

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6944974号
(P6944974)

(45) 発行日 令和3年10月6日 (2021. 10. 6)

(24) 登録日 令和3年9月15日 (2021. 9. 15)

(51) Int. Cl.	F I
G06F 9/30 (2018.01)	G06F 9/30 350F
G06F 9/312 (2006.01)	G06F 9/312 L
G06F 9/345 (2006.01)	G06F 9/312 W
	G06F 9/345 S

請求項の数 26 外国語出願 (全 40 頁)

(21) 出願番号	特願2019-113316 (P2019-113316)	(73) 特許権者	518371892
(22) 出願日	令和1年6月19日 (2019. 6. 19)		グラフィコア リミテッド
(65) 公開番号	特開2020-109604 (P2020-109604A)		Graphcore Limited
(43) 公開日	令和2年7月16日 (2020. 7. 16)		イギリス国, ビーエス1 2ピーエイチ,
審査請求日	令和1年6月28日 (2019. 6. 28)		ブリストル, ワイン ストリート 11-19
(31) 優先権主張番号	1821300.9		11-19 Wine Street, B
(32) 優先日	平成30年12月31日 (2018. 12. 31)	(74) 代理人	100169904
(33) 優先権主張国・地域又は機関	英国 (GB)		弁理士 村井 康司
		(74) 代理人	100121120
			弁理士 渡辺 尚

最終頁に続く

(54) 【発明の名称】 ロード／ストア命令

(57) 【特許請求の範囲】

【請求項 1】

実行ユニットと、メモリと、多数のレジスタを含む1つまたは複数のレジスタファイルとを含むプロセッサであって、前記実行ユニットは、機械語命令のタイプを定義する命令セットからの命令のインスタンスを実行するように構成され、機械語命令の各タイプは、オペコードおよび0またはそれ以上のオペランドからなり、

前記実行ユニットは、ロード／ストアユニットを含み、前記命令セットにおいて定義される前記命令のタイプは、オペランドを有するロード／ストア命令を含み、前記オペランドは、前記1つまたは複数のレジスタファイルのうちの少なくとも1つのレジスタファイルの前記レジスタの中から、2つのロード操作の各々のロード操作のそれぞれの送信先、ストア操作のそれぞれの送信元、および、3つのメモリアドレスを保持するように構成された1対のアドレスレジスタを指定し、前記3つのメモリアドレスは、前記2つのロード操作の各々に対するそれぞれのロードアドレス、および、前記ストア操作に対するストアアドレスであり、

前記ロード／ストア命令は、3つの即値ストライドオペランドをさらに含み、各即値ストライドオペランドは、前記2つのロードアドレスおよび1つのストアアドレスの各々に対するそれぞれのストライド値を指定し、各即値ストライドオペランドの少なくともいくつかの可能な値は、前記1つまたは複数のレジスタファイルのうちの1つのレジスタファイルのストライドレジスタ内の多数のフィールドのうちの1つを指定することによって、前記それぞれのストライド値を指定し、各フィールドは、異なるストライド値を保持し、

10

20

前記ロード／ストアユニットは、前記ロード／ストア命令の前記オペコードに応答して、前記メモリからのデータのそれぞれの部分を前記2つのロードアドレスの各々から前記それぞれのロード操作の前記それぞれの送信先にロードし、データのそれぞれの部分を前記ストア操作の前記送信元から前記メモリ内の前記ストアアドレスに格納し、前記ロードおよびストア操作の各々に続いて、前記それぞれのストライド値の分だけ前記ロードアドレス及び前記ストアアドレスのそれぞれのインクリメントするように構成される、プロセッサ。

【請求項2】

前記ロード／ストア命令は、前記1つまたは複数のレジスタファイルのうちの1つのレジスタファイルの多数の可能なレジスタの中から前記ストライドレジスタを指定するためのストライドレジスタオペランドをさらに含む、請求項1に記載のプロセッサ。

10

【請求項3】

前記ストライドオペランドの1つの可能な値は、1単位のストライド値を指定し、前記ストライドオペランドの他の多数の可能な値は、前記ストライドレジスタの前記フィールドのうちの異なるものを指定する、請求項1または2に記載のプロセッサ。

【請求項4】

前記アドレスレジスタおよびストライドレジスタの前記1つまたは複数のレジスタファイルから前記ロード／ストアユニットへの3つのポートを含み、前記ロード／ストアユニットは、前記1対のアドレスレジスタの各々に対する前記ポートのそれぞれおよび前記ストライドレジスタにアクセスするための前記ポートのうちの1つを使用するように構成される、請求項1～3のいずれか一項に記載のプロセッサ。

