(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4384828号
(P4384828)

(45)発行日　平成21年12月16日(2009.12.16)　(24)登録日　平成21年10月2日(2009.10.2)

(51)Int.Cl.		F I		
G06F 3/153	(2006.01)	G06F	3/153	330B
G06F 9/38	(2006.01)	G06F	9/38	370C
G06F 13/28	(2006.01)	G06F	13/28	310E
G06T 1/20	(2006.01)	G06F	13/28	310Y
		G06T	1/20	Z

請求項の数 18　(全 19 頁)

(21)出願番号　特願2001-357206(P2001-357206)
(22)出願日　平成13年11月22日(2001.11.22)
(65)公開番号　特開2003-167726(P2003-167726A)
(43)公開日　平成15年6月13日(2003.6.13)
審査請求日　平成16年11月10日(2004.11.10)

(73)特許権者　502457803
ユニヴァーシティ　オブ　ワシントン
アメリカ合衆国　９８１０５－４６０８
ワシントン州　シアトル　１１　アベニュー　ノースイースト　４３１１　スイート　５００
(74)代理人　100064746
弁理士　深見　久郎
(74)代理人　100085132
弁理士　森田　俊雄
(74)代理人　100083703
弁理士　仲村　義平
(74)代理人　100091409
弁理士　伊藤　英彦

最終頁に続く

(54)【発明の名称】コプロセッサ装置およびデータ転送を容易にするための方法

(57)【特許請求の範囲】
【請求項１】
メディアプロセッサによってアクセスするためにデータをロードおよびアンロードするデータ転送を扱うためのテンプレートデータ転送コプロセッサ装置であって、
対応のデータ転送動作を規定する一組のパラメータであって、複数の可能性のあるテンプレートタイプの中からテンプレートタイプを特定するテンプレートタイプパラメータを含む１組のパラメータを、各々が含む１つ以上のデータ転送テンプレートを同時にストアするテンプレートバッファと、
前記テンプレートバッファにストアされた複数のデータ転送テンプレートの中からテンプレートを読出し、前記読出されたテンプレートのテンプレートタイプに従ってデータ転送を制御するテンプレートインタプリタと、
各々が別々のバスに関連付けられた複数のアドレス生成ユニットとを備え、前記アドレス生成ユニットの異なる組は異なる１つのテンプレートタイプに対して用いられ、前記アドレス生成ユニットの各々は、前記読出されたテンプレートに対してアクセスするための１組のアドレスを前記テンプレートインタプリタから受け取った情報に基づいて生成し、コプロセッサ装置はさらに、
ソースからデスティネーションに転送されるデータを受けるデータ転送バッファとを備えることを特徴とするコプロセッサ装置。
【請求項２】
前記テンプレートバッファにストアされる前記１つ以上のデータ転送テンプレートの所

与の１つに対するパラメータの組は、テンプレートパラメータと、ランタイムに計算されるブロックアドレス情報とを含み、前記テンプレートインタプリタは、ランタイムの間に動的なデータフロー情報を導出することを特徴とする請求項１に記載のコプロセッサ装置。

【請求項３】

前記１つ以上のデータ転送テンプレートの前記少なくとも１つのタイプは、ブロックデータ転送を容易にし、ブロックデータ転送のブロックサイズが導出されブロックデータ転送に対するソースおよびデスティネーションアドレスが導出されるパラメータを含むことを特徴とする請求項１に記載のコプロセッサ装置。

【請求項４】

前記１つ以上のデータ転送テンプレートの前記少なくとも１つのタイプは、ソースブロックパラメータおよびデスティネーションブロックパラメータと、かつブロック転送が１次元データブロックに対するものであるか、または２次元データブロックに対するものであるかの表示とを含むことを特徴とする請求項３に記載のコプロセッサ装置。

【請求項５】

ブロックデータ転送を容易にする前記１つ以上のデータ転送テンプレートの前記少なくとも１つのタイプは、ソースメモリ空間内で隣接するブロックがオーバーラップするか否かおよび境界データを変更する態様を指定するパラメータをさらに含むことを特徴とする請求項４に記載のコプロセッサ装置。

【請求項６】

前記１つ以上のデータ転送テンプレートの前記少なくとも１つのタイプは、プログラム誘導ブロックデータ転送を容易にし、ランダムに配置される任意のサイズのデータブロックのシーケンスの転送を容易にするためのソースパラメータとデスティネーションパラメータとを含むことを特徴とする請求項１に記載のコプロセッサ装置。

【請求項７】

前記１つ以上のデータ転送テンプレートの前記少なくとも１つのタイプは、間接データ転送を容易にし、インデックスアドレスパラメータ、ソースアドレスパラメータおよびデスティネーションアドレスパラメータを含むことを特徴とする請求項１に記載のコプロセッサ装置。

【請求項８】

前記テンプレートインタプリタは、データ転送するためにキューに基づくテンプレートを読出し、前記キューに基づくテンプレートは物理キューを指定するためのパラメータと、仮想キューを指定するための複数のパラメータとを含み、物理キューと仮想キューとの間を定められた方向で前記データが転送されることを特徴とする請求項１に記載のコプロセッサ装置。

【請求項９】

外部メモリとオンチップメモリを有するメディアプロセッサとの組合せにおいて、テンプレートデータ転送コプロセッサ装置は、外部メモリとオンチップメモリとの間のデータ転送を容易にすることを特徴とする請求項１に記載のコプロセッサ装置。

【請求項１０】

メディアプロセッサのオンチップメモリとオフチップメモリとの間のデータ転送を容易にするための転送制御方法であって、

テンプレートバッファから複数のデータ転送テンプレートの少なくとも１つを読み出すステップを含み、前記複数のデータ転送テンプレートの各々は、オンチップメモリとオフチップメモリとの間のデータ転送を容易にする複数のパラメータを含み、前記複数のパラメータは、テンプレートタイプの範囲の中からテンプレートタイプを特定するテンプレートタイプパラメータを備え、前記方法はさらに

ロードされたテンプレートのテンプレートタイプにしたがって、各々が別々のバスに関連付けられた複数のアドレス生成ユニットから、一組のアドレス生成ユニットを使用して

、前記ロードされたテンプレートに対してアクセスされる一組のアドレスを生成するステップと、

前記ロードされたテンプレートのテンプレートタイプにしたがって、コプロセッサによって、オンチップメモリとオフチップメモリとの間で生成された一組のアドレスを用いてデータ転送するステップと、前記メディアプロセッサに複数の画像データブロックを提供するステップと、

前記複数の画像データブロックに対して前記メディアプロセッサで前記画像処理アルゴリズムを実行するステップとを備えることを特徴とする転送制御方法。

【請求項１１】

前記複数のデータ転送テンプレートの前記少なくとも１つはブロックデータ転送を容易にし、ブロックデータ転送のブロックサイズを導出できブロックデータ転送に対するソースおよびデスティネーションアドレスを導出できるパラメータを含むことを特徴とする請求項１０に記載の転送制御方法。

【請求項１２】

前記複数のデータ転送テンプレートの前記少なくとも１つは、ソースブロックパラメータとデスティネーションブロックパラメータとを含み、かつブロック転送が１次元データブロックに対するものであるか、または２次元データブロックに対するものであるかの表示を含むことを特徴とする請求項１０に記載の転送制御方法。

【請求項１３】

前記複数のデータ転送テンプレートのうちの前記少なくとも１つは、ソースメモリ空間内で隣接するブロックがオーバーラップするか否かおよび境界データ
を変更する態様を指定するパラメータをさらに含むことを特徴とする請求項１２に記載の転送制御方法。

