プログラムカウンタ近傍の命令を同時実行するものである。この技術はコンパイラの最適化技術の進歩にも支えられて、４命令から８命令程度の同時実行が可能になったとされている。しかしながら、命令自体はその直前の演算結果やレジスタの結果を頻繁に使用する事が多く、ピーク性能は別にして、フォワーディングや命令再配置、アウト・オブ・オーダ、

レジスタリネーミング等の各種テクニックを駆使したとしても同時実行可能な命令数は平均的には上記よりかなり低い値とならざるを得ない。特に、条件分岐命令等を複数実行することは不可能なので、スーパー・スケーラ技術の効果は更に低いものとなる。したがって、プロセッサの性能向上への貢献度としては、平均２．０から２．５倍程度と思われる。非常に相性の良いアプリケーションが仮にあったとしても、現実的な貢献度は４倍以下と考えられる。

【０００５】

ＶＬＩＷ技術が、次の技術として浮上する。これは、予めデータパスを並列実行可能なように構成しておき、コンパイラがこの並列実行を高めるように、最適化を行い目的のＶＬＩＷ命令コードを生成するという考え方であり、極めて合理的な考え方を採用している。これにより、スーパー・スケーラのように１つ１つの命令の並列実行の可能性をチェックする回路が不要なので、並列実行を行うハードウェアの実装手段としては、極めて有望とされているものである。しかしながら、条件分岐命令などを複数実行できないことは上記と同様であり、そのため、実際の性能に対する貢献度としては、３．５倍～５倍程度と考えられる。しかしながら、画像処理や特殊データ処理を必要とするアプリケーションの処理を用途とするプロセッサを考えると、ＶＬＩＷも最適な解決策とはならない。特に演算結果の連続処理を要求されるような用途では、汎用レジスタにデータを抱えながらの演算やデータ処理には限界があるからである。これは従来のパイプライン技術でも同様である。

【０００６】

一方、各種のマトリックス計算やベクトル計算等は、専用回路によりこれを実現した方が高い性能を得られることは過去の経験から良く知られている。このため、現在、世界最高性能を目指す最先端の実装技術では、ＶＬＩＷをベースにアプリケーション目的に応じて、各種の専用演算回路を実装して、最高性能を目指すという考え方が主流になりつつある。

【０００７】

しかしながら、ＶＬＩＷは、プログラムカウンタ近傍の並列処理実行効率を改善する技術であり、例えば２つ以上のオブジェクトを同時に実行したり、２つ以上の関数を実行するにはあまり有効な手段とはならない。また、各種の専用演算回路を実装することはハードウェアが増加することとなり、その一方で、ソフトウェアのフレキシビリティーが低下することを意味する。さらに、条件分岐を実行するときに発生するペナルティの問題を本質的に解決し難い。

【０００８】

そこで、本発明においては、これらの従来のプロセッサを高速化する技術と異なった視点から上記の問題を検討し、新たな解決策を提供することを目的としている。すなわち、パイプラインのようにスループットの向上を図ることができると共に、条件分岐を実行する際のペナルティを解決することが可能な制御方法を提供することを目的としている。さらに、複雑なデータ処理であっても、それらのデータ処理に特化した多種多様な専用回路を用いなくても、それぞれのデータ処理をフレキシブルに、そして高速に実行可能な制御方法を提供することも本発明の目的としている。

【課題を解決するための手段】

【０００９】

本発明の一態様は、再構成可能なデータ処理装置の制御方法である。データ処理装置は、演算または他のデータ処理を実行する複数の処理ユニットであって、それぞれ独自のコンフィグレーションメモリを備え、入力および／または出力インタフェースおよび処理内容が変更される複数の処理ユニットと、前記複数の処理ユニットの入力および／または出力インタフェースおよび処理内容を規定する制御ユニットとを含み、複数の処理ユニットの少なくとも一部を用いて少なくとも１つのデータフローが構成される。制御方法は、制御ユニットが、複数の処理ユニットの少なくとも一部の処理ユニットの独自のコンフィグレーションメモリのデータをそれぞれ、特定のデータ処理の実行が決定される１または数

クロック前に書き換える工程と、制御ユニットが、少なくとも一部の処理ユニットに対し、それら少なくとも一部の処理ユニットが、共に、それぞれの入力および／または出力インタフェースおよび処理内容を、それぞれの独自のコンフィグレーションメモリに記憶されたデータにしたがって切り替える命令を出し、それら少なくとも一部の処理ユニットにより特定のデータ処理を実行するデータフローを再構成する工程とを有する。これにより、ハードウェアを随時、実行される蓋然性の高い特定のデータ処理に適した構成に変更でき、さらに、複雑なデータ処理であっても、それらのデータ処理に特化した多種多様な専用回路を用いずに、それぞれのデータ処理をフレキシブルに、そして高速に実行できる。複数の処理ユニットにより複数のデータ処理をそれぞれ実行可能な複数のデータフローを構成可能であれば、再構成する工程では、他のデータ処理と並列して、特定のデータ処理を実行するデータフローを再構成することができる。

本発明の他の態様の１つは、演算または他のデータ処理を実行する複数の処理ユニットであって、それぞれ独自のコンフィグレーションメモリを備え、入力および／または出力インタフェースおよび処理内容が変更される複数の処理ユニットと、複数の処理ユニットの入力および／または出力インタフェースおよび処理内容を規定する制御ユニットとを有し、複数の処理ユニットの少なくとも一部を用いて少なくとも１つのデータフローが構成される、再構成可能なデータ処理装置である。制御ユニットは、複数の処理ユニットの少なくとも一部の処理ユニットの独自のコンフィグレーションメモリのデータをそれぞれ、特定のデータ処理の実行が決定される１または数クロック前に書き換える機能と、少なくとも一部の処理ユニットに対し、それら少なくとも一部の処理ユニットが、共に、それぞれの入力および／または出力インタフェースおよび処理内容を、それぞれの独自のコンフィグレーションメモリに記憶されたデータにしたがって切り替える命令を出し、少なくとも一部の処理ユニットにより特定のデータ処理を実行するデータフローを再構成する機能とを含む。

【００１０】

すなわち、本願の発明者は、上記のような問題がノン・パイプライン技術から今までの技術に用いられている命令セットの制約から上記のような問題が生じていることを見出した。例えば、プロセッサにおけるデータ処理を規定するプログラム（マイクロコード、アセンブリコード、機械語など）の命令セット（命令フォーマット）は命令操作（実行命令）とその命令を実行する際に使用するレジスタなどの環境またはインタフェースを規定するオペランドとが組み合わせされたニーモニックコードである。したがって、命令セットを見れば、それによって指示されている処理の内容を完全に把握できるが、命令セットをデコードするまで処理の内容については全く判らない。そこで、本発明のひとつの形態においては、命令セットの構成方法そのものを大幅に変更することにより、従来技術では対応の難しかった上記の問題を上手く解決し、データ処理装置の性能を飛躍的に向上できるようにしている。

【００１１】

この発明においては、データ処理装置を構成する少なくとも１つの処理ユニットで実行する演算または他のデータ処理の内容を指示する実行命令を記述（記載）可能な第１のフィールドと、実行命令で実行する演算または他のデータ処理が実行可能な状態に処理ユニットを設定する準備情報を記述（記載）可能な第２のフィールドとを備えた命令セットを設け、第１のフィールドに記述された実行命令の内容に対し、独立した演算または他のデータ処理の準備情報が第２のフィールドに記述できるようにしている。したがって、この命令セットを有する制御プログラム製品あるいは制御プログラム装置を提供することができる。この制御プログラムは、データ処理装置が読み取り可能な適当な記録媒体に記録して提供でき、また、その制御プログラムを、コンピュータネットワークあるいはその他の通信を介して伝送される伝送媒体に埋め込んで提供できる。

【００１２】

処理ユニットは、データ処理装置を構成する適当な機能的あるいはデータパス的に分割可能な単位であり、制御ユニット、算術演算ユニット、さらには、ある程度コンパクトな

データパスを備えてテンプレート的に取り扱い可能な特定のデータパスを具備した処理ユニットあるいはデータフロー処理ユニットなどが含まれる。

【００１３】

さらに、このデータ処理装置は、演算または他のデータ処理を実行する少なくとも１つの処理ユニットと、処理ユニットで実行する演算または他のデータ処理の内容を指示する実行命令を記述可能な第１のフィールド、および実行命令で実行する演算または他のデータ処理が実行可能な状態に処理ユニットを設定する準備情報を記述可能な第２のフィールドとを具備する命令セットをフェッチするユニットと、第１のフィールドの実行命令をデコードし、その実行命令の演算または他のデータ処理が実行できるように予め設定された処理ユニットにより当該演算または他のデータ処理を進める第１の実行制御ユニットと、第２のフィールドの準備情報をデコードし、第１の実行制御ユニットの実行内容とは独立して処理ユニットの状態を演算または他のデータ処理が実行できるように設定する第２の実行制御ユニットとを有する。

【００１４】

また、上記の、演算または他のデータ処理を実行する少なくとも１つの処理ユニットを有するデータ処理装置の制御方法は、上記の第１のフィールドおよび第２のフィールドとを具備する命令セットをフェッチする工程と、第１のフィールドの実行命令をデコードし、その実行命令の演算または他のデータ処理が実行できるように予め設定された処理ユニットにより当該演算または他のデータ処理を進める第１の制御工程と、この第１の制御工程とは独立して、第２のフィールドの準備情報をデコードし処理ユニットの状態を演算または他のデータ処理が実行できるように設定する第２の制御工程とを有する。

【００１５】

本発明の１つの形態にかかる命令セットは、実行命令を記述する第１のフィールドと、この実行命令とは独立し、レジスタの情報およびイミーディエイトなどの準備情報（準備命令）を記述する第２のフィールドとを備えたものである。したがって、算術命令などにおいては、第１のフィールドにＡＤＤなどの命令操作が記述され、第２のフィールドにレジスタを特定する命令あるいは情報が記述されるので、一見、従来のアセンブルコードと同様の命令セットとなる。しかしながら、実行命令と準備情報は独立であり、同じ命令セット内では対応していない。このため、その命令セットでは制御ユニットなどのデータ処理装置の処理ユニットで実行される処理が特定されないという特性を備えている。すなわち、本発明にかかる命令セットは従来のニーモニックコードとは大きく異なるものである。そして、従来は１つの命令セットの中に記述されていた命令操作とそれに対応するオペランドを個別に、独立して定義できるようにすることにより、従来の命令セットでは実現できない処理を簡単に実行することができる。

【００１６】

まず、第２のフィールドに、後続の命令セットの第１のフィールドに記述される実行命令を実行するための準備情報を記述することができる。これにより、実行命令を備えた命令セットが表れる前に、その実行命令を実行するための準備を行うことができる。すなわち、実行命令で実行する演算またはその他のデータ処理が実行可能な状態に処理ユニットを設定することができる。例えば、ある命令セット（命令フォーマットあるいは命令レコード）の第１のフィールドにデータ処理装置のある制御ユニットに含まれる少なくとも１つの算術論理演算ユニットを操作する命令を記述し、それに先立つ命令セットの第２のフィールドに、その少なくとも１つの算術論理演算ユニットに用いられるソース側のレジスタあるいはディスティネーション側のレジスタといった算術論理演算ユニットのインタフェースを規定する命令あるいは情報を記述することができる。これにより、実行命令がフェッチされる前に、算術論理演算ユニットのレジスタ情報がデコードされ、レジスタがセットされ、その後にフェッチされた実行命令により所定の論理演算が実行され、その結果が指定されたレジスタに保存される。ディスティネーション側のレジスタは実行命令と共に第１のフィールドに記述することも可能である。