20

【請求項5】

前記3つのポートの各々は、アクセスするために使用される前記それぞれのアドレスレジスタまたはストライドレジスタのビット幅と等しいビット幅を有する、請求項4に記載のプロセッサ。

【請求項6】

前記1対のアドレスレジスタの各々は、32ビット幅であり、前記ロードおよびストアアドレスの各々は、21ビット幅である、請求項1～5のいずれか一項に記載のプロセッサ。

【請求項7】

30

前記命令セットにおいて定義される前記命令のタイプは、オペランドを取る算術命令をさらに含み、前記オペランドは、前記1つまたは複数のレジスタファイルのうちの少なくとも1つのレジスタファイルの前記レジスタの中から、第1の入力および第2の入力を受信する対象となる送信元、ならびに、結果が出力される送信先を指定し、

前記プロセッサは、一連の命令を含むプログラムを実行するようにプログラムされ、前記一連の命令は、前記ロード／ストア命令のインスタンスおよび前記算術命令のインスタンスを含み、前記ロード／ストア命令のうちの少なくともいくつかのロード／ストア命令の送信元は、前記算術命令のうちの少なくともいくつかの算術命令の送信先として設定され、前記ロード／ストア命令のうちの少なくともいくつかのロード／ストア命令の送信先は、前記算術命令のうちの少なくともいくつかの算術命令の送信元として設定される、請求項1～6のいずれか一項に記載のプロセッサ。

40

【請求項8】

前記一連の命令は、一連の複数対の命令を含み、命令の各対は、前記ロード／ストア命令のインスタンスおよび算術命令の対応するインスタンスからなり、

命令の各対では、前記ロード／ストア命令の前記送信元は、前記複数対のうちの先行する対からの前記算術命令の前記送信先として設定され、前記ロード／ストア命令の前記送信先は、前記複数対のうちの現在の対または後続の対の前記算術命令の前記送信元として設定される、請求項7に記載のプロセッサ。

【請求項9】

前記入力および前記結果の各々は、少なくとも1つの浮動小数点値を含み、前記実行ユ

50

ニットは、前記算術命令の前記オペコードに応答して算術演算を実行するように構成された浮動小数点演算ユニットを含む、請求項 8 に記載のプロセッサ。

【請求項 10】

前記算術命令は、ベクトルドット積命令、累積ベクトルドット積命令、行列積命令、累積行列積命令または畳み込み命令である、請求項 8 または 9 に記載のプロセッサ。

【請求項 11】

前記複数対の命令の各々は、同時に実行される命令バンドルであり、

前記実行ユニットは、各バンドルの前記ロード／ストア命令の前記インスタンスを実行するように構成された前記ロード／ストアユニットを含む第 1 実行ユニットと、前記算術命令の前記オペコードに応答して算術演算を実行するように構成された浮動小数点演算ユニットを含む第 2 実行ユニットと、を含む請求項 9 または 10 に記載のプロセッサ。

10

【請求項 12】

前記算術命令は、N 要素入力ベクトルに M 個のカーネルの $M \times N$ 行列を乗じるための累積行列積命令であり、前記 M 個のカーネルの各々は、N 要素ベクトルであり、前記累積行列積命令は、N 1 個の数の連続位相の中から位相を指定する即値オペランドをさらに取り、

前記一連の命令は、ループ内で繰り返すシーケンスを含み、各ループ内の前記シーケンスは、N 1 個の前記複数対の命令のシーケンスを含み、前記シーケンス内の各連続対の前記累積行列積命令の前記インスタンスは、初期位相から N 1 番目の位相までの前記シーケンスの異なる連続位相を指定する位相オペランドの異なる値を有し、

20

各位相では、前記第 1 の入力、前記入力ベクトルのそれぞれのサブベクトルであり、前記第 2 の入力、1 つまたは複数の部分和のそれぞれのセットであり、各入力サブベクトルの要素の数は、 $N_2 = N / N_1$ であり、各セットの部分和の数は、 $N_p = M / N_1$ であり、

各ループの各位相の算術演算は、

次のループで使用するために前記累積行列積命令の前記それぞれのインスタンスの前記第 2 の入力からの N_p 個の部分和の前記それぞれのセットを一時的なプロパゲータ状態にコピーするステップと、

前記 M 個のカーネルの各々に対し、前記現在の位相の前記それぞれの入力サブベクトルと前記カーネルのそれぞれの N_2 要素サブベクトルとのドット積を実行し、それにより、前記 M 個のカーネルの各々に対する対応する中間結果を生成するステップと、