【請求項１４】

前記１つ以上のデータ転送テンプレートの前記少なくとも１つは、プログラム誘導ブロックデータ転送を容易にし、ランダムに配置される任意のサイズのデータブロックのシーケンスを転送することを容易にするためのソースパラメータとデスティネーションパラメータとを含むことを特徴とする請求項１０に記載の転送制御方法。

【請求項１５】

前記１つ以上のデータ転送パラメータの前記少なくとも１つは、間接データ転送を容易にし、インデックスアドレスパラメータ、ソースアドレスパラメータ、およびデスティネーションアドレスパラメータを含むことを特徴とする請求項１０に記載の転送制御方法。

【請求項１６】

前記複数のデータ転送テンプレートの１つは、物理キューを指定するためのパラメータと仮想キューを指定するための複数のパラメータとを備えるキューに基づくテンプレートであって、前記データを転送するステップは、物理キューと仮想キューとの間でデータを転送するステップを含むことを特徴とする請求項１０に記載の転送制御方法。

【請求項１７】

前記識別されたパラメータの少なくとも１つは、コプロセッサによってランタイムに規定されることを特徴とする請求項１０に記載の転送制御方法。

【請求項１８】

前記転送を容易にするステップは、サブワード区分を有するオペランドとしてパックした前記複数の画像データブロックをメディアプロセッサに提供するステップを含み、当該サブワードは、前記複数のデータブロックの画像データブロックの１つのデータ項目に対応することを特徴とする請求項１０に記載の転送制御方法。

【発明の詳細な説明】

【０００１】

【発明の背景】

この発明は、オンチップ並列性を実現するプロセッサのためのデータフロー管理に関し、より特定的には、データ転送を管理し、オンチップ並列性を示す効率のよい性能のプロセ

ッサを可能にするためのコプロセッサに関する。
【０００２】
ここで用いられる「メディアプロセッサ」は、ビデオまたは画像データを処理するためのオンチップ並列性を示すプロセッサを指す。効率のよいデータフロー管理は、メディアプロセッサにおいて高性能を達成するために必須である。しかしながら、現在のメディアプロセッサはアプリケーションプログラマに対して低レベルのデータ転送インターフェイスしか提供せず、これはデータフロープログラミングを困難にするだけでなく、ソースコードを長くし維持を難しくする。
【０００３】
現在のメディアプロセッサは典型的には、広いデータ経路でサブＧＨｚクロック周波数で動作する。これは、メモリに対する高いデータアクセス率を要求する。しかしながら、オフチップメモリへのアクセスは長いレイテンシに関わり、結果として全体的なメモリ帯域幅を制限する。したがって、これらの頻繁に用いられるデータをキャッシュし、かつ外部メモリアクセスペナルティを減じるために高速オンチップメモリが用いられる。さらに、実効アクセスレイテンシを減じるための、すなわちＣＰＵが実際にデータを用いる前にデータをオンチップメモリ上で利用可能にするためのさまざまなデータプリフェッチ技術が開発されている。
【０００４】
負担の大きい、大量の生データに関わる画像およびビデオアプリケーションがメディアプロセッサの主なターゲットである。典型的な画像およびビデオ処理アルゴリズムは、規則的なデータアクセスパターンを有する。こうして、プログラムはまもなく使われるデータのブロックを前もってフェッチできる。しかしながら、そのような負荷を実現するために従来のメディアプロセッサによって要求されるプログラミングは非常に制限されている。ほとんどのメディアプロセッサは、プログラマが処理タスクを区分して、ブロック転送を行なうために区分の各々に対して用いられるデータブロックのサイズおよびアドレスを決定することを要求する。画像とビデオフレームとは異なったサイズを有し得るので、プログラマは不規則な区分および付加的な制御フローを用いてコードを一般化しなければならず、プログラムが長くなる。さらに、いくつかのアルゴリズムにおいては、パッディングのようなさらなるタスクが行なわれることが必要となる。これらは付加されたプログラム命令によって行なわれるので、データ処理タスクの効率性を減じる。ブロックに基づくデータフロープログラミングが用いられる多くのアルゴリズムにおいて、バルクデータに加えて境界のピクセルを明確にフェッチすることが必要である。プログラムにおけるそのような不規則なデータフローを取扱うことは、プログラマの全体的な生産性を減じ、かつソースコードをアップグレードおよび維持することをも困難にする。さらに、計算とデータフローとの間の同時実行の程度が減じられる。したがって、ブロックデータ転送を行なう、より効率的で汎用性の高い態様に対する必要性が存在する。
【０００５】
広いデータ経路を備えたメディアプロセッサに対する特別な懸念は、多倍精度（multiple small precision）オペランドを異なった記憶場所からワイドワード（wide word）へのパッキングのオーバーヘッドである。これは従来、メディアプロセッサにおいて命令を明示的に実行することにより行なわれるので、貴重なＣＰＵサイクルが消費され、かつ全体的な性能が減じられる。したがって、多数のオペランドをパッキングするための、より効率的な態様に対する必要性が存在する。
【０００６】
【発明の概要】
この発明によると、メディアプロセッサからのオフロードブロックデータ転送動作に対してテンプレートデータ転送コプロセッサが実現される。テンプレートデータ転送コプロセッサは、さまざまなタイプのブロックデータ転送動作を指定するための汎用性の高いプロトコルを提供する。
【０００７】

この発明の一局面によると、さまざまなタイプのデータ転送テンプレートが導入される：ユニブロック（uniblock）テンプレート、プログラム誘導（program-guided）テンプレート、間接（indirect）テンプレート、およびキューに基づく（queue-based）テンプレートである。これらのテンプレートは、メディアプロセッサにおけるさまざまなタイプのデータ転送フローを容易にする。この発明の利点によると、ブロック転送ごとに低レベルデータ転送パラメータを計算しかつ設定するのではなく、パラメータ化されたテンプレートがプログラマに特定のアルゴリズムに対して必要なデータフローを容易に設計できる効率的で汎用性の高い機構を提供する。たとえば、２Ｄ畳み込みおよびアフィンワーピング（affine warping）においては、データフロープログラミングに関するソースコードラインの数が実質的に減じられる（たとえば、テンプレートなしで要求されるデータ転送を達成する場合に要求される数の約６分の１となる）。

【０００８】

この発明の別の局面によると、ユニブロックおよびプログラム誘導テンプレートは、メディアプロセッサのオンチップメモリとオフチップメモリとの間でブロックに基づくデータ転送において用いられる。そのようなテンプレートの利点によると、オンチップメモリにおける入力および出力データブロックはダブルバッファリングされることが可能であり、それによりメディアプロセッサ機能ユニットは高速計算のためにオンチップメモリにおけるデータにアクセスするだけでよい一方で、遅い外部メモリとの間のデータ転送は機能ユニットから隠される。ブロックごとに、アドレスおよびサイズはテンプレートデータ転送コプロセッサによってテンプレートから計算される。したがって、データフロープログラミングは簡略化され、ＣＰＵの計算負荷は減じられる。

【０００９】

ユニブロックテンプレートは、オーバーラップすることができるソースメモリ空間からのブロックを指定する。したがって、境界のブロックはパッディングされるか、またはラップアラウンドされて、境界に沿った必要なオーバーラップ領域を準備する。

【００１０】

パッディングおよびラッピングは実際のデータ転送の間に行なわれる。したがって、メディアプロセッサ計算はさらに減じられ、かつこれらの不規則なデータコピー動作はテンプレートデータ転送コプロセッサによって自動的に扱われるという事実により、プログラマの負担は軽くなる。