【００１７】

したがって、この命令セットにおいても、パイプライン処理と同様にデータ処理を多段階に分けて実行することが可能でありスループットを向上することができる。また、例えば、ＡＤＤ，Ｒ０，Ｒ１，＃１２３４Ｈという命令は、レジスタＲ１と＃０１２３４Ｈを加算してこれをレジスタＲ０に格納するという意味になるが、ハードウェア構成上は、前の命令セットの実行サイクルとオーバラップさせて、ＡＤＤという実行命令を実行する１ＣＬＫ前にレジスタＲ０と「＃０１２３４Ｈ」を算術論理演算ユニットである算術加算器ＡＤＤが属するデータパスの入力レジスタにリードを実行しておくと高速実行させる観点からは、都合が良い。つまり、ＡＣ特性上は、純粋に算術加算を行うようにできるので、実行周波数特性が向上する。パイプライン処理において、パイプライン段数を増加させて、レジスタファイルからのリードサイクル専用に１ステージ消費する設計方針により、この問題をある程度回避することができる。しかしながら、その結果、遅延は確実に増加することになるのに対し、本発明においては遅延を増加させずに問題を解決できる。

【００１８】

そして、この命令セットにおいては、準備情報を実行命令に先立って記述できるので、条件分岐命令などの分岐命令においては、分岐先の情報が実行命令に先立って制御ユニットに与えることができる。すなわち、従来のニーモニックコードでは、命令セットの内容は人間が一目で分かるが、その命令セットが表れるまで処理内容が判らなかった。これに対し、本発明にかかる命令セットでは、命令セットの内容は一目では分からないが、実行命令が表れる前に、その実行命令に関連する情報が分かる。したがって、実行命令に先立って分岐先が判るので、その分岐先の命令セットをフェッチすることも可能であり、さらには、分岐先の実行命令に先立ってその準備を行うことも可能となる。

【００１９】

一般に、現在のＣＰＵ／ＤＳＰの殆どがパイプライン処理を後段（時間軸が後方）にシフトすることで、処理の高速化を図ることに成功しているが、プログラムの分岐時やＣＡＬＬ／ＲＥＴ実行時には、この問題が表面化する。つまり、先行してフェッチアドレス情報が得られていない為に、本質的にペナルティとなり、原理的にこれを解消することができない。もちろん、分岐予測やディレイディド・ブランチ、高速ブランチバッファ、或いはＤＳＰにて採用されている高速ループ処理技術等は、このペナルティをかなり緩和する事に成功しているが、連続分岐が数多く発生したりすると、その問題点が表面化し、本質的な解決にはなっていないことは周知の事実である。

【００２０】

また、後続命令が必要とするレジスタ情報が先に得られない為に、パイプライン処理を高速化する為のフォワーディング処理やバイパス処理の複雑さが増大し、従来技術で高速化を図ろうとすること自体が膨大なハードウェア・コストの上昇を招く要因となる。

【００２１】

したがって、従来の命令セットでは、分岐先のアドレス情報はデコード後にしか得られず、条件分岐を実行するときに発生するペナルティを本質的に解決し難いのに対し、本発明の命令セットにおいては、分岐先の情報を事前に与えることができるので、条件分岐を実行するときのペナルティを無くすことができる。さらに、ハードウェアに余裕があれば、分岐先の準備命令をフェッチして、それに続く実行命令のための準備を行うことも可能となる。分岐条件が整わない場合は、その準備が無駄になるだけであり、実行時間のペナルティになることはない。

【００２２】

また、後続命令が必要とするレジスタ情報が、実行命令と同時に、あるいは先立って判るので、ハードウェア・コストを増大させずに高速化を図ることが可能となる。つまり、本発明において、従来はハードウェア側にて行っていたパイプライン処理の１ステージ分の処理を、コンパイル時やアッセンブル時に、ソフトウェア処理により静的に事前に実現する事に成功している。

【００２３】

本発明の対象となるデータ処理装置としては、準備情報に基づく処理を実行する第２の

実行制御ユニットは、ＦＰＧＡ（Field Programmable Gate Arrays）のようにトランジスタ間の接続を変更可能なアーキテクチャを動的に制御できるものであっても良い。しかしながら、ＦＰＧＡは、ハードウェアをダイナミックに変更するには時間がかかり、また、その時間を短縮するためのハードウェアが必要となる。例えば、ＦＰＧＡの再構成情報を二面以上のＲＡＭに保持し、バックグラウンドで実行する事により、見かけ上短い時間で動的なアーキテクチャ変更を行う方式も可能であるが、もし、数クロック以内にこの再構成を行う事を可能とするためには、考えられる組み合わせの数の再構成情報を全て格納するＲＡＭを実装する必要があり、これは、本質的にＦＰＧＡの再構成時間が大きく掛かるという経済的な問題を一切解決していない。また、ＦＰＧＡが、本来ハードウェアのゲートに注目したマッピングを効率良く実現しようとするために抱えている問題、即ち実用上のＡＣ特性の悪さをについては、当面解決出来そうも無い。

【００２４】

これに対し、本発明において、準備情報として、処理ユニットの入力および／または出力インタフェースを、その処理ユニットの実行時期とは独立して、別に規定し、第２の実行制御ユニットあるいは第２の制御工程において、処理ユニットの入力および／または出力インタフェースを、その処理ユニットの実行時期とは独立して、別に設定することが可能となる。このため、複数の処理ユニットを備えたデータ処理装置においては、第２の実行制御ユニットあるいは第２の制御工程において、これらの処理ユニットによるデータパスの組み合わせを制御することが可能となる。すなわち、第２のフィールドに、データ処理装置に含まれる少なくとも１つの算術論理演算ユニットなどの処理ユニットのインタフェースを規定する命令を記載あるいは記述することにより、データフロー指定を行うことが可能となる。これにより、データパスの独立性を高めることが可能となり結果的にデータフロー指定を別命令プログラムを実行しながら行ったり、アイドル状態にある制御ユニットあるいはデータ処理装置の内部のデータパスを、外部の他の制御ユニットあるいはデータ処理装置において実行されている緊急度の高い処理のために貸し出すことも許す構造を容易に提供可能である。

【００２５】

さらに、準備情報に、処理ユニットの処理内容または回路構成も規定する情報を採用し、第２の実行制御ユニットまたは第２の制御工程により、処理ユニットの処理内容または回路構成も規定することによりさらにフレキシブルにデータパスを構成することができる。

【００２６】

また、第２の実行制御ユニットあるいは第２の制御工程に、レジスタ情報をデコードしてフェッチするなどの算術論理演算ユニットのインタフェースや、他の処理ユニットのインタフェースを規定するスケジューラとしての機能を持たせてデータパスの組み合わせを管理することにより、多種多様なデータ処理に対応することができる。例えば、ある一定時間だけ、マトリックス計算を行い、その後にフィルター処理を行う場合は、予めそれらの処理に必要なデータ処理装置内部の処理ユニット間の接続を指定し、時間を計数するカウンターを使ってこれを実現する事が出来る。計数カウンターを別の比較回路や外部イベント検出器に置き換える事で、より複雑で柔軟性のあるスケジューリング処理を実現可能となる。

【００２７】

個々の処理ユニットにＦＰＧＡのアーキテクチャを採用することが可能である。しかしながら、ハードウェアをダイナミックに変更するには時間がかかり、また、その時間を短縮するためのハードウェアが必要となる。このため、アプリケーションの実行中に処理ユニット内部のハードウェアを動的に制御することは難しい。仮に、これを複数のＲＡＭをバンク構成にして、瞬時に切り換える方式にしたとしても、数クロック～数十クロック単位での切り換えを実現する為には、相当数のバンク構成が必要となり、基本的にＦＰＧＡ内部のマクロセル一つ一つが独立してプログラム構成可能な構造にすると同時に、この切り換えタイミングを検出し、プログラムによる制御機構を持たせる必要がある。しかし、

このような構成に対処することは現状のＦＰＧＡでは不十分である。さらに、対処可能となったとしても、動的に制御するためには切替のタイミングなどを制御するために、本発明にあるような新しい命令制御機構が必要である事を意味する。

【００２８】

このため、本発明においては、処理ユニットとして、特定の内部データパスを備えた回路ユニットを採用することが望ましい。すなわち、ある程度コンパクトなデータパスを備えた処理ユニットをテンプレート的に用意しておき、そのデータパス間の組み合わせを指示してデータフロー型の処理に持ち込むと共に、準備情報あるいは準備命令により、処理ユニットの内部データパスの一部を選択して処理ユニットの処理内容を変更することにより、さらにフレキシブルに、そして短時間にハードウェアを再構成できる。

【００２９】

たとえば、適当な論理ゲートと、この論理ゲートと入出力インタフェースを接続する内部データパスを予め備えたテンプレート的に使用可能な特定のデータパスを備えた処理ユニットは、以下の説明においてはテンプレートと称されている。このような処理ユニットであれば、入出力されるデータの順番を変えたり、論理ゲート間の接続あるいは選択を変えることにより処理ユニットの処理内容を変更できる。そして、トランジスタレベルで回路を再構成するＦＰＧＡに比較すると、予め用意された内部データパスの一部を選択するだけで良いので、短時間で処理内容を変更できる。さらに、予め用意された内部データパスを使用するので、冗長な回路要素は少なく、トランジスタの面積利用効率も高い。したがって、実装密度も高く、経済的である。さらに、高速処理に適したデータパスを構築でき、ＡＣ特性も高い。このため、本発明においては、準備情報により、第２の実行制御ユニットおよび第２の制御工程において、処理ユニットの内部データパスの一部を選択可能とすることが望ましい。

【００３０】

さらに、準備情報に基づき設定された各処理ユニットのインタフェースを保持するスケジュールを管理するように、第２の実行制御ユニットは処理ユニットのインタフェースを管理するスケジューラとしての機能を備えていることが望ましい。

【００３１】

また、準備情報により、複数の処理ユニットにより構成される処理ブロックの入力および／または出力インタフェースを規定できるようにすることが望ましい。複数の処理ユニットのインタフェースを１つ命令で変更可能とすることにより、複数の処理ユニットが関連するデータパスの変更が１命令で処理することができる。したがって、第２の実行制御ユニットあるいは工程では、準備情報に基づき、複数の処理ユニットにより構成される処理ブロックの入力および／または出力インタフェースを変更可能であることが望ましい。

【００３２】

さらに、処理ブロックの入力および／または出力インタフェースを規定する複数のコンフィグレーションデータを格納したメモリを設け、準備情報によりメモリに格納された複数のコンフィグレーションデータの１つを選択し、処理ブロックの入力および／または出力インタフェースを変更できるようにすることが望ましい。データフロー指定命令によりコンフィグレーションデータを指定できるようにすることにより、命令自体は冗長にせずに複数の処理ユニットのインタフェースの変更をプログラムから制御することができる。

【００３３】

さらに、処理ユニットとして算術論理演算ユニットを備えた汎用処理に適した第１の制御ユニットと、処理ユニットとして特定のデータパスを具備する複数のデータフロー処理ユニットを備えた専用処理に適した第２の制御ユニットとを設けることにより、ネットワーク処理や画像処理などの高速性およびリアルタイム性が要求される処理に適したシステムＬＳＩを提供することが可能となる。そして、本発明の命令セットであれば、第１のフィールドに、算術論理演算ユニットを操作する実行命令を記述でき、第２のフィールドに、算術論理演算ユニットおよび／またはデータフロー処理ユニットのインタフェースを規定する準備情報が記述することが可能であり、上記のシステムＬＳＩの制御に適したプロ

グラム製品を提供できる。

【００３４】

従来は、複雑なデータ処理は、専用回路を用意し、その専用回路を用いる専用命令化するしか対応方法が無くハードウェアコストが増大する。これに対し、本発明の命令セットにおいては、実行命令とは独立して第２のフィールドにより論理演算ユニットのインターフェースおよびその処理内容を記述できるので、パイプライン制御やデータパス制御の構造を命令セットの中に取り込むことが可能となる。したがって、本発明は、プログラムカウンタ近傍の並列処理を実行だけでなく、２つ以上オブジェクトの同時擬似実行や２つ以上の関数の同時擬似実行に有効な手段を提供することになる。つまり、従来の命令セットでは、２つ以上のコンテキストの異なるデータ処理やアルゴリズム実行等の、それぞれ離れたプログラムカウンタに基づく処理が同時に起動ができなかったのに対し、本発明の命令セットを用いてデータフローを適当に定義することにより、プログラムカウンタにかかわらずに処理を実行することが可能となる。