30

前記位相オペランドが前記初期位相を指定する場合は、以前のループからの前記対応する部分積を前記 M 個の中間結果の各々に加えるステップと、

前記対応する中間結果をアキュムレータ状態の前記 M 個の要素の各々に加えるステップと、

前記累積行列積命令の前記それぞれのインスタンスの前記送信先に出力結果として前記現在のまたは以前のループからの前記アキュムレータ状態の N_p 個の前記要素のそれぞれのサブセットを出力するステップと

を含む、請求項 8 ～ 11 のいずれか一項に記載のプロセッサ。

【請求項 13】

40

前記プログラムは、使用可能な出力結果を生成するループの実行前に、有益な結果が生み出さないループを実行する、請求項 12 に記載のプロセッサ。

【請求項 14】

$M = 8$ 、 $N = 16$ 、 $N_1 = 4$ 、 $N_2 = 4$ および $N_p = 2$ である、請求項 12 または 13 に記載のプロセッサ。

【請求項 15】

前記累積行列積命令を繰り返すことで畳み込みを実行し、前記累積行列積命令にて用いられる前記入力ベクトルは、前記累積行列積命令の実行毎に、入力データの異なる部分を抽出して生成され、以前の実行ループからの結果は、後の実行ループに対する部分和として使用される、請求項 12 ～ 14 のいずれか一項に記載のプロセッサ。

50

【請求項 1 6】

前記入力データの部分は、三次元のデータを含み、前記M個のカーネルの各々は、より大きなM個の三次元カーネルのうちの対応するものの一次元構成部分を表し、先のループからの前記結果は、対応する構成一次元カーネルからのデータと各三次元カーネルとの畳み込みを構築するために、後のループに対する前記部分として使用される、請求項 1 5 に記載のプロセッサ。

【請求項 1 7】

前記三次元カーネルの各々は、ニューラルネットワークにおけるノードの重みのセットを表す、請求項 1 6 に記載のプロセッサ。

【請求項 1 8】

前記M個のカーネルのそれぞれは、前記入力データに関する異なる特徴を表す、請求項 1 5 ～ 1 7 のいずれか一項に記載のプロセッサ。

【請求項 1 9】

前記入力ベクトルの各要素は、1 6 ビット浮動小数点値であり、各カーネルの各要素は、1 6 ビット浮動小数点値である、請求項 1 2 ～ 1 8 のいずれか一項に記載のプロセッサ。

【請求項 2 0】

前記N p 個の出力結果の各々は、3 2 ビット浮動小数点値である、請求項 1 2 ～ 1 9 のいずれか一項に記載のプロセッサ。

【請求項 2 1】

前記レジスタファイルは、第 1 のレジスタファイルおよび別個の第 2 のレジスタファイルを含み、前記アドレスレジスタは、前記第 1 のレジスタファイルのレジスタであり、前記ロード／ストア命令の前記送信元および送信先は、前記第 2 のレジスタファイルのレジスタである、請求項 1 ～ 2 0 のいずれか一項に記載のプロセッサ。

【請求項 2 2】

前記ストライドレジスタは、前記第 1 のレジスタファイルのレジスタである、請求項 2 1 に記載のプロセッサ。

【請求項 2 3】

前記算術命令の前記送信元および送信先は、前記第 2 のレジスタファイルのレジスタである、請求項 8 を引用する請求項 2 1 又は 2 2 に記載のプロセッサ。

【請求項 2 4】

前記レジスタファイルは、前記第 1 および第 2 のレジスタファイルとは別の第 3 のレジスタファイルをさらに含み、前記カーネルは、前記第 3 のレジスタファイルに保持される、請求項 1 2 を引用する請求項 2 1 ～ 2 3 のいずれか一項に記載のプロセッサ。

【請求項 2 5】

コンピュータ可読記憶装置上で具体化されるコンピュータプログラムであって、請求項 1 ～ 2 4 のいずれか一項に記載のプロセッサ上で実行するように構成されたコードであって、前記ロード／ストア命令の 1 つまたは複数のインスタンスを含むコードを含む、コンピュータプログラム。

【請求項 2 6】

請求項 1 ～ 2 4 のいずれか一項に従って構成されたプロセッサがプログラムを実行する方法であって、前記実行ユニットを通じて前記プロセッサ上で前記ロード／ストア命令の 1 つまたは複数のインスタンスを含むプログラムを実行するステップを含む、方法。