【００１１】

この発明の別の局面によると、間接データフローテンプレートは、プロセッサの計算エンジンに顕著な負荷を生成することなく、ランダムな場所からデータをアセンブルするための効率的な方法を可能にする。

【００１２】

この発明の別の局面によると、キューに基づくデータフローテンプレートは、仮想キューと物理キューとの間のデータ転送を容易にする。仮想キューは、循環的にアクセスされるメモリ空間であってもよく、任意のサイズを有する。物理キューはデータをバッファし、かつ計算エンジンに対してオペランドストリームを提供する。テンプレートデータ転送コプロセッサは、２Ｄメモリ空間からオペランドバッファへのデータをストリーム化するか、またはデスティネーションオペランドバッファにストアされるメディアプロセッサ結果を２Ｄメモリ空間にディスパッチする。そのようなテンプレートの利点によると、キューに基づくデータフローは多くのロード／ストア命令をなくし、レジスタファイルサイズに対する圧力を緩和する。さらに、長いレイテンシロード／ストア命令がなくなってコードスケジューリングはより簡略化され、高性能のために要求されるループアンローリングの量は減じられ、よりコンパクトなコードをもたらす。

【００１３】

この発明の別の局面によると、テンプレートデータ転送コプロセッサは２Ｄアドレス生成ユニット（ＡＧＵ）を含み、これは割込なしでアドレスのシーケンスが生成されることを可能にする組込み論理ユニットを有する。これらのアドレスが２Ｄブロックに対応し、こ

れは最も小さなデータ転送ユニットとしての役割を果たす。この態様で、テンプレートデータ転送コプロセッサ内のセントラルコントローラは、アドレス生成事象の各々に関与する必要がなく、よってたとえば、次の転送ブロックの位置の突き止めや、データフローのアービトレーションなどの他のテンプレート計算のためのさらなる時間をもたらす。

【００１４】

この発明の別の局面によると、テンプレートデータ転送コプロセッサはテンプレートインタプリタを含み、これはコプロセッサのメインコントローラとしての役割を果たす。一実施例においては、インタプリタはテンプレートをセットアップし、テンプレートごとにブロック情報を計算するための事象駆動制御機構を用いる。インタプリタの利点によると、テンプレートのタイプごとの計算はモジュラ化される。したがって、新しいテンプレートタイプを簡単に追加することができる。

【００１５】

この発明のこれらおよび他の局面と利点とは、添付の図面と併せて以下の詳細な説明を参照することにより、よりよく理解されるであろう。

【００１６】

【特定の実施例の説明】

概要

図１を参照すると、画像またはビデオデータを処理するためのホストシステム１０は、１つ以上のバス構造２２によって相互接続される、メディアプロセッサ１２、テンプレートデータコプロセッサ１４、メインメモリ１６、不揮発性メモリ１８、およびユーザインターフェイス２０を含む。ユーザインターフェイス２０はディスプレイ装置２４、キーボード２６およびポイント／クリック装置２８を含む。

【００１７】

図２を参照すると、一実施例においてメディアプロセッサ１２は、日本国東京の株式会社日立製作所およびカリフォルニア州キャンベルのイクエータ・テクノロジー（Equator Technologies）によって製造されるメディア加速プロセッサ（Media Accelerated Processor）１０００（ＭＡＰ１０００）によって形成される。ＭＡＰ１０００は、直接メモリアクセス（ＤＭＡ）コントローラ２９、オンチップメモリ（データキャッシュ３０および命令キャッシュ３２）およびクラスタ３４と呼ばれる並列実行ユニットを含む。クラスタ３４の各々は、整数演算および論理ユニット（ＩＡＬＵ）３６と、整数浮動小数点グラフィック演算および論理ユニット（ＩＦＧＡＬＵ）３８とを含む。また、クラスタ３４の各々はいくつかの汎用レジスタ（たとえば３２ビットレジスタ）、いくつかの１ビットプレディケートレジスタおよび多数の特別レジスタ（たとえば、１２８ビットレジスタ）を含む。

【００１８】

他のさまざまなメディアプロセッサ実施例もまた実現される。特に、ここで用いられる「メディアプロセッサ」は、ビデオまたは画像データを処理するためのオンチップ並列性を示すプロセッサを指す。マイクロプロセッサおよびデジタル信号プロセッサは、命令レベル並列性と呼ばれる技術によってオンチップ並列性を用いる。命令レベル並列性は、多数の動作が単一のクロックサイクルで開始されるものである。命令レベル並列性への２つの方策は：超長命令語（ＶＬＩＷ）アーキテクチャおよびスーパースカラアーキテクチャである。ＶＬＩＷアーキテクチャにおいては、プロセッサは多数の独立処理ユニットを含む。長い命令の各々は機能ユニットの各々に対するオペレーションコードを含む。すべての機能ユニットは、実質的に同じ時点でそれらのオペレーションコードを受取る。機能ユニットはそれらの割当てられたタスクを同時に実行する。スーパースカラアーキテクチャは特別なオンチップハードウェアを用いて、命令ストリームを調べ、並列性を最大化するために同時に実行し得る独立した動作を見出す。

【００１９】

命令レベル並列性は、サブワード並列性を用いていくつかのシステムにおいてさらに展開されるが、ここでは実行ユニットは多数のより小さなユニットに区分される。たとえば、

主に６４ビット論理演算装置（ＡＬＵ）によって実現されるプロセスは、ＡＬＵを論理的に４つの小さな１６ビットＡＬＵに分割する。特定的には、ＡＬＵへのデータ入力は４つの小さなサブワードの連結である。ＡＬＵ出力は４つのサブワードでの結果の連結である。そのようなサブワード並列性は、「単一命令多重データ」（ＳＩＭＤ）命令と呼ばれるものを与えることによりアーキテクチャに組入れられる。メディアプロセッサ１２のＳＩＭＤ実現化の例は：サン・マイクロシステムズ（Sun Microsystems）のビジュアル命令セット、インテル（Intel）のマルチメディア拡張機構、ヒューレット・パッカード（Hewlett-Packard）のmultimedia acceleration extensions-s、デジタル・イクイップメント・コーポレーション（Digital Equipment Corporation）のマルチメディア拡張機構、シリコン・グラフィックス（Silicon Graphics, Inc.）のデジタルメディア拡張機構である。これらの拡張機構における命令は、データワード（たとえば３２ビットまたは６４ビット）を１組の多数のサブワード（８、１６または３２）として処理する。サブワードの各々で区分された動作を実行することができ、最小限の付加的なハードウェアで２倍、４倍または８倍の性能向上が得られる。

【００２０】

テンプレートデータ転送コプロセッサ（ＴＤＴＰ）１４は、メディアプロセッサの処理ユニット（たとえばクラスタ３４）、メディアプロセッサのオンチップメモリキャッシュ３０、３２および外部メモリ（たとえばシステムキャッシュ１６および不揮発性メモリ１８）の間のさまざまなデータフローパターンを扱う役割を果たす。テンプレートを用いることにより、最小限のプログラミングが実現されてこのデータフロー制御を達成する。