【００３５】

したがって、本発明の命令セットを用いると、並列処理に対して、予めアプリケーション側から見て性能向上に有効と思われるデータパスを第２のフィールドを用いてソフトウェアから組み込むことが可能であり、それにより実現されたデータパス（データフロー）を必要に応じて、さらにソフトウェアから命令レベルで起動することができる。このデータパスは、特定の目的に対応したデータ処理だけでなく、一般のステートマシンを起動するような目的にも使用可能なので、極めて自由度が高い。

【００３６】

また、この第２のフィールドの情報により、先行して次命令の準備サイクルを簡単に発生させることが可能となるために、従来はその演算対象をレジスタにせざるを得なかったものが、バッファリングを前提とすればメモリ（シングルポート／デュアルポート）やレジスタファイルで代用可能となる。すなわち、第２のフィールドに、処理ユニットなどに含まれるレジスタまたはバッファとメモリの間の入出力を指示する命令を記述することを可能とし、第２の実行制御ユニットまたは第２の制御工程において、レジスタまたはバッファとメモリの間の入出力を制御する機能を持つようにすれば、実行命令とは独立してメモリに対する入出力を行うことができる。

【００３７】

このことは、１つ１つの命令シーケンスの関連性を高めると同時にハードウェアリソースの競合を事前に回避する事に貢献するので、複数命令の並列同時実行や外部からの割り込み要因への対応を早めることが可能となる。そして、基本的に、メモリをレジスタと見なせるので、高速なタスクスイッチの実現が可能となる。さらに、従来のファーストフェッチのペナルティを消せないキャッシュ・メモリの代わりに、プリローディング型の高速バッファを採用する事も可能となる為、１００％のヒット率を保証しながら一切ペナルティの発生しない高速の組み込みシステムの実現も可能となる。

【００３８】

すなわち、メモリをレジスタとみなせるようにすることにより、割り込み等の複数の非同期処理要求に対し高速対応が可能となり、複雑なデータ処理や連続データ処理への対応を非常にフレキシブルに行うことができる。また、レジスタの対比および復帰に時間がかからないので、タスクスイッチ等への高速対応が極めて簡単である。そして、外部メモリと内部メモリのアクセススピード差の影響を完全に消すことができるので、キャッシュは、ファーストフェッチ・ペナルティの問題を効率良く解決できるといったメリットを得ることができる。したがって、ＣＡＬＬ／ＲＥＴや割り込み処理／ＩＲＥＴを高速で処理することができるので、イベントに対する応答環境を簡単に構築でき、イベントによってデータ処理性能が低下するのを防止できる。

【００３９】

さらに、第１または第２のフィールドを、ＶＬＩＷのように、複数の実行命令または準備命令を記述なフィールドとし、第１または第２の実行制御ユニットが第１または第２の

フィールドに記述された複数の独立した実行命令または準備命令を独立して処理可能な複数の実行制御部を備えているようにすれば、さらにパフォーマンスを向上できる。

【００４０】

そして、本発明にかかる制御ユニットをコアあるいは周辺回路に採用したデータ処理装置を実現することにより、上述したようなメリットを活かし、処理速度が速く、さらに経済的なデータ処理装置を提供できる。

【発明の効果】

【００４１】

本発明においては、再構成可能なハードウェアを有するデータ処理装置の制御方法であって、第１のデータ処理の実行が決定される１または数クロック前に、ハードウェアの少なくとも一部を、第１のデータ処理を実行するように再構成する工程を有するデータ処理装置の制御方法を提供する。これにより、ハードウェアを随時、実行される蓋然性の高い特定のデータ処理に適した構成に変更でき、さらに、複雑なデータ処理であっても、それらのデータ処理に特化した多種多様な専用回路を用いずに、それぞれのデータ処理をフレキシブルに、そして高速に実行できる。

【発明を実施するための最良の形態】

【００４２】

以下に図面を参照して、本発明をさらに詳しく説明する。図１に、本発明にかかる命令セット（命令フォーマット）の構成を示してある。本発明にかかる命令セット（ＤＡＰ／ＤＮＡの命令セット）１０は、第１のフィールドである命令実行基本フィールド（Ｘフィールド）１１と呼ばれる部分と、次の命令実行の効率化を図ることができる第２のフィールドである次命令実行準備サイクル（追加フィールドあるいはＹフィールド）１２と呼ばれる２つのフィールドを備えている。命令実行基本フィールド（Ｘフィールド）１１は、加減演算、論理和、論理積、比較などのデータの演算、および分岐などのその他の各種のデータ処理の内容を指定し、その結果が格納される先（ディスティネーション）を指定する。また、Ｘフィールド１１は、命令長の使用効率を上げるために実際に実行される命令の情報しか含まない。一方、追加フィールド（Ｙフィールド）１２は、同一の命令セットのＸフィールド１１の実行命令とは独立した命令（情報）が記述可能であり、たとえば、次の命令の実行準備サイクルに割当てられる。

【００４３】

さらに詳しく命令セット１０を説明すると、Ｘフィールド１１は、算術論理演算ユニットなどの処理ユニットに対する命令操作あるいは実行命令（Execution ID）を記述する実行命令フィールド１５と、Ｙフィールド１２の有効／無効およびＹフィールド１２で示す準備命令（準備情報）のタイプを示すフィールド（タイプフィールド）１６と、ディスティネーションのレジスタを示すフィールド１７とを備えている。タイプフィールド１６の内容は、Ｙフィールド１２に関連したものであり、Ｘフィールド１１の他のフィールドの内容とは独立して定義できることは上述した通りである。

【００４４】

また、Ｙフィールド１２は、タイプフィールド１６によって規定される準備情報が記述される。このＹフィールド１２に記述される準備情報は、演算または他のデータ処理を実行可能な状態にするための情報であり、図２に具体的な幾つかの例を示してある。先ず、ＴＹＰＥフィールド１６はＸフィールド１１に含まれているが、実行命令フィールド１５とは独立あるいは無関係に記述できる。そして、Ｙフィールド１２には、アドレスＩＤ（ＡＩＤ）２１と、それによって利用目的が規定されるアドレス情報２２、たとえば、アドレス（ＡＤＲＳ）、入出力アドレス（ＡＤＲＳ．ＦＲＯＭ／ＴＯ）などを記述するアドレス情報フィールド２６として利用することができる。このＹフィールド１２に記述されたアドレス情報は、レジスタあるいはバッファとメモリ（レジスタファイルを含む）との間のリードおよびライトに用いられ、ＤＭＡのようにブロック転送も可能な構成になっている。さらに、入出力（Ｒ／Ｗ）だけでなく、分岐命令を実行したときの分岐先を示すアドレス（フェッチアドレス、Ｆ）、並列実行するときのスタートアドレス（Ｄ）などの情報

もアドレス情報としてＹフィールド１２に記述することができる。

【００４５】

また、レジスタタイプの命令、たとえば、算術演算あるいはその他の論理演算命令（ＭＯＶＥ、メモリーリード／ライトなども含む）に対してソース側となるレジスタ（Ｒｅｇ）の情報あるいは即値（イミーディエイト、ｉｍｍ）を規定する情報２３もＹフィールド１２に記述することができる。すなわち、Ｙフィールド１２を以降の実行命令のためのソースを規定するフィールド２７として利用することができる。

【００４６】

さらに、Ｙフィールド１２には、算術論理演算ユニット（ＡＬＵ）あるいは他のデータ処理ユニット、たとえば所定のデータパスを備えたテンプレートのインタフェース（ソース、ディスティネーション）および処理内容の組み合わせを規定する情報２５も記述することが可能である。すなわち、Ｙフィールド１２は、リコンフィグラブルなデータパスなどを、特定のデータ処理を行うために、それらのパイプライン（データフローあるいはデータパス）を定義するためのデータフロー指定命令２５を記述するフィールド２８として利用することができる。もちろん、Ｙフィールド１２には、そのデータフローをスタートする情報および終了する情報を記述することが可能である。したがって、Ｙフィールド１２を用いてリコンフィグラブルなデータパスを定義して生成したデータフローにより、コードＲＡＭからコードをフェッチするプログラムカウンタとは独立した処理を行うことができる。

【００４７】

なお、図１および図２に示した命令セットのフォーマットは、本発明にかかる２つの独立した命令フィールドを備えた命令セットの一例であり、これに限定されないことはもちろんである。たとえば、ＸおよびＹフィールド内でのフィールドの位置は限定されるものではない。また、独立したフィールド、例えば、タイプフィールド１６の位置は、本例に限定される必要はなく、Ｙフィールド１２の先頭に位置させることも可能である。また、Ｘフィールド１１とＹフィールド１２の順番を変えることも可能である。本例においては、実行命令が記述されるＸフィールド１１にＹフィールド１２の情報を含ませることによりＸフィールド１１をデコードすることで、Ｙフィールド１２に準備情報があるか否か、およびその情報の種類を判断できるようにしている。

【００４８】

また、以下ではＸフィールド１１およびＹフィールド１２に実行命令あるいは準備命令が記載あるいは記述された例を説明するが、これらのフィールドに命令を記述せず（ＮＯＰを記述し）、Ｘフィールド１１あるいはＹフィールド１２だけが意味を持つような命令セットも可能である。さらに、Ｘフィールド１１に記述された実行命令にかかるレジスタ情報などのオペランドを備えた準備命令、すなわち、同一命令セット１０のＹフィールド１２に、Ｘフィールド１１の実行命令に対し独立していない準備命令が同時に記述された命令セットも可能である。そして、これらの命令セットを、本発明の、Ｘフィールド１１とＹフィールド１２が独立し、同一命令セット内では無関係となった命令セットと混在してプログラミングすることも可能である。以下では本発明をわかりやすく説明するためにそのような例を具体的には記載していない。しかしながら、Ｘフィールド１１とＹフィールド１２に記述された内容が独立した命令セット１０と、ＸフィールドとＹフィールドに記述された内容が関連した命令セットが混在したプログラム製品あるいはプログラムを記録した記録媒体なども本発明の範囲に含まれる。

【００４９】

図３に、本例の命令セット１０の簡単な例を示してある。ｊ－１番目の命令セット１０であるＴ（ｊ－１）は、そのＸフィールド１１のタイプフィールド１６に、同一の命令セットのＹフィールド１２に３２ビットのイミーディエイトが記述されていることが示されている。そして、その命令セットＴ（ｊ－１）のＹフィールド１２には、イミーディエイトとして「＃００００１２３４Ｈ」が記載されている。次のｊ番目の命令セットＴ（ｊ）には、Ｘフィールド１１の実行命令フィールド１５にＭＯＶＥが記述され、ディスティネ

ーションフィールド１７にレジスタＲ３が記載されている。このため、このｊ番目の命令セットＴ（ｊ）をフェッチすると、制御ユニットのＡＬＵは、前の命令フィールドＴ（ｊ－１）に定義されたイミーディエイト「＃００００１２３４Ｈ」をレジスタＲ３に格納する。

【００５０】

このようにして、本例の命令セット１０（以降では、ｊ番目の命令セット１０を命令セットＴ（ｊ）で示す）では、実行命令が記述された命令セットＴ（ｊ）の前の命令セットＴ（ｊ－１）によりその実行命令の準備が行われる。したがって、命令セットＴ（ｊ）だけでは制御ユニットを構成するＡＬＵが実行する処理内容は判らないが、２つの命令セットＴ（ｊ－１）およびＴ（ｊ）によりＡＬＵが実行する処理内容は一義的に決定される。また、命令セットＴ（ｊ－１）の実行命令フィールド１５には、その命令セットのＹフィールド１２とは独立して命令セットＴ（ｊ－１）の前の命令セットのＹフィールド１２により準備された処理を実行する命令が記述されている。さらに、命令セットＴ（ｊ）のタイプフィールド１６およびＹフィールド１２には、次の命令セットの実行命令フィールドに記述された実行命令の準備をする情報が記述されている。