【発明の詳細な説明】

【技術分野】

【0 0 0 1】

本開示は、高い意味的密度を有するロード／ストア命令の形態に関する。

【背景技術】

【0 0 0 2】

グラフィックス処理装置（GPU）およびデジタル信号プロセッサ（DSP）などの特定のアプリケーション用に設計されたプロセッサの開発への関心が高まってきている。最近関心を集めた別のタイプのアプリケーション特有のプロセッサは、出願人によって「I P U（i n t e l l i g e n c e p r o c e s s i n g u n i t）」（知能処理装置）と呼ばれる機械知能アプリケーション専用のものである。これらは、例えば、ニューラルネットワークなどの知識モデルを訓練するかまたは知識モデルの訓練を補助するため、あるいは、そのようなモデルに基づいて予測もしくは推論を実行するかまたは予測もしくは推論の実行を補助するために、ホストによって割り当てられた仕事を実行するように構成されたアクセラレータプロセッサとして採用することができる。

【0003】

機械知能アルゴリズムは、複数の相互接続ノードのグラフによって表すことができる「知識モデル」に反復更新を実行することに基づく。各ノードは、その入力関数を表す。ノードは、グラフへの入力を受信するものもあれば、1つまたは複数の他のノードからの入力を受信するものもある一方で、ノードの出力は、他のノードの入力を形成するものもあれば、グラフの出力を提供するものもある（いくつかの事例では、所定のノードは、これらのグラフへの入力、グラフからの出力および他のノードとの接続のすべてを有するものさえもあり得る）。さらに、各ノードにおける関数は、1つまたは複数のそれぞれのパラメータ（例えば、重み）によってパラメータ化される。学習段階では、経験に基づく入力データセットに基づいて、グラフが全体として、可能な入力の範囲に対して所望の出力を生成するように、様々なパラメータに対する値を見出すことを目的とする。この学習を行うための様々なアルゴリズムは、確率的勾配降下法に基づく誤差逆伝播アルゴリズムなど、当技術分野において知られている。入力データに基づく複数の反復にわたり、パラメータは、それらの誤差を減少するように徐々に調節され、グラフは、解に向けて収束する。次いで、後続の段階では、学習済みのモデルを使用して、指定された入力セットに与えられる出力の予測を行うこと、または、指定された出力セットに与えられる入力（原因）に関する推論を行うことができる。

【0004】

機械知能アプリケーション用に設計されたプロセッサは、機械知能アプリケーションにおいて一般的に採用されている算術演算を実行するための専用命令をプロセッサの命令セットに含み得る（命令セットは、機械語命令タイプの基本セットであり、プロセッサの実行ユニットは、それぞれのオペコードおよび0またはそれ以上のオペランドによって定義されるタイプの各々を認識するように構成される）。例えば、ニューラルネットワークなどの機械知能アプリケーションにおいて必要な共通の演算は、入力データセットにわたるカーネルの畳み込みであり、カーネルは、ニューラルネットワークのノードの重みを表す。データにわたって大きなサイズのカーネルの畳み込みを実行するため、畳み込みは、複数のベクトルまたは行列積に分解することができ、その各々は、後続の積の出力と累計される部分和を出力する。畳み込みの実行において使用するためのベクトルおよび行列乗算タイプの演算を実行するための専用の算術命令をプロセッサの命令セットに含むプロセッサは、既に存在している。

【発明の概要】

【発明が解決しようとする課題】

【0005】

これらの種類の算術命令は、より簡単かつ伝統的なタイプの算術命令（加算、乗算など）と比べて、多量の意味的コンテンツを単一の命令に含む（すなわち、1つの命令当たり大量の論理を実行する）。しかし、そのような複雑な命令または動作を伴わせるため、ロード／ストア命令も命令セットに提供することが望ましく、ロード／ストア命令もまた、高濃度の意味的情報を単一の命令に含む。