【００２１】

テンプレート

テンプレートは、（ｉ）パラメータ化されたデータ構造、または（ii）規定されたパラメータの組を備えたパラメータ化されたデータ構造、のいずれかである。メディアプロセッサにおいて異なったタイプのデータフローを扱うためにさまざまなテンプレートが用いられる。ここでは、ユニブロックテンプレート、プログラム誘導テンプレート、間接テンプレートおよびキューに基づくテンプレートを含む４つのテンプレートを説明する。これらのテンプレートタイプの各々は、同様のデータフローの群に対して用いられる。ユニブロックテンプレートは、２つの記憶場所領域の間のブロックごとのデータ転送を説明する。すべてのデータブロックパラメータ、たとえばアドレスおよびサイズは、テンプレートから導出される。プログラム誘導テンプレートもまたブロックに基づくデータ転送を規定するが、データブロックパラメータはプログラムによって明示的に与えられる。ユニブロックテンプレートおよびプログラム誘導テンプレートは主に画像／ビデオ処理に対して望ましいフローパターンであるダブルバッファリングデータフローを実現するために用いられる。間接テンプレートにより、離散的データ要素を（隣接するデータをインデクシングすることにより）連結するメモリ空間へマッピングすることが可能になる。間接テンプレートは効率的にメディアプロセッサの作業負荷を減じる。キューに基づくテンプレートは、メディアプロセッサに対するメモリ空間とバッファされたオペランドの小さな物理ストリームキューとの間のデータ転送を指定する。一実施例においてはテンプレートの各々における第１の項目はテンプレートのタイプを識別するためのコードである。各々のテンプレートタイプのより詳細な説明を以下に述べる。

【００２２】

ユニブロックテンプレート

ほとんどの画像およびビデオアルゴリズムは、単一のデータフレームまたは多数のデータフレームで計算を実行する。多くのアルゴリズムは連続的なデータが独立して計算できるデータレベルの並列性さえも実現する。たとえば、処理は区分され、各々の区分が制限されたオンチップメモリ空間に整合する画像データのブロックに動作し得る。そのような処理に対するデータ転送を容易にするために、ユニブロックテンプレートが導入される。ユニブロックテンプレートは、予め定められた転送ブロック次元およびサイズでオンチップメモリ空間とオフチップメモリ空間との間のデータ転送を説明するために用いられる。

【００２３】
図３を参照すると、ユニブロックテンプレートは、ソースブロックパラメータ３８とデスティネーションブロックパラメータ４０とを用いてソースブロックとデスティネーションブロックとを規定する。ソースブロックパラメータ３８は、ベースアドレス、幅、高さ、ピッチおよびソースアクセスモードを含む。同様に、デスティネーションブロックパラメータ４０は、ベースアドレス、幅、高さ、ピッチおよびデスティネーションアクセスモードを含む。次元フラグ４２（すなわち、１Ｄ／２Ｄフラグと標識付けされる）は、データ転送の次元を決定する。典型的には、２次元（２Ｄ）ブロックはブロック幅４４およびブロック高さ４６パラメータで規定される。フラグ４２が１次元（１Ｄ）を示す場合、ブロック幅４４のみが有効である。２次元データ転送に関しては、オーバーラップモードパラメータ４８が用いられ、これはソースメモリ空間における隣接するブロックがオーバーラップするか否かを指定する。オーバーラップが設定される場合、パッディングモードパラメータ５０またはラッピングモードパラメータ５２のいずれかが有効であり、オーバーラップする境界で必要であるデータを処理するためにパッディングまたはラッピングのいずれかが用いられることを示す。さらに、パッディングモードおよびラッピングモードの各々に対して、水平オーバーラップパラメータ５４値および垂直オーバーラップパラメータ５６値が設定される。オーバーラップする部分で、ソースブロックおよび隣接するデータのウィンドウからピクセル値が導出される。

【００２４】
そのようなパッディングおよびラッピング指定は、あるアルゴリズムをより効率的に実現するために望ましい。たとえば、２Ｄ畳み込みを含むアルゴリズムにおいては、境界上のデータブロックがパッディングされる必要がある。ウェーブレット変換を含むアルゴリズムにおいては、データブロックはラップアラウンドされる必要がある。ブロックに基づくデータフロープログラミングが用いられる多くのアルゴリズムにおいて、バルクデータに加えて境界のピクセルを明示的にフェッチすることが必要である。プログラム内でそのような不規則なデータフローを扱うことは、プログラマの全体的な生産性を減じ、かつソースコードをアップグレードし維持することを難しくする傾向がある。さらに、計算とデータフローとの間の同時性の度合が減じられる。ユニブロックテンプレートは、そのような機能を指定するための効率的なプログラミングインターフェイスを提供する。

【００２５】
図４を参照すると、パッディングが実施されるべき９つのブロック５８のソースデータが示される。ピクセルを計算するのに隣接するピクセルが用いられる場合、示される対称的な隣接ウィンドウが境界で用いられる。図４は、ブロック６４に対する水平オーバーラップ６０および垂直オーバーラップ６２を示す。対称的な隣接ウィンドウに対しては、パッディングは同様である。

【００２６】
図５を参照すると、ラッピングモードが用いられる場合、オーバーラップは対称的ではない。そうではなく、これは単方向性である。パッディングモードは、ゼロパッディングと境界拡張との間を選択可能である一方、ラッピングモードはさらに垂直（上または下）および水平（左または右）ラッピングを指定できる。図５は、ブロック６４′に対する水平オーバーラップ６０′および垂直オーバーラップ６２′を示す。

【００２７】
ソースおよびデスティネーションアクセスモードを用いることにより、付加的な柔軟性が得られる。アクセスモードは、アクセスされたデータがオンチップに向けられているか、またはオフチップメモリに対して向けられているかを判断する。これはオンチップメモリにおけるオフチップデータのダブルバッファリングにおいて主要な問題である。また、アクセスモードはソースまたはデスティネーションメモリ空間における次のデータブロックがどのようにアドレスされるかを決定する。図６（Ａ）を参照すると、アクセスモードは次のブロックが２Ｄメモリ空間において行ごとにまたは列ごとにアクセスされるかを決定する。図６（Ｂ）を参照すると、メモリ空間境界に到達すると、アクセスモードは次に逆

方向のブロックがアクセスされるべきかまたはメモリ空間は循環的にアクセスされるべきかを判断する。

【００２８】

ユニブロックテンプレート３６は、ソースブロックとデスティネーションブロックとが同じサイズであることを必要としない。たとえば、テンプレート３６が入力データフローを規定する場合、ソースメモリ空間はソース画像に対応する一方、デスティネーションメモリ空間は入力データをダブルバッファするオンチップメモリ領域に対応する。データ転送は、大きい方のソースメモリ空間におけるすべてのデータが転送されたときに終了する。小さい方のオンチップメモリ空間は逆方向のまたは循環的なアクセスモードのいずれかによって、図６（Ｂ）に示されるように再利用される。

【００２９】

テンプレートデータ転送コプロセッサ１４は、ブロック転送ごとにデータブロックアドレスを計算し、ソースまたはデスティネーションメモリ空間に整合するようブロックサイズを調整する。したがって、画像境界におけるブロックは異なった幅および／または高さ値を、テンプレートに指定されるものからは異なって有し得る。ランタイムブロック情報、たとえばアドレス、幅、および高さは、プログラムにパスされる。こうして、プログラムは単にデータフローテンプレートを初期化し、ブロック転送を同期させ、ブロックをオンチップに処理するだけでよい。

【００３０】

プログラム誘導テンプレート

いくつかのアルゴリズムは不規則なブロックアクセスを要求し、すなわちデータブロックの各々のアドレスおよびサイズがプログラムから計算されなければならない。図７に示すプログラム誘導テンプレート６６は、そのようなプログラムによって導かれるデータ転送を容易にするために用いられる。プログラム誘導テンプレート６６は記述アドレスパラメータ６８、ソースフィールド６９およびデスティネーションフィールド７３を含む。ソースフィールド６９は、ソースアドレスパラメータ７０およびソースピッチパラメータ７２を含む。デスティネーションフィールドは、デスティネーションアドレスパラメータ７４およびデスティネーションピッチパラメータ７６を含む。