【００５１】

本例では、ある実行命令がＸフィールド１１に記述された命令セットＴ（ｊ）の直前の命令セットＴ（ｊ－１）のＹフィールド１２に、その実行命令の準備情報（準備命令）が記述されている。すなわち、準備命令のレイテンシーが１クロックの例となっているが、準備情報が記述される命令セットは、直前の命令セットにかぎられるものではない。例えば、複数のＡＬＵを備えた制御ユニットの制御プログラム、あるいは後述するデータフロー制御を目的とする準備命令などであれば直前の命令セットである必要はない。準備命令によってセットされたＡＬＵの状態（環境あるいはインタフェース）あるいはテンプレートの構成が、その準備命令に対応する実行命令を備えた命令セットがフェッチされて実行されるまで保持されるのであれば、実行命令を備えた命令セット１０の数命令前の命令セット１０のＹフィールド１２で準備命令を記述できる。

【００５２】

図４に、図３に示した命令セットによりレジスタとして機能するレジスタファイルあるいはメモリに値が格納される様子を示してある。プロセッサがｊ－１番目の命令セットＴ（ｊ－１）をフェッチして、そのＹフィールド１２の準備命令によりイミーディエイト「＃００００１２３４Ｈ」がプロセッサのＡＬＵのソース側のレジスタＤＰ０．Ｒにラッチされる。そして、プロセッサが次のｊ番目の命令セットＴ（ｊ）をフェッチし、そのＸフィールド１１の実行命令であるＭＯＶＥを実行するサイクルでバッファ２９ｂにストアされる。その後、メモリまたはレジスタファイル２９ａのレジスタＲ３のアドレスにバッファ２９ｂの値が格納される。したがって、格納先がレジスタではなくメモリであっても、本例の命令セット１０を用いると、準備情報に基づく処理を実行命令に先立って行うことにより、実行命令のサイクルでデータをロードあるいはストアすることができる。

【００５３】

図５に、本例の命令セット１０により処理内容が記述されたプログラムを実行可能な制御ユニット３０を備えたプロセッサ（データ処理装置）３８の概略構成を示してある。本例の命令セット１０を具備したマイクロコードあるいはマイクロプログラム１８はコードＲＯＭ３９に記憶されている。制御ユニット３０は、コードＲＯＭ３９からマイクロプログラムの命令セット１０をプログラムカウンタによって随時フェッチするフェッチユニット３１と、フェッチされた命令セット１０のＸフィールド１１をデコードしてＡＬＵ３４の処理内容を決定あるいはアサートすると共に、ＡＬＵ３４の論理演算結果をディスティネーションのレジスタ３４ｄを選択してラッチする機能を備えた第１の実行制御ユニット３２を備えている。

【００５４】

さらに、制御ユニット３０は、フェッチされた命令セット１０のＹフィールド１２をＸフィールド１１のタイプフィールド１６の情報に基づいてデコードし、演算処理ユニット

（ＡＬＵ）３４のソース側のレジスタ３４ｓを選択する機能を備えた第２の実行制御ユニット３３を備えている。この第２の実行制御ユニット３３は、タイプフィールド１６の情報を除き、Ｙフィールド１２の命令あるいは情報をＸフィールド１１の内容とは独立して解釈することができる。第２の実行制御ユニット３３は、さらに、Ｙフィールド１２に記述された情報がデータフローを規定するものであれば、ＡＬＵ３４のソース側およびディスティネーション側の選択あるいは設定、すなわち、ＡＬＵ３４のインタフェースを決定し、さらに、その状態を所定のクロックあるいは解除の指示があるまで連続的に保持する機能も備えている。また、Ｙフィールド１２の情報がデータフローを規定する場合は、この第２の実行制御ユニット３３は、さらに、ＡＬＵ３４の処理内容も決定し、その状態を所定の期間保持する。

【００５５】

したがって、第１の実行制御ユニット３２は、Ｘフィールド１１の実行命令をデコードし、その実行命令の演算または他のデータ処理が実行できるように予め設定された処理ユニットにより演算または他のデータ処理を進める第１の制御工程を行う。一方、第２の実行制御ユニット３３は、Ｙフィールド１２の準備情報をデコードし、第１の実行制御ユニット３２の実行内容、およびこの第１の実行制御ユニット３２で行われる第１の制御工程とは独立に、処理ユニットの状態を演算または他のデータ処理が実行できるように設定する第２の制御工程を行う。

【００５６】

本例の制御ユニット３０は、さらに、このような実行制御ユニット３２および３３と、ＡＬＵ３４の組み合わせを複数備えており、これらによって様々な処理が実行できるようになっている。したがって、本例の制御ユニット３０をコアあるいは周辺回路として画像データを高速で処理するようなＤＳＰ、汎用のデジタル処理を高速で行えるＣＰＵあるいはＭＰＵなどを構成することが可能である。

【００５７】

図６ないし図９に、本例の制御ユニット３０で実行するプログラムの一例を示してある。図６に示したサンプルプログラム４１は、従来のＣＰＵあるいはＤＳＰで実行可能なように作成した例である。このプログラムは、＃ＳＴＡＲＴのアドレスから始まるテーブルから最も大きな値を抽出し、最終データであることを示す＃ＥＮＤを検出すると終了するプログラムである。

【００５８】

図７に記載したプログラム４２は、図６と同じ処理を本発明にかかる命令セットを実行可能な制御ユニット３０に適したプログラムに変換したものであり、２命令を１つの命令セットで実行できる例を示してある。図７に示したプログラムは、コンパイラを通して本発明にかかる命令セットの実行プログラムに変換され、制御ユニット３０で実行される。

【００５９】

図８にコンパイルされた本発明の命令セット１０を有するプログラム４３を示してあり、このような命令セット１０を有するプログラム製品１８がＲＯＭ３９、ＲＡＭあるいは他の適当なデータ処理装置で読取可能な記録媒体に記憶されて提供される。また、ネットワーク環境で交換される伝送媒体にプログラム製品４３あるいは１８を埋め込んで流通することも可能である。このプログラム４３と、プログラム４２とを比較すると判るように、第１の番目の命令セット１０のＹフィールド１２で２番目の命令セット１０の実行命令１５の準備が行われる。すなわち、タイプフィールド１６に準備情報としてイミーディエイトがＹフィールド１２に記述されていることが示されており、Ｙフィールド１２をデコードした第２の実行制御ユニット３２によりイミーディエイトがＡＬＵ３４のソースとなるキャッシュあるいはレジスタに提供される。そして、２番目の命令セット１０を実行するときは、その実行命令を行う準備が整ったＡＬＵ３４に対し実行命令１５を行うことができる。すなわち、ディスティネーションフィールド１７に規定されたレジスタに対し、実行命令フィールド１５のＭＯＶＥ命令を単に実行するだけになる。

【００６０】

同様に、２番目の命令セット１０のＹフィールド１２には、次の３番目の命令セット１０の実行命令フィールド１５の実行命令、ＭＯＶＥおよびＡＤＤの準備情報として、ソース側のレジスタを設定する命令が記述されている。このため、タイプフィールド１６にはレジスタとイミーディエイトがＹフィールド１２に記述されていることが定義されている。

【００６１】

本例のプログラム４３は、３番目以降の命令セット１０も上記と同様であり、３番目の命令セット１０のタイプフィールド１６およびＹフィールド１２に、次の４番目の命令セット１０の実行命令１５の準備情報が記述されている。４番目の命令セット１０の実行命令１５は、比較処理（ＣＭＰ）と、条件分岐処理（ＪＣＣ）である。このため、３番目の命令セット１０では、そのタイプフィールド１６とＹフィールド１２とにより、次の実行命令１５で比較対象となるレジストＲ１と＃ＥＮＤのイミーディエイトの値（＃ＦＦＦＦＦＦＦＨ）と、分岐先＃ＬＮＥＸＴのアドレス（＃０００００５００Ｈ）が準備情報として記述されている。したがって、４番目の命令セット１０の実行命令１５を実行するときは、比較回路として動作する演算処理ユニット３４に入力値がセットされているので、そのサイクルで比較結果を出す。また、ジャンプアドレスがフェッチアドレスレジスタにセットされているので、実行命令１５の条件分岐では、比較結果によって、そのサイクルで遷移先の命令セット１０をフェッチすることができる。

【００６２】

４番目の命令セット１０では、そのタイプフィールド１６およびＹフィールド１２により、次の５番目の命令セット１０の実行命令１５である比較処理（ＣＭＰ）と条件分岐処理（ＪＣＣ）の準備情報として、比較するレジスタの情報（Ｒ０およびＲ１）と、分岐先＃ＬＯＯＰのアドレス（＃０００００４９６Ｈ）が記述されている。したがって、４番目の命令セットと同様に、５番目の命令セット１０を実行すると、すでにＸフィールド１１に記述されたＣＭＰとＪＣＣを演算処理ユニット３４で実行するインタフェースは整っているので、そのサイクルで比較および条件分岐処理が実行される。

【００６３】

その５番目の命令セット１０のＹフィールド１２には、次の６番目の命令セット１０の実行命令である移行処理（ＭＯＶＥ）および分岐処理（ＪＭＰ）の準備情報として、ソース側のレジスタ情報（Ｒ１）と遷移先＃ＬＯＯＰのアドレスが記述されている。したがって、６番目の命令セット１０を実行すると、そのサイクルでデータをディスティネーションのレジスタＲ０に格納し、遷移先の＃ＬＯＯＰのアドレスから命令をフェッチすることができる。

【００６４】

このように、本発明の命令セットによれば、実行命令と、その実行命令を行うためのインタフェースなどを記述した準備命令とを分離することができ、さらに、準備命令を実行命令に先立ってフェッチされる命令セットに記述して処理することができる。したがって、各々の命令セットに記述された実行命令を行うときは、ＡＬＵ３４のソース側にデータがリードされているので純粋に算術命令だけを行うようになる。このため、ＡＣ特性が良く、実行周波数特性が向上する。さらに、実行命令に対する前後の差はあるが、従来のパイプラインと同様に、命令フェッチ、レジスタデコード、処理実行などを段階的に行うことが可能であり、スループットも向上できる。

【００６５】

また、本例のプログラムは２命令を１命令セットに記述できるようになっているので、ＶＬＩＷと同様にプログラムカウンタの近傍の複数の命令を並列実行することにより処理速度を向上することができる。

【００６６】

さらに、４番目の命令セットの実行命令フィールド１５には条件分岐が記述されており、その分岐先のアドレスは、この命令セットに先行する３番目の命令セットのＹフィールド１２に記述されている。したがって、４番目の命令セットを実行する際に、あるいはそ

れに先立ってフェッチレジスタに分岐先のアドレスをセットし、分岐条件が成立したときにペナルティなく分岐先の命令セットをフェッチあるいは実行することができる。さらには、分岐先の命令をプリフェッチしておくことも可能であり、分岐先の実行命令を実行する準備を事前に整えておくことも可能となる。したがって、分岐先の命令であっても１クロックの無駄もなく実行することが可能であり、１クロック単位で処理を正確に定義することができる。

【００６７】

図９には、さらに、本発明の命令セット１０のＹフィールド１２を用いてデータフローを定義し、そのデータフローにより上記と同様の処理を行う、本発明のプログラム４４を示してある。このプログラム４４に記述されたデータフロー指定命令２５の内、ＤＦＬＷＩは、データフローの初期設定を行う命令であり、ＤＦＬＷＣはデータフロー（データパス）を構成する演算処理ユニット３４の接続情報（インタフェースの情報）および処理内容を規定する命令である。また、ＤＦＬＷＴはデータフローの終了条件を規定する命令であり、最後に、このようにして定義されたデータフローにデータを入力して処理を行うＤＦＬＷＳが記述されている。これらのデータフロー指定命令２５は、Ｙフィールド１２に準備情報として記述され、第２の実行制御ユニット３３でデコードされ、処理ユニット３４でデータ処理を行うための構成（コンフィグレーション）がセットされる。