【課題を解決するための手段】

【0006】

本明細書で開示される一態様によれば、実行ユニットと、メモリと、多数のレジスタを

10

20

30

40

50

含む1つまたは複数のレジスタファイルとを含むプロセッサが提供され、実行ユニットは、機械語命令のタイプを定義する命令セットからの命令のインスタンスを実行するように構成され、機械語命令の各タイプは、オペコードおよび0またはそれ以上のオペランドからなる。実行ユニットは、ロード／ストアユニットを含み、命令セットにおいて定義される命令のタイプは、オペランドを有するロード／ストア命令を含み、オペランドは、1つまたは複数のレジスタファイルのうちの少なくとも1つのレジスタファイルのレジスタの中から、2つのロード操作の各々のロード操作のそれぞれの送信先、ストア操作のそれぞれの送信元、および、3つのメモリアドレスを保持するように構成された1対のアドレスレジスタを指定する。3つのメモリアドレスは、2つのロード操作の各々に対するそれぞれのロードアドレス、および、ストア操作に対するそれぞれのストアアドレスである。ロード／ストア命令は、3つの即値ストライドオペランドをさらに含み、各即値ストライドオペランドは、2つのロードアドレスおよび1つのストアアドレスの各々に対するそれぞれのストライド値を指定し、各即値ストライドオペランドの少なくともいくつかの可能な値は、1つまたは複数のレジスタファイルのうちの1つのレジスタファイルのストライドレジスタ内の多数のフィールドのうちの1つを指定することによって、それぞれのストライド値を指定し、各フィールドは、異なるストライド値を保持する。ロード／ストアユニットは、ロード／ストア命令のオペコードに応答して、メモリからのデータのそれぞれの部分を2つのロードアドレスの各々からそれぞれのロード操作のそれぞれの送信先にロードし、データのそれぞれの部分をストア操作の送信元から上記メモリ内のストアアドレスに格納し、ロードおよびストア操作の各々に続いて、それぞれのストライド値の分だけそれぞれのアドレスをインクリメントするように構成される（すなわち、ロード／ストアユニットは、各即値ストライドオペランドの少なくともいくつかの可能な値に対し、ストライドレジスタ内の多数のフィールドのうちの1つからそれぞれのストライド値を取り入れるように構成され、フィールドは、即値ストライドオペランドの値によって指定される）。

10

20

【0007】

このロード／ストア命令は、有利には、高レートかつ低コードオーバーヘッド（高コード密度）で入力データをメモリから流すことおよび結果をメモリに戻すことを可能にする。例えば、このロード／ストア命令は、算術命令によって消費および出力されているレートのそれぞれと同じレートで、かつ、この目的のために必要な命令の比較的低いオーバーヘッドで、メモリにおよびメモリから算術演算の入力および出力を流すために使用することができる。このロード／ストア命令は、例えば、機械学習アプリケーションにおいて、入力データと重みとの畳み込みを実行するために使用される行列乗算または累積行列乗算命令を伴わせるために有益であろう。

30

【0008】

しかし、開示されるロード／ストア命令の適用性は、そのような適用に限定されないことに留意されたい。そのような意味的コンテンツを有するロード／ストアタイプ命令は、複雑な算術命令の他のシーケンスにおいてなど、畳み込みまたは行列乗算を実行する単なる機械学習以外の適用を見つけることもできる。

【0009】

実施形態では、ロード／ストア命令は、1つまたは複数のレジスタファイルのうちの1つのレジスタファイルの多数の可能なレジスタの中からストライドレジスタを指定するためのストライドレジスタオペランドをさらに含み得る。

40

【0010】

実施形態では、ストライドオペランドの1つの可能な値は、1単位のストライド値を指定することができ、ストライドオペランドの他の多数の可能な値は、ストライドレジスタのフィールドのうちの異なるものを指定することができる。

【0011】

すなわち、ロード／ストアユニットは、各即値ストライドオペランドの少なくともいくつかの可能な値に対し、ストライドレジスタのフィールドからストライド値を取り入れる

50

というよりむしろ、それぞれのストライド値を1単位に設定するように構成される（1単位は、1アトム（すなわち、データアクセスのサイズ）を意味し、必ずしも1バイトである必要はない。例えば、オペランドが16ビットの浮動小数点値の4要素ベクトルである場合は、インクリメントは、1単位当たり1アトムであり、これは、8バイト＝64ビットに等しい）。他のいくつかまたはすべてに対しては、ロード／ストアユニットは、即値ストライドオペランドの値によって指定されるように、ストライドレジスタのフィールドのうちの1つからそれぞれのストライド値を取り入れる。

【0012】

実施形態では、プロセッサは、アドレスレジスタおよびストライドレジスタの1つまたは複数のレジスタファイルからロード／ストアユニットへの3つのポートを含み、ロード／ストアユニットは、1対のアドレスレジスタの各々に対するポートのそれぞれおよびストライドレジスタにアクセスするためのポートのうちの1つを使用するように構成することができる。