【００３１】

記述アドレスパラメータ６８は、転送されるべきブロックの各々のソースアドレスオフセット、デスティネーションアドレスオフセット、およびサイズ情報を含むブロック記述テーブル７８をポイントするアドレスをストアする。ソースフィールド６９およびデスティネーションフィールド７３は、それぞれソースブロックおよびデスティネーションブロックを開始するためのベースアドレスおよびピッチ値をストアする。異なったベースアドレスを用いることにより、同じブロック記述テーブルは異なったプログラムによって再利用されることができる。

【００３２】

ブロックを転送する場合、テンプレートデータ転送コプロセッサ１４はブロック記述テーブル７８からブロック情報をフェッチし、ソースおよびデスティネーションブロックアドレスを計算し、データ転送を開始する。ブロック記述テーブルにおいて規定されるブロックは、記述テーブルが到達されるまでシーケンシャルに転送される。

【００３３】

プログラム誘導テンプレートはまた、外部メモリ１６、１８とオンチップメモリ３０、３２との間での計算の実行と同時のデータ転送のために用いられ、こうしてプロセッサ計算サイクルからメモリレイテンシサイクルを隠す。特に、プログラム誘導データフローは、ランダムに配置される任意の大きさのデータブロックのシーケンスを転送することを可能にする。したがって、これはプログラムがデータフローを規定することに、より柔軟性を与える。関連するオーバーヘッドは、テンプレートデータ転送コプロセッサ１４が、ブロック転送ごとにブロック記述にアクセスすることである。

【００３４】

間接データフローテンプレート
図８を参照すると、間接データフローテンプレート８０によって容易になるデータ転送は、３つのメモリ空間、すなわちインデックスデータ領域８２、ソースデータ領域８４、およびデスティネーションデータ領域８６に関わる。インデックスデータはソースデータアドレスを計算するために用いられる。したがって、ソースデータ領域８４へのアクセスは極めてランダムになり得る。インデックスデータ領域８２およびデスティネーションデータ領域８６へのアクセスは、シーケンシャルであることが期待される。間接データフローテンプレート８０はインデックスフィールド８８（たとえばインデックスアドレス９０およびインデックス幅９２）、ソースデータアドレスパラメータ９４およびデスティネーションデータアドレスパラメータ９６を含む。これはまた、データ幅９８およびデータカウント１００を指定する。インデックス幅パラメータ９２は、さまざまな形式のデータ、たとえば８、１６または３２ビットのデータをインデックスとして用いることを可能にする。テンプレートデータ転送コプロセッサ１４は、インデックスデータ８２をシーケンシャルな順序に参照し、現在アクセスされているインデックスデータ値を対応のソースアドレスパラメータ値９４に加え、結果として生じるアドレスをソースデータ領域８４へのアクセスに用いる。アクセスされたソースデータはデスティネーション領域８６にシーケンシャルに書込まれる。転送されるデータ項目の数および項目の各々のデータ幅は、カウントパラメータ１００および幅パラメータ９８によって決定される。この態様で、ランダム記憶場所からのデータはパックされることができる。間接データフローは、たとえばジオメトリ変換およびグレースケールマッピングなどの、ルックアップテーブルが用いられるアルゴリズムに望ましい。

【００３５】
間接データ転送は、データフローを通してパックまたはアンパックし、かつプロセッサのアドレス計算を緩和するために用いられるので、ＴＤＴＰ１４を用いて達成されるそのような転送速度は、好ましくはメディアプロセッサクラスタ３４によって行なわれるものに匹敵する。しかしながら、デスティネーションデータ要素の移動を終了させるためには３つのメモリアクセス、すなわち１つはインデックス領域８２からの読出、１つはソース領域８４からの読出、および１つはデスティネーション領域８６への書込、が必要となるが、これはオフチップメモリアクセスに関わる場合に遅くなる。このような潜在的なボトルネックを避けるために、間接データフローはいくつかの実施例においては、上のセクションにおいて説明されたブロックに基づくデータフローを通してプリフェッチすることができるオンチップデータを扱うためにのみ限定される。

【００３６】
キューに基づくデータフロー
ストリームキューは、計算エンジンに対して一定のオペランドのフローを効率的に提供し、こうしてレジスタに対する圧力を緩和する。キューはハードウェア、たとえばＦＩＦＯ（先入れ先出し）メモリを用いるか、またはソフトウェア、たとえば規則的なメモリ空間を用いることのいずれかによって実現される。一実施例においてはテンプレートデータ転送コプロセッサは、ハードウェアキューを用いて計算エンジンに対して必要なオペランドをバッファする。別の実施例においては、テンプレートデータ転送コプロセッサによって循環的にアドレスされるメモリ空間として仮想キューが規定される。さらに別の実施例においては、物理キューとその関連の仮想キューとの組合せとして、プログラマに対して可視である論理キューが規定される。テンプレートデータ転送コプロセッサ１４は、仮想キューと物理キューとの間のデータ転送を制御し、それにより論理キューが仮想キューに匹敵する深さを有するようにする。物理キューはデータをバッファするためにだけ用いられるので、これはデータ転送レートの変動を扱える限り、小さくてもよい。

【００３７】
従来は、キューにストアされたデータはシーケンシャルにアクセスされる。しかしながら、ＴＤＴＰ１４は、キューに基づくテンプレートを用いることによりさらなる柔軟性を可能にする。図９を参照すると、キューに基づくテンプレート１０２は、物理キュー名を指

定するためのパラメータ１０４と、関連の仮想キュー１０７を指定するためのパラメータのフィールド１０６とを含む。仮想キューパラメータは、仮想キューアドレス１０８、幅１１０、ストライド１１２、およびサイズ１１４を含む。仮想キュー１０７に対するアクセスは一定のストライドを有し、これはメモリ空間におけるシーケンシャルではないデータがキューにストリーム化されることを可能にする。ＴＤＴＰ１４は、仮想および物理キューの間で、方向パラメータ１１６の値に従ってデータを転送する。方向は、キューがクラスタ３４機能ユニット内でソースまたはデスティネーションのどちらとして用いられるかを決定する。仮想キューメモリ空間は循環状にアドレスされ、かつ一度にいくつかの物理キューに関連付けられることができる。たとえば、同じ仮想キューが、ソースオペランドキューおよびデスティネーションオペランドキューに関連付けられることができる。機能ユニットはメモリ空間におけるソースキューデータを消費する一方で、その結果でデスティネーションキューメモリ空間を埋める。

【００３８】

キューに基づくデータ転送は、物理キューステータスによって制御される。ソースオペランドキューに対しては、オンチップメモリからオペランドバッファへの転送は、物理キューがフルになったときに停止する。デスティネーションオペランドキューに対しては、オペランドバッファからオンチップメモリへの転送は物理キューが空になったときに停止する。

【００３９】

テンプレートデータ転送コプロセッサアーキテクチャ

図１０を参照すると、一実施例においてテンプレートデータ転送コプロセッサ１４は、テンプレートインタプリタ１１０、２Ｄアドレス生成ユニット（ＡＧＵ）１１２の群、テンプレートバッファ１１４、パッディングバッファ１１６およびデータ転送バッファ１１８を含む。テンプレートバッファ１１４は、１つ以上のテンプレートタイプ３６、６６、８０、１０２に対してＴＤＴＰを用いるアプリケーションプログラムによって設定されるテンプレートパラメータのような静的なデータフロー情報を含む。テンプレートバッファ１１４はまた、ランタイムに計算されるブロックアドレスのような動的データフロー情報をも含む。