【００６８】

図９に示した本例のプログラム４４を実行する際には、プログラムのデータフロー指定にしたがって第２の実行制御ユニット３３が、第２の制御工程として処理ユニットの入力および／または出力インタフェースを、その処理ユニットの実行時期とは独立して設定し、さらに、処理ユニットの処理内容も規定する処理を行う。また、第２の実行制御ユニット３３は、スケジューラ３６としても機能し、第２の制御工程として各処理ユニットのインタフェースを維持するスケジュールを管理する。

【００６９】

このため、図１０に示すように、スケジューラ３６として機能する第２の実行制御ユニット３３により、３つの演算処理ユニット３４のインタフェース（入出力）と、その処理内容が規定され、その状態あるいはコンフィグレーションが終了条件が成立するまで保持される。したがって、これらの演算処理ユニット３４により構成されるデータフローあるいはデータパスにより、プログラムカウンタとは独立して次々と図６に示した処理と同じ処理が進行する。すなわち、データフロー指定を行うことにより、３つの演算処理ユニット３４によって制御ユニット３０の中に、その処理のための専用回路が事前に設けられた状態となり、プログラムカウンタの制御から外れて最大値を求める処理を実行することができる。そして、ＤＰ１．Ｒ１と＃ＥＮＤが同じになることをＤＰ１．ＳＵＢとしての機能を果たすＡＬＵ３４で判断するとデータフローが終了する。

【００７０】

したがって、図９から判るように、データフローを定義することにより分岐命令を用いずに図６あるいは図７に記載されたプログラムを同じ処理を実行することができる。このため、汎用の制御ユニット３０でありながら、専用回路を備えた制御ユニットと同様に特定の処理を非常に高速に効率良く行うことが可能となる。

【００７１】

本発明にかかる命令セットおよび制御ユニットにより、様々な処理を行うデータフローあるいは疑似データフローを制御ユニットに設けることができる。これらのデータフローはテンプレートとして他の処理あるいは他のプログラムにも適用できるものであり、ソフトウェアを用いてハードウェアを随時、特定のデータ処理に適した構成に変更でき、それを他のプログラムあるいは他のハードウェアにおいても実現できることを意味する。そして、このようなデータフローを複数設定することも可能であり、マルチコマンドストリームをソフトウェアを用いて制御ユニットの中に定義することができる。したがって、複数の処理を並列実行することが極めて簡単となり、その実行内容をプログラミングにより自由に制御することができる。

【００７２】
図１１に、本例のＸフィールド１１およびＹフィールド１２を備えた命令セット１０によりデータフローを定義することができる複数の処理ユニット（テンプレート）を備えたデータ処理装置の概略構成を、システムＬＳＩ５０のイメージで示してある。このシステムＬＳＩ５０は、データの処理動作を行うプロセッサ領域５１と、そのプロセッサ領域５１の処理を制御するプログラム１８が格納されたコードＲＡＭ５２と、その他の制御情報あるいは処理用のデータを記憶し、さらに、一次的なワーク領域ともなるデータＲＡＭ５３とを備えている。プロセッサ領域５１は、プログラムコードをフェッチするフェッチユニット（ＦＵ）５５と、多目的な処理を行う汎用的なデータ処理ユニット（多目的ＡＬＵ、第１の制御ユニット）５６と、データフロー方式でデータを処理することができるデータフロー処理ユニット（ＤＦＵ、第２の制御ユニット）５７とを備えている。

【００７３】
本例のＬＳＩ５０は、１つの命令セット１０に１組のＸフィールド１１およびＹフィールド１２を含んだプログラムコードをデコードして処理を実行できるようになっている。このため、ＦＵ５５は、フェッチした命令セット１０のＸフィールド１１の命令を格納できるフェッチレジスタ（ＦＲ（Ｘ））６１ｘと、Ｙフィールド１２の命令を格納できるフェッチレジスタ（ＦＲ（Ｙ））６１ｙとを備えている。また、ＦＲ（Ｘ）６１ｘにラッチされた命令をデコードするＸデコーダ６２ｘと、ＦＲ（Ｙ）６１ｙにラッチされた命令をデコードするＹデコーダ６２ｙとを備えている。また、これらのデコーダ６２ｘおよび６２ｙのデコード結果により次の命令セットのアドレスが格納され、プログラムカウンタとして機能するレジスタ（ＰＣ）６３を備えている。したがって、コードＲＡＭ５２に格納されているプログラムの所定のアドレスから次の命令セットを随時フェッチすることができる。

【００７４】
本例のＬＳＩ５０においては、Ｘデコーダ６２ｘが上述した第１の実行制御ユニット３２としての機能を果たす。したがって、Ｘデコーダ６２ｘが、命令セット１０のＸフィールド１１に記述された実行命令に基づき、本発明の第１の制御工程を実行する。また、Ｙデコーダ６２ｙが第２の実行制御ユニット３３としての機能を果たす。したがって、Ｙデコーダ６２ｙが、命令セット１０のＹフィールド１２に記述された準備情報に基づき、本発明の第２の制御工程を実行する。すなわち、本例のデータ処理装置の制御においては、フェッチユニット５５において、本発明の命令セットをフェッチする工程が行われ、Ｘデコーダ６２ｘにおいて、第１のフィールドの実行命令をデコードし、その実行命令の演算または他のデータ処理が実行できるように予め設定された処理ユニットにより当該演算または他のデータ処理を進める第１の制御工程が行われ、Ｙデコーダ６２ｙにおいて、第１の制御工程とは独立して、第２のフィールドの準備情報をデコードし処理ユニットの状態を演算または他のデータ処理が実行できるように設定する第２の制御工程が行われる。

【００７５】
多目的ＡＬＵ５６は、図５で説明した演算ユニット（ＡＬＵ）３４と、このＡＬＵ３４の入出力のデータを格納するレジスタ群３５とを備えている。ＦＵ５５でデコードされた命令がＡＬＵ３４の実行命令と準備情報であれば、Ｘデコーダ６２ｘでデコードされた信号φｘと、Ｙデコーダ６２ｙでデコードされた信号φｙは多目的ＡＬＵ５６に供給され、上記にて説明したようにＡＬＵ３４における処理が実行される。

【００７６】
ＤＦＵ５７は、様々な処理を行うデータフローあるいは疑似データフローを構成するための複数のテンプレート７１が配置されたテンプレート領域７２を備えている。それぞれのテンプレート７１は、図９および図１０に基づき説明したように、演算処理ユニット（ＡＬＵ）などのような特定のデータパスあるいはデータフローとしての機能を備えている処理ユニット（処理回路）である。そして、Ｙフィールド１２に準備情報として記述されたデータフロー指定命令２５をＹデコーダ６２ｙがデコードし、その信号φｙにより、ＤＦＵ５７の処理ユニットであるテンプレート７１それぞれのインタフェースと処理内容を

規定することができる。

【００７７】

したがって、これらのテンプレート７１の接続および処理内容をＹフィールド１２に記述したデータフロー指定命令２５によって変更することが可能である。このため、これらのテンプレート７１の組み合わせにより、テンプレート領域７２に特定のデータ処理に適したデータパスをプログラム１８によりフレキシブルに構成することが可能となる。したがって、プロセッサ５１の中に、特定の処理のための専用回路が設けられた状態となり、そこでの処理をプログラムカウンタの制御から外れて実行することができる。すなわち、データフロー指定命令２５によりテンプレート７１の入出力と処理内容を変更することができるので、本例のプロセッサ５１はソフトウェアを用いてハードウェアを随時、特定のデータ処理に適した構成に変更することができる。

【００７８】

図１２（ａ）に示したように、本例のプロセッサ５１のＤＦＵ５７で入力データφｉｎに処理を施して出力データφｏｕｔにする場合、たとえば、図１２（ｂ）に示すように、テンプレート１－１、１－２および１－３を直列に繋いであるデータ処理を行うようにテンプレート７１のインタフェースをデータフロー指定命令２５で設定することができる。同様に、テンプレート領域７２の他のテンプレート７１に対してもそれらのインタフェースをセットして複数のテンプレート７１を適当に組み合わせてデータパスあるいはデータフローを構成することが可能であり、テンプレート領域７２に入力データφｉｎの処理に適した専用処理ユニットあるいは専用データパス７３を複数個、プログラム１８により随時構築することができる。

【００７９】

一方、入力データφｉｎに対する処理が変わったときは、図１２（ｃ）に示すように、データフロー指定命令２５によりテンプレート７１の間の接続を変えることが可能である。すなわち、データフロー指定命令２５をＹデコーダ６２ｙがデコードし、該当するテンプレート７１のインタフェースを変更することができる。このようなＹデコーダ６２ｙの制御（第２の制御工程）により、テンプレート１－１、２－ｎおよびｍ－ｎを直列に接続して、他の異なる処理を実行するのに適した１つあるいは複数のデータパス７３をテンプレート領域７２に構築することが可能である。

【００８０】

また、テンプレート７１を単独で、あるいは複数のテンプレート７１を組み合わせて構成された処理ユニットは、並列して実行される他の処理あるいは他のプログラムに割り当てることも可能である。複数のプロセッサ５１が適当なバスで接続されていれば、他のプロセッサ５１が主として行っているデータ処理のためにテンプレート７１を組み合わせたトレイン（データパス）７３を構成することも可能であり、テンプレート７１というデータ処理資源を極めて有効に活用することができる。

【００８１】

さらに、ＡＮＤやＯＲなどの単純な論理ゲートの実現をもカバーする目的のＦＰＧＡとは異なり、本発明に係るテンプレート７１は、ＡＬＵなどとしての機能あるいは論理ゲートを基本的に備えた特定のデータパスを内部に実装する、より高いレベルのデータ処理ユニットである。そして、データフロー指定命令２５により、テンプレート７１のインタフェースを定義する、あるいは再定義することにより、それらの組み合わせを変えて特定の処理に適したさらに大きなデータパスを構成している。さらに、データフロー指定命令２５によりテンプレート７１で実行する処理内容を定義できるが、その際も、テンプレート７１の内部のＡＬＵあるいは他の論理ゲートなどの接続を変更することで、テンプレート７１の内部データパスの一部を選択する形で、テンプレート７１で実行する処理内容を定義するようにしている。

【００８２】

したがって、本例のテンプレート７１が複数配置されたＤＦＵ５７のハードウェアを特定のデータ処理に適した構成に変更するときには、ＦＰＧＡのようにチップ全体を、ある

いは限定された論理ブロック単位でもマッピングしなおす必要はなく、テンプレート７１あるいはテンプレート領域７２に予め設けられたデータパスを切り替えたり、それらの一部を選択することによりに、予め用意されたＡＬＵあるいは論理ゲートを用いて所望のデータパスを実現することができる。すなわち、テンプレート７１の内部では論理ゲートのコネクションを必要な範囲で設定しなおし、テンプレート７１の間でもそのコネクションを必要な範囲で設定し直すだけでよい。このため、極めて短時間に、クロック単位で、ハードウェアを特定のデータ処理に適した構成に変更することができる。

【００８３】

さらに、論理ゲートが内蔵されていないＦＰＧＡでは、極めて汎用的である反面、特定のアプリケーションの機能を実現するロジック回路を形成するためには無駄となる配線も多く、冗長で信号経路も短くはならない。したがって、実行するアプリケーションに特化したＡＳＩＣに対して実装面積が大きくなり、また、ＡＣ特性も劣化する。これに対し、予め適当な論理ゲートを内蔵している本例のテンプレート７１を採用したプロセッサ５１では、ＦＰＧＡのように膨大な無駄な領域が発生するのを防止でき、ＡＣ特性も改善することができる。したがって、テンプレート７１をベースとした本例のデータ処理ユニット５７は、ハードウェアをプログラムで変更可能なリコンフィグラブルな構成の処理装置であり、ＦＰＧＡを採用した処理装置に対し、より高いレベルでソフトウェアのフレキシビリティとハードウェアの高速性とを備えたデータ処理装置を提供することができる。