10

【0013】

実施形態では、3つのポートの各々は、アクセスするために使用されるそれぞれのアドレスレジスタまたはストライドレジスタのビット幅と等しいビット幅を有する。

【0014】

実施形態では、プロセッサは、問題の1つまたは複数のレジスタファイルからロード／ストアユニットへのこれらの単に3つのポートを含むのみである（マルチスレッドプロセッサの事例では、コンテキスト間で共有される）。実施形態では、1対のアドレスレジスタおよびストライドレジスタは、同じレジスタファイル内にあってもよい。実施形態では、アドレスおよびストライドレジスタはすべて、同じビット幅を有してもよい。実施形態では、この幅は、32ビットであってもよい。

20

【0015】

実施形態では、1対のアドレスレジスタの各々は、32ビット幅であってもよく、ロードおよびストアアドレスの各々は、21ビット幅であってもよい。

【0016】

実施形態では、ロードの各々の送信先は、1対の32ビット幅レジスタであってもよく、ストアの送信元は、1対の32ビット幅レジスタであってもよい。

【0017】

30

実施形態では、プロセッサは、メモリからのロードを実行するための2つのポートと、上記メモリへの格納を実行するための1つのポートとを含んでもよい。実施形態では、プロセッサは、メモリからロードするためのこれらの2つのポートのみと、メモリに格納するためのこの1つのポートのみとを含んでもよい（マルチスレッドプロセッサの事例では、コンテキスト間で共有される）。実施形態では、これらのポートの各々は、64ビット幅である。実施形態では、上記ロードの各々によってロードされるデータ部分は、64ビット幅であり、上記ストアによって格納されるデータ部分は、64ビット幅である。

【0018】

実施形態では、上記ロードの各々によってロードされるデータ部分は、1対の32ビット浮動小数点値または16ビット浮動小数点値の4要素ベクトルから構成されていてもよいし、上記ストアによって格納されるデータ部分は、1対の32ビット浮動小数点値から構成されてもよい。

40

【0019】

実施形態では、命令セットにおいて定義される命令のタイプは、オペランドを取り入れる算術命令をさらに含んでもよい。オペランドは、1つまたは複数のレジスタファイルのうちの少なくとも1つのレジスタファイルのレジスタの中から、第1の入力および第2の入力を受信する対象となる送信元ならびに結果が出力される送信先を指定する。プロセッサは、一連の命令を含むプログラムを実行するようにプログラムすることができ、一連の命令は、ロード／ストア命令のインスタンスおよび算術命令のインスタンスを含み、ロード／ストア命令のうちの少なくともいくつかのロード／ストア命令の送信元は、算術命令

50

のうちの少なくともいくつかの算術命令の送信先として設定することができ、ロード／ストア命令のうちの少なくともいくつかのロード／ストア命令の送信先は、算術命令のうちの少なくともいくつかの算術命令の送信元として設定することができる。

【0020】

実施形態では、上記一連の命令は、一連の複数対の命令を含んでもよく、命令の各対は、ロード／ストア命令のインスタンスおよび算術命令の対応するインスタンスからなる。命令の各対では、ロード／ストア命令の送信元は、複数対のうちの先行する対からの算術命令の送信先として設定することができ、ロード／ストア命令の送信先は、上記複数対のうちの現在の対または後続の対の算術命令の送信元として設定することができる。

【0021】

実施形態では、入力および結果の各々は、少なくとも1つの浮動小数点値を含んでもよく、実行ユニットは、算術命令のオペコードに応答して上記算術演算を実行するように構成された浮動小数点算術ユニットを含む。

【0022】

実施形態では、算術命令は、ベクトルドット積命令、累積ベクトルドット積命令、行列積命令、累積行列積命令または畳み込み命令のうちの1つであってもよい。

【0023】

実施形態では、複数対の命令の各々は、同時に実行される命令バンドルであってもよく、プロセッサは、2つの並列パイプラインに分割することができ、第1のパイプラインは、各バンドルのロード／ストア命令のインスタンスを実行するように構成されたロード／ストアユニットを含み、第2のパイプラインは、算術命令の対応するインスタンスを並列して実行するように構成された浮動小数点算術ユニットを含む。