【００４０】

テンプレートインタプリタ１１０は、アクティブなテンプレートエントリのリストを維持し、アクティブなテンプレートごとのランタイムデータ転送パラメータを計算する。ブロックに基づくテンプレートに対しては、すなわちユニブロックおよびプログラム誘導テンプレート３６、６６に対しては、ブロックごとの転送パラメータはプログラムに同期して計算される。

【００４１】

２Ｄ　ＡＧＵ１１２の各々は、２Ｄブロック情報を受け、そのブロックに対してアドレスのシーケンスを生成し、これはデータ転送のためのオンチップバスを駆動するために用いられる。２Ｄ　ＡＧＵ１１２の数は、外部メモリ１６、１８およびオンチップメモリ３０、３２に接続されるオンチップデータバスの数によって決定される。マルチバンクオンチップメモリは、多数のデータフローが最小限のコンフリクトで同時に進行することを可能にする。

【００４２】

異なったテンプレートのタイプは、異なった２Ｄ　ＡＧＵ１１２の組を用い得るが、これは２Ｄ　ＡＧＵが別々のデータバスに関連付けられるためである。たとえば、キューに基づくテンプレート１０２は、オンチップメモリ３０、３２およびオペランドキュー１２０に接続するデータバス２２に結合される２Ｄ　ＡＧＵ１１２の組を用いる。多数のデータフローが同じＡＧＵ１１２を共用し得る。ＡＧＵが利用可能になると、テンプレートインタプリタ１１０は、このＡＧＵを用いることができるアクティブなテンプレートからブロックを選択する。この選択は、ラウンドロビン方式で行なわれる。しかしながら、テンプレートにおいて優先パラメータを用いることにより、代替的な選択規則もまた実施し得る

。
【００４３】
テンプレートの各々は、ソースおよびデスティネーションメモリ空間を規定し、データはソースメモリから転送されて、（必要であれば）データ転送バッファ１１８にストアされる前に整列される。データ転送バッファ１１８の数は、外部メモリ１６、１８とオンチップメモリ３０、３２との間の同時データフローの最大数を決定する。転送バッファ１１８内のデータもまた、（必要であれば）デスティネーションメモリに転送される前に整列される。
【００４４】
さらに、２つの特別な場合を説明する。第１に、間接テンプレート８０に対して、ソースデータアドレスは、ソースアドレスパラメータ９４（図８を参照）にストアされるソースベースアドレスにインデックスデータを加えることにより計算される。関連の２Ｄ　ＡＧＵ１１２は、オペランドを直接オンチップメモリまたはこのインデックスデータをストアするオペランドキューから取ることによりこれらの加算を素早く実行する。第２に、パッディングおよびラッピングデータフローを実現するために、境界のピクセルはＴＤＴＰ１４にロードされる。これらのピクセルはパッディングバッファ１１６にストアされて、他のデータとともにデータ整列ユニットにシンクロナスに挿入される。
【００４５】
テンプレートインタプリタ１１０は、データ転送を制御し、メディアプロセッサクラスタ３４およびＡＧＵ１１２からの信号に応答する。図１１を参照すると、テンプレートインタプリタ１１０の制御フロー１２２が示される。テンプレートインタプリタ１１０は、ステップ１２４においてクラスタ３４の信号または次の利用可能な２Ｄ　ＡＧＵ１１２を待機する。４つの潜在的な動作のうちの１つが、受信される信号に応じて発生する。クラスタ信号「Ａ」に対しては、インタプリタはステップ１２６において新しいテンプレートを初期化する。これを行なうために、インタプリタ１１０は、テンプレートタイプに従って適切なテンプレートセットアップモジュールを呼出し、ステップ１２７において転送されるべき最初のブロックを決定する。クラスタ信号「Ｂ」に対しては、インタプリタ１１０はステップ１２８においてテンプレートを削除することによりテンプレート転送を終了させる。テンプレートに対するブロック転送を行なうためのクラスタ信号「Ｃ」に対しては、インタプリタはステップ１３０において現在のブロックを準備モードに設定する。ステップ１３２において、インタプリタは、最後のブロックに到達したか否かをテストする。もし到達していなければ、次に転送されるべきブロックがステップ１３４において決定される。テンプレートで指定されるブロックがない場合、インタプリタ１１０はクラスタ３４に（クラスタによってポーリングされるフラグを設定することにより）信号を送り、テンプレートの終了を示す。
【００４６】
第４の起こり得る動作は、ＡＧＵ１１２のうちの１つのテンプレートデータ転送コプロセッサ１４内で生成される。２Ｄ　ＡＧＵユニット１１２が利用可能になると、ＡＧＵはインタプリタ１１０に対して信号「Ｄ」を生成する。次いでステップ１３６において、インタプリタ１１０はテンプレートを選択し、準備されたブロックをＡＧＵユニットに対してディスパッチする。２Ｄ　ＡＧＵはまた、ブロック転送の各々が完了した場合にインタプリタに信号を送る。
【００４７】
以下の例は、テンプレートデータ転送のプログラミングインターフェイスを示す。第１の例においては、関数はset_uniblockルーチン（ライン１－２）を用いて２つのブロックに基づくデータフローを生成するが、１つは入力データのためであり、他方は出力データのためのものである。set_uniblockルーチンはＴＤＴＰ１４がテンプレートを指定し、テンプレートバッファ１１４にパラメータをコピーし、ハンドラにそのテンプレートを返すことを要求する。データフローは、transferルーチン（ライン３、６および１０）によって開始される。transferルーチンへのコールの各々は、データのブロックをソースメモ

リからデスティネーションメモリに転送する。テンプレートで指定されたデータのすべてが転送されると、さらなる転送は無効にされる。

【００４８】

waitルーチン（ライン５および９）は、テンプレートに関連の準備信号をポーリングすることによりブロック転送が完了することを待機する。テンプレートが生成されると、その準備信号が自動的に設定される。ＴＤＴＰは、ブロックの転送を開始するときに準備信号をリセットする。準備信号は、ブロック転送が完了したときに設定される。タイトループコールは、クラスタ３４とＴＤＴＰ１４との間の共通のデータ構造から入力および出力データブロックに関するアドレスおよびサイズ情報を受ける。タイトループがクラスタ３４で実行される一方で、次の入力データブロックおよび先行する出力データブロックはＴＤＴＰ１４によって転送される。テンプレートに対するすべてのデータが転送されると、ＴＤＴＰ１４はテンプレート完了信号を設定し、これもまたプログラムにおいてループ制御のために用いられる。

【００４９】

deleteルーチン（ライン１１－１２）は、関連のテンプレートバッファを無効化することによりテンプレートを終了させる。例１に示されるように、ＴＤＴＰ１４はアプリケーションプログラムに対して効率的で簡単なプログラミングインターフェイスの組を提供する。特に、データフローの詳細は、適切なテンプレートパラメータを選択することによりＴＤＴＰ１４において隠されている。