【００８４】

そして、本例のテンプレート７１は、適当な論理ゲートを予め内蔵しているので、特定のアプリケーションの処理を実現するために必要な論理ゲートを適当な実装密度で実現することができる。このため、テンプレート７１を用いたデータ処理ユニットは経済的である。また、ＦＰＧＡでデータ処理装置を構成した場合には、実装密度の低下をカバーするために、論理を再構成するプログラムのダウンロードを頻繁に行うことを検討する必要があり、そのための時間も処理速度が低下する原因となる。これに対し、本例のテンプレート７１を用いたプロセッサ５１では、実装密度が高いので、実装密度の低下をカバーする必然性は減少し、そのためにハードウェアを再構成する要求は少なくなる。そして、ハードウェアの再構成もクロック単位で制御することができる。これらの点でも、ＦＰＧＡをベースとしたリコンフィグラブルな処理装置と異なり、ハードウェアをソフトウェアにより再構築できる処理装置であって、コンパクトで実行速度の速いデータ処理装置を提供することができる。

【００８５】

さらに、図１１に示したＤＦＵ５７は、テンプレート領域７２に配置されたテンプレート７１のインタフェースおよび処理内容（以降においてはコンフィグレーションデータ）を一括して定義あるいはセットすることができるコンフィグレーションレジスタ（ＣＲＥＧ）７５と、そのＣＲＥＧ７５にセットする複数のコンフィグレーションデータＣｉ（ｉは適当な整数を示す、以下においても同様である）を記憶したコンフィグレーションＲＡＭ（ＣＲＡＭ）７６を備えている。そして、データフロー指定命令２５として「ＤＦＳＥＴ　Ｃｉ」といった命令が用意されており、Ｙデコーダ６２ｙがこの命令をデコードすると、ＣＲＡＭ７６に記憶されているコンフィグレーションデータＣｉの中から所望のデータがＣＲＥＧ７５にロードされる。その結果、テンプレート領域７２に配置された複数のテンプレート７１のコンフィグレーションを一括して変更できる。あるいは、複数のテンプレート７１からなる処理ブロック単位でそのコンフィグレーションを変更することができる。

【００８６】

また、ＤＦＬＷＩあるいはＤＦＬＷＣといった上記のようなデータフロー指定命令２５をＹデコーダ６２ｙがデコードすることにより、個々のテンプレート７１のコンフィグレーションを設定あるいは変更することも可能である。したがって、本例のＤＦＵ５７では、多くの情報が必要となる複数のテンプレート７１のコンフィグレーションを１命令で変更することが可能であり、命令効率がよく、さらに、再構成のために消費される時間が短

縮されている。

【００８７】

さらに、本例のＤＦＵ５７は、ＣＲＡＭ７６にブロック単位でコンフィグレーションデータをダウンロードするコントローラ７７を備えている。また、データフロー指定命令２５として「ＤＦＬＯＡＤ　ＢＣｉ」が用意されており、Ｙデコーダ６２ｙがこの命令をデコードすると、データＲＡＭ５３などに予め用意されている多数のコンフィグレーションデータ７８の中から、進行中の処理あるいは今後発生するであろう処理のためのコンフィグレーションデータＣｉを予めコンフィグレーションメモリであるＣＲＡＭ７６にダウンロードしておくことができる。このような構成によりＣＲＡＭ７６に小容量の高速な連想メモリなどを採用することが可能となり、さらに短時間でハードウェアをフレキシブルに変更することができる。

【００８８】

図１３に、テンプレート７１の一例を示してある。このテンプレート７１は、ＤＦＵ５７に用意されたデータフローＲＡＭ（ＤＦＲＡＭ）７９を介して他のテンプレート７１とデータを交換することができる構成となっており、Ｉ／Ｏインタフェース８１を介して他のテンプレート７１の処理結果が入力キャッシュ８２ａ～８２ｄに入力され、処理された結果が出力キャッシュ８３ａ～８３ｄに出力される。このテンプレート７１は、これらの入力キャッシュ８２ａ～８２ｄに各々ストアされたデータＡ、Ｂ、ＣおよびＤに対し以下の処理を実行し、演算結果は出力キャッシュ８３ｂに、比較した結果は出力キャッシュ８３ｃにストアすることができるデータパス８８を備えている。このテンプレート７１の処理結果は、再びＩ／ＯインタフェースおよびＤＦＲＡＭ７９を介して他のテンプレートに出力される。

```
IF A == ?
  THEN (C+B)==D
  ELSE (C-B)==D        ・・・（Ａ）
```

このテンプレート７１は、独自のコンフィグレーションレジスタ８４を備えており、このレジスタ８４に格納されるデータによって複数のセレクタ８９を制御し、制御部８５、加算器８６、比較器８７などの論理ゲートに入力する信号を選択することができる。したがって、テンプレート７１は、コンフィグレーションレジスタ８４のデータを変更することにより、データパス８８の一部を用いた処理も可能であり、たとえば、制御部８５を用いずに、以下のような処理を実行させることも可能である。

【００８９】

```
  (B+C)==D
  (B-C)==D        ・・・（Ｂ）
```

また、同様にコンフィグレーションレジスタ８４のデータを変えることにより、このテンプレート７１は、データパス８８の一部を用いて、制御部８５による条件判定回路、加算器８６を用いた加減演算回路、比較器８７を用いた比較回路としても使用することができる。これらの論理ゲートはテンプレート７１に予め作りこまれた専用回路で構成されているので、回路構成としても、処理時間としても無駄がない。そして、入力および出力データのコンフィグレーションは、コンフィグレーションレジスタ８４によって制御されるインタフェース８１により変更することが可能であり、所望のデータ処理を行うデータフローの全部あるいは一部を、本例のテンプレート７１で処理することができる。

【００９０】

このテンプレート７１は、さらに、独自のコンフィグレーションレジスタ８４のデータを上述したＣＲＥＧ７５からのデータと、ＦＵ５５のＹデコーダ（ＹＤＥＣ）６２ｙからのデータのいずれに基づいても書き換えることが可能であり、その選択はＹデコーダ６２ｙからの信号により制御することができる。すなわち、上述したようなテンプレート７１のコンフィグレーションは、データフロー指定命令２５に基づきＹデコーダ６２ｙあるいはこのＹデコーダ６２ｙで実行される第２の制御工程によって行うことができる。さらに、ＤＦＳＥＴ命令などによりＣＲＡＭ７６に記憶されたコンフィグレーションデータＣｉ

にしたがって、他のテンプレートと共にコンフィグレーションを変えてハードウェア構成を変更することも可能である。また、データフロー指定命令２５によりコンフィグレーションレジスタ８４のデータを設定できるので、テンプレート７１の特定のデータパス８８を部分的に選択して使用することも可能である。

【００９１】

このため、テンプレート７１を個別でもグループあるいはブロック単位でもデータフロー指定命令２５によってコンフィグレーションを変え、プロセッサ５１のデータパスをフレキシブルに構成することができる。

【００９２】

テンプレート７１の構成は本例に限定されるものではなく、他のデータ処理を実現可能なように論理ゲートを組み合わせた、適当な種類と数のテンプレートを用意しておくことにより、それらの組み合わせを変えたり、処理内容の一部を変更することにより、多くのデータ処理をテンプレート７１を組み合わせたデータパスにより処理することができる。すなわち、本発明によれば、ある程度コンパクトなデータパスを幾種類かのテンプレートとして用意しておき、そのデータパス間の組み合わせを指示して、データフロー型の処理に持ち込むことにより高性能化を図ることが可能である。そして、テンプレートでは対応できない処理は、プロセッサ５１の多目的ＡＬＵ５６の機能を用いて実行することが可能である。さらに、本例の多目的ＡＬＵ５６は命令セット１０のＹフィールド１２に記述された準備命令により分岐などにより発生するペナルティを最小限に止められるようになっている。このため、本例のプロセッサ５１を搭載したシステムＬＳＩ５０により、プログラムで処理を記述するのと同様に柔軟にハードウェアを変更し、高速処理あるいはリアルタイム処理が可能な高性能のＬＳＩを提供することができる。また、アプリケーションの変更や仕様変更などに対して柔軟に対応でき、仕様変更などに伴い処理性能が低下することも防止できる。

【００９３】

システムＬＳＩ５０を開発あるいは設計する時点で、システムＬＳＩ５０を用いて実行するアプリケーションの概要が判明している場合には、そのアプリケーションの処理に適した構成のテンプレートを中心にテンプレート領域７２を構成することが可能であり、より多くのデータ処理をデータフロー型の処理で実行し、処理性能を高めることが可能である。汎用的なＬＳＩを提供する場合には、浮動小数点演算、乗除算、画像処理などの汎用のアプリケーションで多く発生する処理に適したテンプレートを中心にテンプレート領域７２を構成することが可能である。

【００９４】

このように、本発明にかかる命令セットおよびデータ処理装置により、様々な処理を行うデータフローあるいは疑似データフローを備えたＬＳＩを提供することが可能であり、ソフトウェアを用いてデータフローを実行するハードウェアを随時、特定のデータ処理に適した構成に変更できる。また、上記に説明した、テンプレートの組み合わせによりデータフロー型の処理を実行するアーキテクチャ、すなわち、ＤＦＵ５７あるいはテンプレート領域７２は、Ｘフィールド１１およびＹフィールド１２を備えた命令セット１０とは独立して、制御ユニットあるいはプロセッサなどのデータ処理装置に組み込むことが可能である。そして、ＦＰＧＡよりも高速処理が可能であり、ハードウェアの変更に係る時間も短く、ＡＣ特性も良いデータ処理装置を提供することができる。

【００９５】

また、本例のＤＦＵ５７あるいはテンプレート領域７２を、従来型の汎用の組込プロセッサ、すなわち、ニーモニックなコードで動作するプロセッサと共に組み込んでシステムＬＳＩを構成することも可能であり、テンプレート７１で対応できない処理は、汎用のプロセッサで処理することができる。しかしながら、従来のプロセッサでは、分岐のペナルティや、演算処理のためのレジスタを準備するためにクロックを消費するなどの問題があることは上述した通りであり、本例のＸ－Ｙフィールドを備えた命令セット１０をデコードして実行できるプロセッサ５１のような形態が望ましい。

【００９６】
さらに、本例のプロセッサ５１および命令セット１０であれば、Ｙフィールド１２を用い、他の処理と並列して、ＤＦＵ５７のコンフィグレーションをデータ処理を実行する前に設定あるいは変更することが可能であり、処理効率およびプログラム効率の面で優れている。従来のニーモニックな命令コードと、データフロー型の命令コードとを１つの命令セットに記述することによりプログラム効率を高めることも可能である。しかしながら、本例の命令セット１０のＹフィールド１２の機能は、データフロー型の命令コードを記述するだけでないことは上述したとおりである。
【００９７】
また、本発明に係るプロセッサは、Ｙフィールド１２により実行に先立って物理的なデータパスの構成を変えることができる。これに対し、従来のプロセッサでは、複数のマルチプロセッサ間の接続方法が、共有メモリ等を通す方法しか存在せず、アイドル状態のプロセッサが存在しても、その内部のデータ処理ユニットを外部から利用する方法が無かった。本発明にかかるデータ処理装置においては、適当なデータフローを設定することにより、余っているハードウェアを他の制御ユニットあるいはデータ処理装置により使用するといったことも可能となる。
【００９８】
さらに、副次的な効果として、命令実行シーケンスの効率化と内部データパスの独立性の確保と自由度（流用度）の向上により、本発明にかかる制御ユニットあるいはそれを用いたプロセッサにおいては、実行するハードウェアに余裕さえあれば、全く性質の異なるコンテキストの命令シーケンスを同時に供給しても問題無く実行することが可能となる。
【００９９】
更に、現在、ハードウェアとソフトウェアの強調設計によるメリットが盛んに指摘されるようになったが、本発明による命令セットおよび制御ユニットを採用することにより、ユーザ側の要求するアルゴリズムやデータ処理を許されるハードウェア・コストでどう効率良く経済的に実現可能かという事に対しする１つの回答を与えることができる。例えば、ハードウェア・コストを最小に抑制しながら、性能向上に貢献可能なデータパス（データフロー）を、過去のデータパスに関する構成結果情報である本発明にかかる命令セット（旧ＤＡＰ／ＤＮＡ）のデータ情報と、その後に追加されるハードウェア構成情報およびデータ処理を実行するシーケンス情報から新しいタイプの組み合わせ結果、すなわち、新しいデータフローを定義するソフトウェアを導き、極めて無駄の少ない最適解を提供することが可能となる。
【０１００】
また、従来は、ハードウェア構成が要素化され難いために、その相互の組み合わせ自体の柔軟性が無く、基本的には、性能を上げるために１つ新規のデータパスを追加するというようなやり方が主流であった。そして、性能向上のための情報蓄積の点でも、実際にそれを実現する上で必要となるハードウェア情報の追加という観点でも、数値化し難くデータベース化することは困難であった。これに対し、本発明によれば、ある程度コンパクトなデータパスをいくつかテンプレート的に用意しておき、そのデータパス間の組み合わせを指示して、データフロー型の処理に持ち込むことにより高性能化を図ることが可能である。そして、極めて細かい単位でのハードウェアとソフトウェアとの連携の見積もりが容易となる。また、ハードウェアとソフトウェアのトレードオフ情報を蓄積することも可能で、データパス単位でその組み合わせの可能性が、処理性能に対する貢献度と密接に結びつくことになる。したがって、ハードウェアとソフトウェアの緊密な実行性能データや処理要求に応じた性能コストの正確な見積もりを蓄積することが可能となる。もちろん、これらのデータパスは主要な処理あるいは汎用的な処理の実行を停止させないで実現することも可能となるため、性能要求に対して、何をどれだけどのように追加すれば、どのような結果が期待出来るということを、純粋に過去に蓄積された本発明にかかる命令セットおよびハードウェアのデータから予測する事が可能とする。
【０１０１】