【0024】

実施形態では、算術命令は、N要素入力ベクトルにM個のカーネルのM×N行列を乗じるための累積行列積命令であってもよく、M個のカーネルの各々もまた、N要素ベクトルであり、累積行列積命令は、N1個の数の連続位相の中から位相を指定する即値オペランドをさらに取り入れる。この事例では、一連の命令は、ループ内で繰り返すシーケンスを含んでもよく、各ループ内のシーケンスは、N1個の上記複数対の命令のシーケンスを含み、シーケンス内の各連続対の累積行列積命令のインスタンスは、初期位相からN1番目の位相までのシーケンスの異なる連続位相を指定する位相オペランドの異なる値を有する。各位相では、第1の入力は、入力ベクトルのそれぞれのサブベクトルであり、第2の入力は、1つまたは複数の部分和のそれぞれのセットであり、各入力サブベクトルの要素の数は、 $N_2 = N / N_1$ であり、各セットの部分和の数は、 $N_p = M / N_1$ である。そのような実施形態では、各ループの各位相の上記算術演算は、

— 次のループで使用するために累積行列積命令のそれぞれのインスタンスの第2の入力からの N_p 個の部分和のそれぞれのセットを一時的なプロパゲータ状態にコピーするステップと、

— M個のカーネルの各々に対し、現在の位相のそれぞれの入力サブベクトルとカーネルのそれぞれの N_2 要素サブベクトルとのドット積を実行し、それにより、M個のカーネルの各々に対する対応する中間結果を生成するステップと、

— 位相オペランドが初期位相を指定する場合は、以前のループからの対応する部分積をM個の中間結果の各々に加えるステップと、

— 対応する中間結果をアキュムレータ状態のM個の要素の各々に加えるステップと、

— 累積行列積命令のそれぞれのインスタンスの送信先に出力結果として現在のまたは以前のループからのアキュムレータ状態の N_p 個の要素のそれぞれのサブセットを出力するステップと

を含んでもよい。

【0025】

従って、実施形態は、ループごとにM個の命令のみで、メモリにおよびメモリから入力および出力を同じレートで流しつつ、ループごとに1回ずつN要素入力ベクトルとM×N

10

20

30

40

50

行列との乗算を可能にする。例えば、実施形態では、 $M = 8$ 、 $N = 16$ 、 $N_1 = 4$ 、 $N_2 = 4$ および $N_p = 2$ である。

【0026】

伝播状態およびアキュムレータ状態は、算術ユニットの論理の内部状態であってもよく、複数のレジスタファイルのうちの1つのレジスタファイルのレジスタとして実装することができるか、または、それらの組合せである。

【0027】

実施形態では、プログラムは、使用可能な出力結果を生成する前に、上記ループのうちの少なくとも2つのループのウォームアップ期間を含んでもよい。

【0028】

実施形態では、プロセッサは、畳み込みを実行するために、ループを経るシーケンスを使用するようにプログラムすることができる。この事例では、各ループの入力ベクトルは、入力データ部分の異なるサンプルを表してもよく、以前のループからの結果は、後のループに対する部分和として使用することができる。

【0029】

実施形態では、入力データ部分は、3Dボリュームのデータを含んでもよく、上記M個のカーネルの各々は、より大きなM個の3Dカーネルのうちの対応するものの1D構成部分を表してもよい。この事例では、以前のループからの結果は、対応する構成1Dカーネルからの入力データと各3Dカーネルとの畳み込みを構築するために、後のループに対する部分和として使用することができる。

【0030】

実施形態では、3Dカーネルの各々は、ニューラルネットワークにおけるノードの重みのセットを表してもよい。

【0031】

実施形態では、Mの各値は、入力データと異なる特徴との畳み込みを表してもよい。

【0032】

実施形態では、入力ベクトルの各要素は、16ビット浮動小数点値であってもよく、各カーネルの各要素は、16ビット浮動小数点値である。

【0033】

代替の実施形態では、入力ベクトルの各要素は、例えば、32ビット浮動小数点値であってもよく、各カーネルの各要素は、32ビット浮動小数点値であってもよい。

【0034】

実施形態では、 N_p 個の出力結果の各々は、32ビット浮動小数点値であってもよい。

【0035】

代替の実施形態では、出力結果は、代わりに、例えば、16ビット浮動小数点値であってもよい。

【0036】

実施形態では、レジスタファイルは、第1のレジスタファイルおよび別個の第2のレジスタファイルを含んでもよく、アドレスレジスタは、第1のレジスタファイルのレジスタであり、ロード／ストア命令の送信元および送信先は、第2のレジスタファイルのレジスタである。

【0037】

実施形態では、ストライドレジスタは、第1のファイルのレジスタであってもよい。

【0038】

実施形態では、算術命令の送信元および送信先は、第2のレジスタファイルのレジスタであってもよい。

【0039】

実施形態では、プロセッサは、第1のレジスタファイルからロード／ストアユニットへの3つのポートと、第2のレジスタファイルからロード／ストアユニットへの1つのポートとを含んでもよい。実施形態では、第1のレジスタファイルからロード／ストアユニッ