【００５０】

例１：ユニブロックテンプレート転送

【００５１】

【表１】

```
Function() {
input_comm = set_uniblock(input data flow template parameters);     /*----- ライン1 -----*
output_comm = set_uniblock(output data flow template parameters); /*-----ライン2 ------*/
transfer(input_comm);                                              /*------ライン3 ------*/
while(output_comm.not finished) {                                  /*------ ライン4 ------*/
        wait(input_comm);                                          /*------ライン5 ------*/
        input_block = get_dst(input_comm);                         /*------ライン6 ------*/
        transfer(input_comm);                                      /*------ライン6 ------*/
        output_block = get_nextsrc(output_comm);                   /*------ライン7 ------*/
        if(input_comm.not finished)
        function_tight_loop(input_block, output_block, ...);       /*------ライン8 ------*/
        wait(output_comm);                                         /*------ライン9 ------*/
        transfer(output_comm);                                     /*------ライン10 ------*/
    }
    delete(input_comm);                                            /*------ライン11 ------*/
    delete(output_comm);                                           /*------ライン12 ----*/
}
```

【００５２】

プログラム誘導テンプレートは、ユニブロックテンプレートに加えてデータブロックをダブルバッファリングするために用い得るので、そのプログラミングインターフェイスは例１のものに似ているが、テンプレートがset_guidedルーチンによって設定され、ブロック記述テーブルがデータフローを活性化する前に誘導テンプレートに対してオンチップメモリで確立される点が異なる。

【００５３】
第２の例は、間接データフローのためのプログラミングインターフェイスを示す。間接データフローは、set__indirectルーチン（ライン１）によって生成され、これはＴＤＴＰ１４がテンプレートバッファ１１４における間接テンプレートを指定し、次いで指定されたテンプレートにパラメータを転送することを要求する。ルーチンはまた、ハンドラを返し、これによりプログラムがテンプレートに関するステータス情報にアクセスすることができる。間接データ転送は、transferルーチン（ライン２）を用いることにより開始され、その後に他のタスク（ライン３）が続く。同じ時点で、プログラムは間接データ転送が完了するのを待機し、次いで転送されたデータを計算する。deleteはテンプレートを終了させる。

【００５４】
例２：間接テンプレートデータ転送
【００５５】
【表２】

```
Function(){
    indirect_comm = set_indirect(parameter list)        /*------ライン 1 -----*/
    transfer(indirect_comm);                            /*------ライン 2 -----*/
    ……                                                  /*------ライン 3 -----*/
    wait(indirect_comm);                                /*------ライン 4 -----*/
    ……                                                  /*------ライン 5 -----*/
    delete(indirect_comm);                              /*------ライン 6 -----*/
}
```

【００５６】
例３は、キューに基づくデータフローインターフェイスを示す。ブロックに基づくデータフローおよび間接データフローと同様に、set__voq（ライン１）がテンプレートを生成するために用いられ、transferルーチン（ライン２）がデータ転送を開始するために用いられる。しかしながら、プログラムは転送が完了するのを待機することなく、transferルーチンのすぐ後にキューを使い始める。

【００５７】
例３：キューに基づくテンプレート転送
【００５８】
【表３】

```
Function(){
    queue_comm = set_voq(parameter list);               /*------ライン 1 -----*/
    transfer(queue_comm);                               /*------ライン 2 -----*/
    ……                                                  /*------ライン 3 -----*/
    delete(queue_comm);                                 /*------ライン 4 -----*/
}
```

【００５９】
価値のある有利な効果
パラメータ化されたテンプレートの１つの利点は、そのようなテンプレートはブロック転送ごとに低レベルデータ転送パラメータを計算し設定するのではなく、プログラマが特定のアルゴリズムのために必要なデータフローを容易に設計するために効率的で汎用性の高

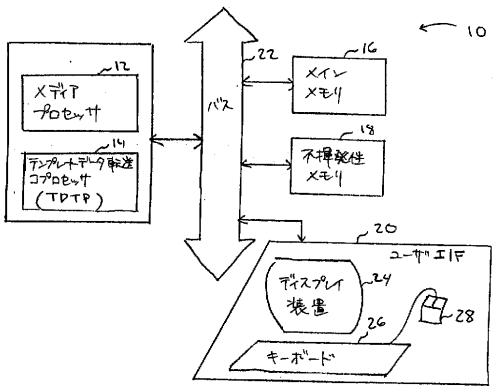
い機構であることである。
【００６０】
ユニブロックテンプレートの利点は、オンチップメモリにおける入力および出力データブロックがダブルバッファリングされることが可能であり、それによりメディアプロセッサ機能ユニットは、高速計算のためにオンチップメモリにおけるデータにアクセスするだけでよく、一方で遅い外部メモリとの間のデータ転送は機能ユニットから隠されることである。また、ブロックごとに、テンプレートデータ転送コプロセッサによってテンプレートからアドレスおよびサイズが計算される。したがって、データフロープログラミングは簡略化され、ＣＰＵの計算負荷は減じられる。別の利点とは、ユニブロックテンプレートがソースメモリ空間からオーバーラップされるべきブロックを指定することである。したがって、境界のブロックはパッディングされるかまたはラップアラウンドされ、境界に沿った必要なオーバーラッピングを準備する。
【００６１】
パッディングおよびラッピングは実際のデータ転送の間に行なわれる。したがって、メディアプロセッサ計算はさらに減じられ、かつプログラマの負担は、テンプレートデータ転送コプロセッサによってこれらの不規則なデータコピー動作が自動的に扱われるという事実により、減じられる。
【００６２】
間接テンプレートの利点は、これがプロセッサの計算エンジンに対して顕著な負荷を生成することなく、ランダムな場所からデータをアセンブルするための効率的な方法を提供することである。
【００６３】
キューに基づくデータフローテンプレートの利点は、仮想キューと物理キューとの間のデータ転送を促進することである。キューに基づくデータフローは、ロード／ストア命令をなくし、レジスタファイルサイズに対する圧力を緩和する。さらに、コードスケジューリングは長いレイテンシロード／ストア命令をなくして簡略化され、高性能のために必要となるループアンローリングの量は減じられ、よりコンパクトなコードをもたらす。
【００６４】
この発明の好ましい実施例を例示し説明したが、さまざまな代替例、変形および等価物を用い得る。したがって、上述の説明は前掲の特許請求の範囲によって規定されるこの発明の範囲を限定するものと解されてはならない。
【図面の簡単な説明】
【図１】　この発明の実施例に従った、メディアプロセッサおよびテンプレートデータ転送コプロセッサを有する画像／ビデオ処理システムのブロック図である。
【図２】　例示的なメディアプロセッサ実施例のブロック図である。
【図３】　図１のテンプレートデータ転送コプロセッサによって実現されるデータ転送のためのユニブロックテンプレートの図である。
【図４】　１組の画像データブロックの間のソースブロックに対するパッディング領域の図である。
【図５】　１組の画像データブロックの間のソースブロックに対するデータラッピングの図である。
【図６】　データブロックに対するアクセスパターンを示す図である。
【図７】　図１のテンプレートデータ転送コプロセッサによって実現されるデータ転送のためのプログラム誘導テンプレートの図である。
【図８】　図１のテンプレートデータ転送コプロセッサによって実現されるデータ転送のための間接テンプレートの図である。
【図９】　図１のテンプレートデータ転送コプロセッサによって実現されるデータ転送のためのキューに基づくテンプレートの図である。
【図１０】　この発明の実施例に従った、図１のテンプレートデータ転送コプロセッサのブロック図である。