これは、現在行われている設計コストや仕様策定コストの著しい低減に貢献するだけで無く、次の新しい設計に対して、新規に追加すべきハードウェアとソフトウェアのトレードオフを必要最小限で完了させる事に貢献する。また、処理形態に応じて、内部のデータパスを外部へ貸し出しする事も容易にする為、ハードウェアのリソースシェアリング化が可能となり、複数の本発明にかかるモジュール（ＤＡＰ／ＤＮＡモジュール）の間で並列処理化を極め、コンパクトなハードウェアで実現する事が可能となる。

【０１０２】

なお、上記に示したデータ処理装置および命令セットなどは、本発明の一例に過ぎず、たとえば、データ処理装置においては、コードＲＡＭあるいはデータＲＡＭなどを外部のＲＡＭあるいはＲＯＭとしたり、これらに加えて外部のＤＲＡＭあるいはＳＲＡＭなどとのインタフェースを設けることも可能である。さらに、外部の他のデバイスと接続するための入出力インタフェースなど、システムＬＳＩなどのデータ処理装置として公知の機能を備えたデータ処理装置も本発明に含まれる。したがって、本発明は以下の請求の範囲の記載により理解および把握され、それらの請求の範囲に含まれる変形例は全て本発明の範囲に含まれる。

【０１０３】

また、本発明の命令セットおよびデータ処理装置により提供される新しいプログラミング環境においては、上述した以外にも特殊な命令を設けることが可能である。例えば、現在のプログラムとは別に、１つ以上のオブジェクト（プログラム）を同時に起動し、並列処理起動を命令レベルでサポートするＸＦＯＲＫ、オブジェクト（プログラム）間の同期を指定するＸＳＹＮＫ、並列処理間のパイプライン結合を命令するＸＰＩＰＥ、現在のオブジェクトを終了し、次のオブジェクトを起動するＸＳＷＩＴＣＨなどが考えられている。

【０１０４】

以上に説明したように、上記にかかる命令セットおよびそれを用いたプログラミングおよびそれを実行可能なデータ処理装置の技術は、従来の命令セットの構成方法そのものを大幅に変更するものであり、これにより、従来技術では対応の難しかった上述したような問題を上手く解決し、大きな性能向上を図ることができる。

【０１０５】

すなわち、上記において説明した命令セットは、命令セットの構成方法を従来の命令セットの構成方法とは全く異なる視点から見直すことにより、従来技術では解決の極めて困難と思われる多くの問題を、極めて効率良く解決している。実際、従来技術においては、その命令セットの構成法とハードウェアによる命令供給（入手）方法が、極めて画一的で伝統的な先入観により実現されていたため、本質的な意味での解決を遠ざけており、その問題点を全て膨大で複雑なハードウェア構成により解決しようとすることで社会へ貢献すべきテクノロジーとその上に構築される各種の情報処理製品の開発コストを膨大に引き上げる原因となっていた。本発明は、これを本来あるべきアプリケーション要求を優先した命令セットを実現することにより、単に製品性能の効率化に止まらず、その高い開発効率と製品の品質保証を得やすい手段を提供することができる。

【０１０６】

また、上記においては、性能向上に貢献可能なデータパス（データフロー）をテンプレートという資産と、それを使用する命令セットという資産で蓄積できる。さらに、その後に追加されるハードウェア構成情報およびデータ処理を実行するシーケンス情報に基づき随時更新し最適解を求めるようにすることができる。したがって、従来存在したアプリケーション間の資産の共有化とハードウェア資産の共有化、及び高性能化に対する適切なハードウェア投資がより健全な方向へ向かい、ネットワーク化社会を構築する上でのテクノロジー・インフラとしても大きく貢献可能となることが期待できる。

【０１０７】

上記において説明したデータ処理装置は、様々なデータ処理を実行可能なプロセッサあるいはＬＳＩなどとして提供することが可能であり、電子素子の集積回路のみならず、光

素子、さらには電子素子および光素子を集積した光集積回路装置にも適用することができる。特に、本発明の命令セットを備えた制御プログラムおよびデータ処理装置においては、データ処理を柔軟に、そして高速に実行できるので、ネットワーク処理や、画像処理などの高速性およびリアルタイム性能を要求されるデータ処理装置に好適なものである。

【図面の簡単な説明】

【０１０８】

【図１】命令セットの概要を示す図である。

【図２】図１に示す命令セットのＹフィールドをさらに詳しく説明する図である。

【図３】図１に示す命令セットを実際に用いた簡単な例を示す図である。

【図４】図３に示す命令セットによりデータがレジスタに格納される様子を示す図である。

【図５】図１の命令セットを実行可能なデータ処理装置の例を示す図である。

【図６】従来のＣＰＵあるいはＤＳＰで実行可能なサンプルプログラムである。

【図７】本発明のプログラム例である。

【図８】図７に示すプログラムを本発明にかかる命令セットの実行プログラムにコンパイルした例を示す図である。

【図９】異なるプログラム例である。

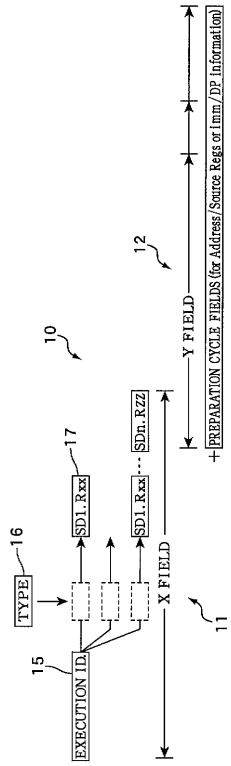
【図１０】図９のプログラムにより構成されたデータフローを示す図である。

【図１１】図１に示す命令セットによりデータ処理を実行可能なデータ処理装置の概略構成を示す図である。

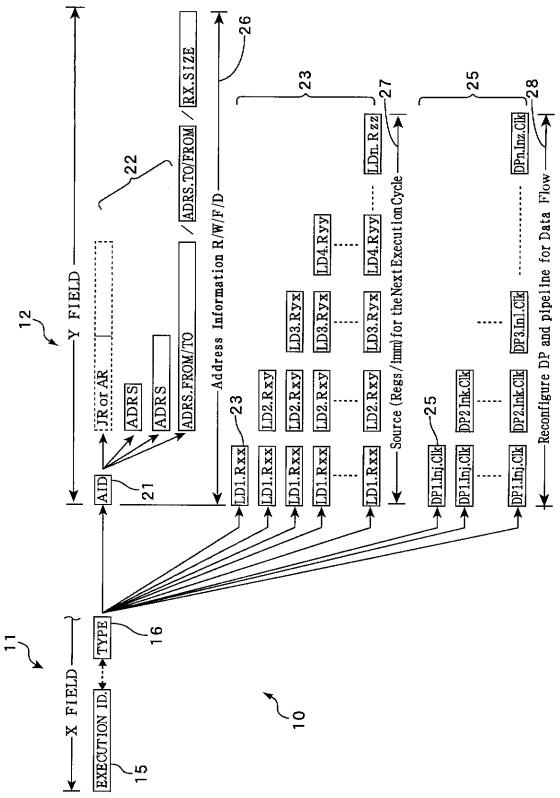
【図１２】テンプレートの組み合わせを変えて異なる専用回路を構成する様子を示す図である。

【図１３】テンプレートの一例を示す図である。

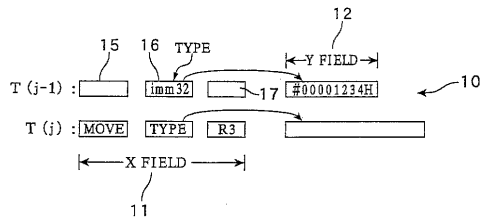
【図１】



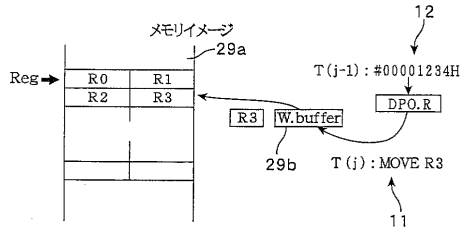
【図２】



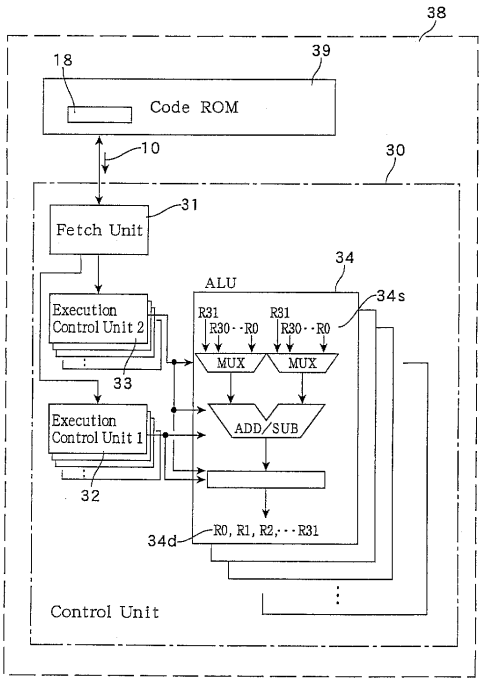
【図３】


12
15
16
TYPE
Y FIELD
T (j-1) :
imm32
17
#00001234H
10
T (j) :
MOVE
TYPE
R3
X FIELD
11

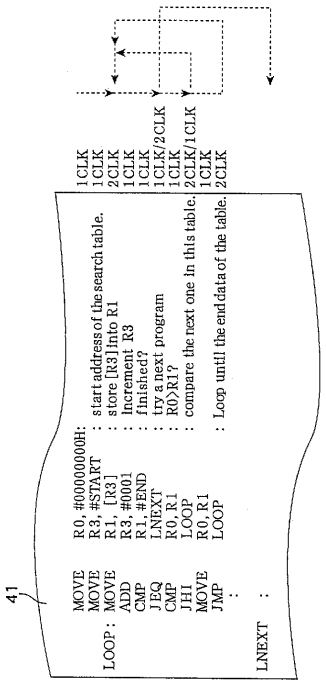
【図４】


メモリイメージ
29a
12
Reg
R0
R1
R2
R3
T(j-1) : #00001234H
DPO.R
R3
W.buffer
29b
T (j) : MOVE R3
11