【図１１】　図１０のテンプレートインタプリタに対する処理状況のフローチャートである。

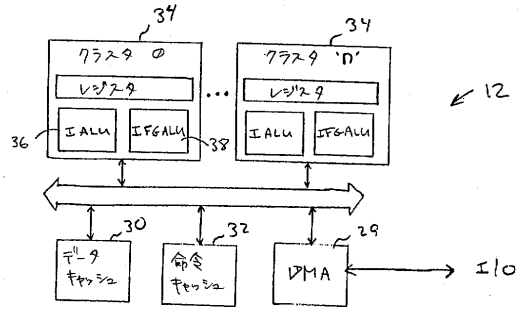
【符号の説明】

１２　メディアプロセッサ、１４　テンプレートデータ転送コプロセッサ、３６　ユニブロックテンプレート。

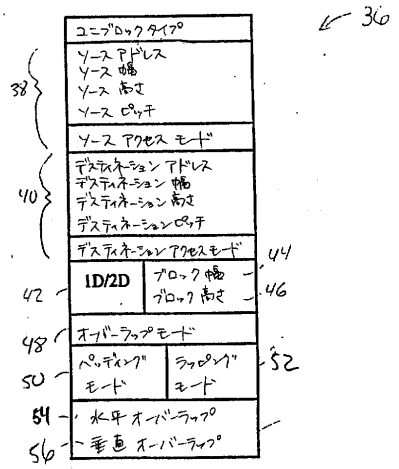
【図１】



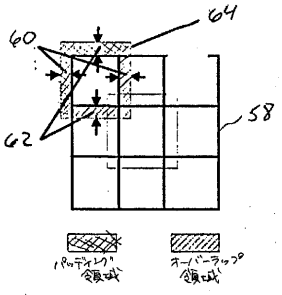
【図２】



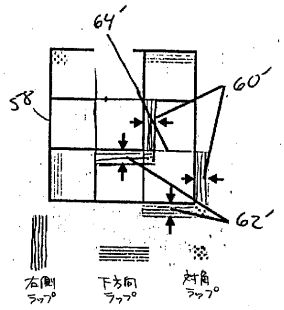
【図３】



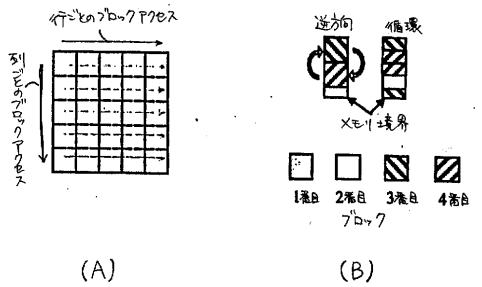
【図４】



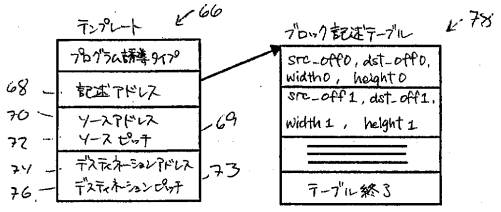
【図５】



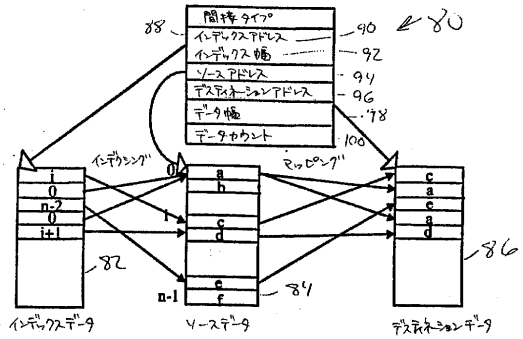
【図６】



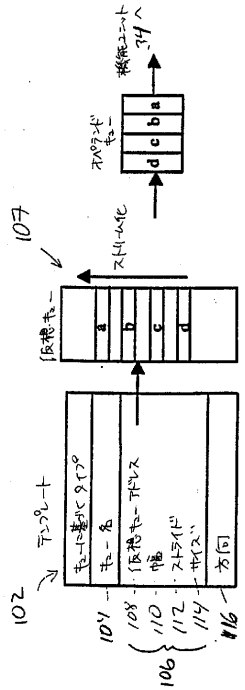
【図７】



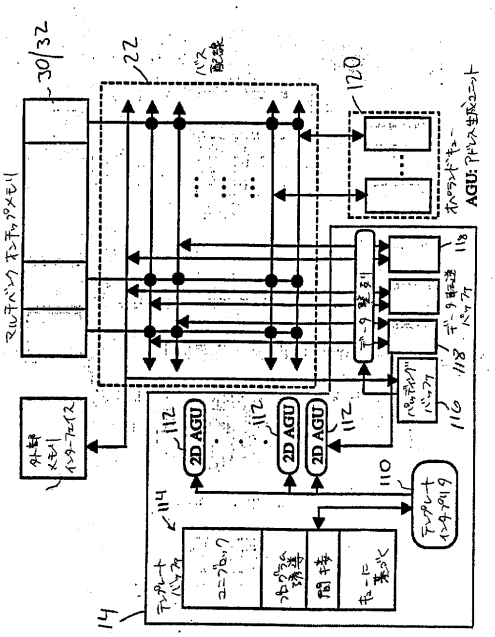
【図８】



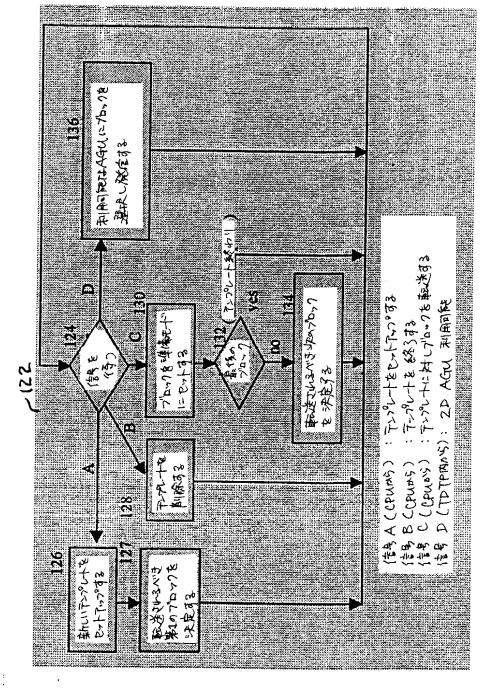
【図９】



【図１０】



【図１１】


122
124
信号 A (CPUから) : テンプレートをセットする
信号 B (CPUから) : テンプレートを 終了する
信号 C (CPUから) : テンプレートに対しブロックを転送する
信号 D (TDTPから) : 2D Accu 利用開始

フロントページの続き

(74)代理人 100096781
弁理士 堀井 豊
(74)代理人 100096792
弁理士 森下 八郎
(72)発明者 ウェイユン・スン
アメリカ合衆国、９８１２５ ワシントン州、シアトル、ワンハンドレッドアンドシックスス・ストリート、エヌ・イー、８１８、アパートメント・４１０
(72)発明者 ドンロク・キム
アメリカ合衆国、９８１０５ ワシントン州、シアトル、ミサン・プレイス・エヌ・イー、５２９０
(72)発明者 ヨンミン・キム
アメリカ合衆国、９８１５５ ワシントン州、シアトル、エヌ・イー・ワンハンドレッドアンドエイティナインス・プレイス、４４３１

審査官 横山 佳弘

(56)参考文献 特開平０４－２４８５９０（ＪＰ，Ａ）
特開平０５－２３３２８７（ＪＰ，Ａ）
特開２００１－２０９３７０（ＪＰ，Ａ）
特開平０５－２０４８２９（ＪＰ，Ａ）
特開平０１－２２８０５０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 3/153
G06F 9/38
G06F 13/28
G06T 1/20