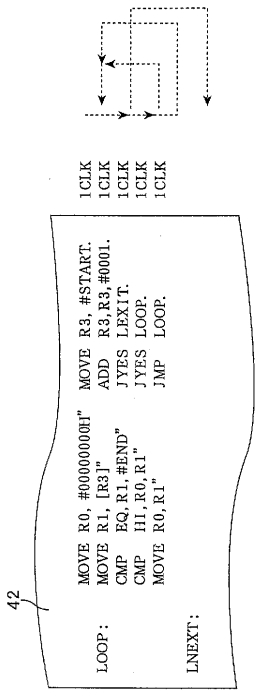
【図５】


38
18
39
Code ROM
10
30
Fetch Unit
31
34
ALU
34s
Execution Control Unit 2
R31
R30･･R0
R31
R30･･R0
MUX
MUX
33
Execution Control Unit 1
ADD/SUB
32
34d
R0, R1, R2,･･･R31
Control Unit

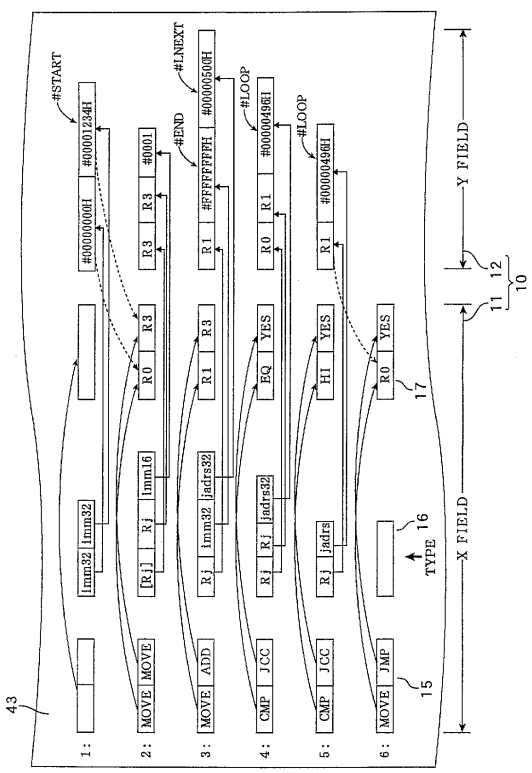
【図６】


1CLK
1CLK
2CLK
1CLK
1CLK/2CLK
1CLK
2CLK/1CLK
1CLK
2CLK
MOVE R0, #00000000H: ; start address of the search table.
MOVE R3, #START ; store [R3] into R1
LOOP: MOVE R1, [R3] ; Increment R3
ADD R3, #0001 ; finished?
CMP R1, #END ; try a next program
JEQ LNEXT ; R0)R1?
CMP R0, R1 ; compare the next one in this table.
JHI LOOP
MOVE R0, R1 ; Loop until the end data of the table.
JMP LOOP
;
LNEXT ;
41

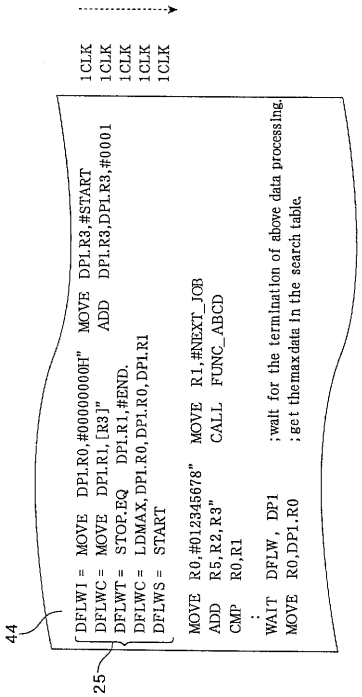
【図７】


1CLK
1CLK
1CLK
1CLK
1CLK
MOVE R0, #00000000H" MOVE R3, #START.
MOVE R1, [R3]" ADD R3,R3,#0001.
CMP EQ,R1,#END" JYES LEXIT.
CMP H1,R0,R1" JYES LOOP.
MOVE R0,R1" JMP LOOP.
LOOP:
LNEXT:
42

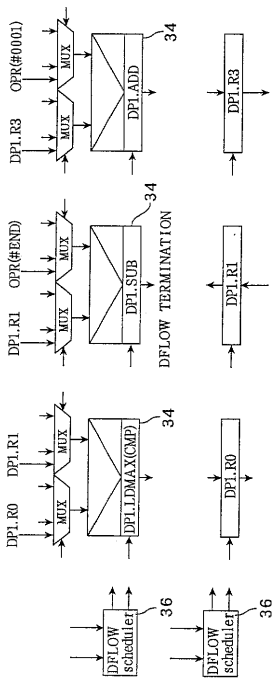
【図８】


#START
#LNEXT
#END
#LOOP
#LOOP
#LOOP
#00000000H
#00001234H
#0001
#FFFFFFFFH
#00000500H
#00000496H
#00000496H
imm32
imm32
imm16
jadrs32
jadrs
jadrs32
jadrs
MOVE
MOVE
ADD
CMP
JCC
CMP
JCC
MOVE
JMP
YES
YES
YES
TYPE
X FIELD
Y FIELD
43
10
11
12
15
16
17

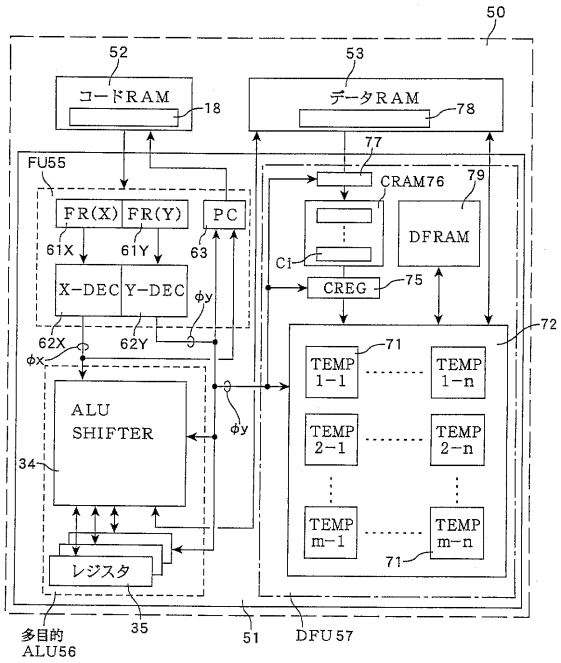
【図９】


1CLK
1CLK
1CLK
1CLK
1CLK
DFLWI = MOVE DP1.R0,#00000000H" MOVE DP1.R3,#START
DFLWC = MOVE DP1.R1,[R3]" ADD DP1.R3,DP1.R3,#0001
DFLWT = STOP.EQ DP1.R1,#END.
DFLWC = LDMAX,DP1.R0,DP1.R0,DP1.R1
DFLWS = START
MOVE R0,#012345678" MOVE R1,#NEXT_JOB
ADD R5,R2,R3"
CMP R0,R1
:
WAIT DFLW, DP1 ;wait for the termination of above data processing.
MOVE R0,DP1.R0 ;get the max data in the search table.
44
25

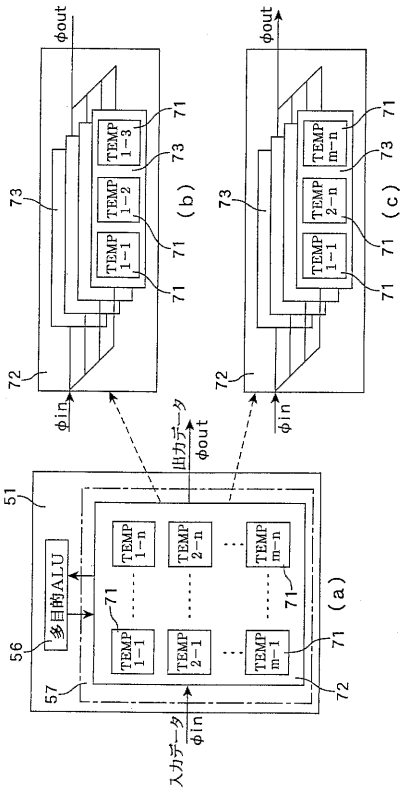
【図１０】


OPR(#0001)
DP1.R3
DP1.ADD
34
DP1.R3
OPR(#END)
DP1.R1
DP1.SUB
34
DFLOW TERMINATION
DP1.R1
DP1.R1
DP1.R0
DP1.LDMAX(CMP)
34
DP1.R0
DFLOW scheduler
36
DFLOW scheduler
36

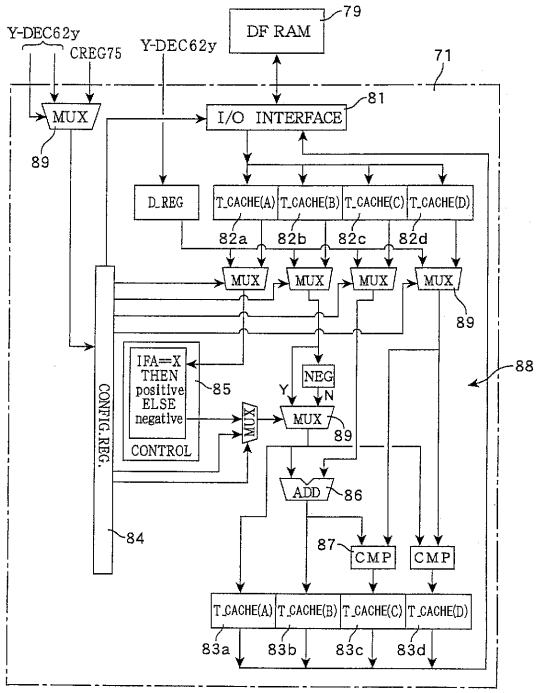
【図１１】


50
52
コードRAM
18
53
データRAM
78
77
FU55
CRAM76
79
FR(X)
FR(Y)
PC
DFRAM
61X
61Y
63
Ci
X-DEC
Y-DEC
CREG
75
φy
72
62X
62Y
φx
71
TEMP 1-1
TEMP 1-n
ALU
SHIFTER
TEMP 2-1
TEMP 2-n
34
TEMP m-1
TEMP m-n
71
φy
レジスタ
多目的 ALU56
35
51
DFU57

【図１２】


φout
φin
TEMP 1−3
TEMP 1−2
TEMP 1−1
TEMP m−n
TEMP 2−n
TEMP 1−1
(b)
(c)
(a)
出力データ
入力データ
多目的ALU
TEMP 1−n
TEMP 2−n
TEMP m−n
TEMP 1−1
TEMP 2−1
TEMP m−1
51
56
57
71
72
73

【図１３】


Y-DEC62y
CREG75
Y-DEC62y
DF RAM
79
71
MUX
89
I/O INTERFACE
81
D_REG
T_CACHE(A)
T_CACHE(B)
T_CACHE(C)
T_CACHE(D)
82a
82b
82c
82d
MUX
MUX
MUX
MUX
89
CONFIG.REG.
IFA=X
THEN
positive
ELSE
negative
CONTROL
85
NEG
Y
N
MUX
89
MUX
ADD
86
84
87
CMP
CMP
T_CACHE(A)
T_CACHE(B)
T_CACHE(C)
T_CACHE(D)
83a
83b
83c
83d
88

フロントページの続き

(56)参考文献 特開平１１－０８５５０７（ＪＰ，Ａ）
特開平０５－２４２０５０（ＪＰ，Ａ）
米国特許第０５５３５４０６（ＵＳ，Ａ）
米国特許第０５８０５８７５（ＵＳ，Ａ）
Scott Hauck, "Configuration prefetch for single context reconfigurable coprocessors", International Symposium on Field Programmable Gate Arrays Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, 米国, ACM, １９９８年, pp.65 - 74 , ISBN:0-89791-978-5
B.Kastrup et al., ConCISe: a compiler-driven CPLD-based instruction set accelerator, Field-Programmable Custom Computing Machines, 1999. FCCM '99. Proceedings. Seventh Annual IEEE Symposium on, 米国, IEEE, １９９９年 ４月２３日, pp.92-101, ISBN: 0-7695-0375-6

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ ９／３０－９／３８