10

20

30

40

50

トへのこれらの3つのポートの各々は、32ビット幅であってもよい。実施形態では、第2のレジスタファイルからロード／ストアユニットへの1つのポートは、64ビット幅であってもよい。実施形態では、プロセッサは、第1のレジスタファイルからロード／ストアユニットへのこれら3つのポートのみと、第2のレジスタファイルからロード／ストアユニットへの1つのポートとを含むのみであってもよい。

【0040】

実施形態では、プロセッサは、第2のレジスタファイルから算術ユニットへの2つのポートを含んでもよい。実施形態では、これらの2つのポートの各々は、64ビット幅であってもよい。実施形態では、プロセッサは、第2のレジスタファイルから算術ユニットへの単に2つのポートを含むのみであってもよい（マルチスレッドプロセッサの事例では、すべてのコンテキストの間で単に2つのポートが共有されるのみであることを意味する）。

10

【0041】

上記に記載されるパラメータを用いると、各ロード／ストア命令は、1対当たり入力ベクトルの2つの32ビット部分和および4つの16ビット要素、従って、4対のシーケンスの1ループ当たり（1ループ当たり4対）入力ベクトルの8つの32ビット部分和および16の16ビット要素をロードすることができる。また、各ロード／ストア命令は、1対当たり2つの32ビット部分和、従って、4対のシーケンスの1ループ当たり8つの32ビット部分和出力を格納することができる。これにより、対応するレートで、かつ、1ループ当たり単に4つのロード／ストアおよび4つの算術命令インスタンスのコードオーバーヘッドで、メモリにおよびメモリから入力および出力を流している間に、1ループ当たり1回の16要素入力ベクトルと8×16行列との実行が可能になる。

20

【0042】

実施形態では、レジスタファイルは、第1および第2のレジスタファイルとは別の第3のレジスタファイルをさらに含んでもよく、カーネルは、第3のファイルに保持することができる。

【0043】

第1のレジスタファイルは、メモリアドレスおよびアドレスオフセット（ストライドなど）を保持するためのメモリアドレスレジスタファイルであってもよい。第2のレジスタファイルは、算術演算の入力および出力オペランドを保持するための算術レジスタファイルであってもよい。第3のレジスタファイルは、重み（例えば、ニューラルネットワークの）を保持するための重みレジスタファイルであってもよい。

30

【0044】

本発明の別の態様によれば、コンピュータ可読記憶装置上で具体化されるコンピュータプログラムであって、上記または本明細書の他の場所の任意の実施形態のプロセッサ上で実行するように構成されたコードであって、ロード／ストア命令の1つまたは複数のインスタンスを含むコードを含む、コンピュータプログラムが提供される。

【0045】

実施形態では、プログラムは、本明細書で開示される構成のいずれかに従って動作するようにさらに構成することができる。例えば、実施形態では、プログラムは、算術命令の1つまたは複数のインスタンスを含んでもよい。プログラムは、ロード／ストア命令のインスタンスが散在する算術命令のインスタンスを含んでもよい。プログラムは、上記で言及される一連の命令（例えば、ルーピングシーケンス）を含んでもよい。

40

【0046】

本明細書で開示される別の態様によれば、上記または本明細書の他の場所の任意の実施形態に従って構成されたプロセッサを操作する方法であって、実行ユニットを通じてプロセッサ上でロード／ストア命令の1つまたは複数のインスタンスを含むプログラムを実行するステップを含む、方法が提供される。

【0047】

本開示の実施形態を理解することを支援するため、および、そのような実施形態をどの

50

ように実施するかを示すため、単なる例示として、添付の図面を参照する。

【図面の簡単な説明】

【0048】

【図1】例示的なマルチスレッドプロセッサの概略ブロック図である。

【図2】交互に起こる時間スロットのスキームを概略的に示す。

【図3】多数の交互に起こる時間スロットにおいて実行するスーパーバイザスレッドおよび多数のワーカースレッドを概略的に示す。

【図4】例示的なプロセッサの論理ブロック構造を概略的に示す。

【図5】構成プロセッサのアレイを含むプロセッサの概略ブロック図である。

【図6】機械知能アルゴリズムにおいて使用されたグラフの概略図である。

【図7】あるタイプのロード／ストア命令を実装する際に使用するためのアドレスパッキングスキームを概略的に示す。

【図8】ストライドレジスタセットにおける既定のストライド値の構成を概略的に示す。

【図9】ある入力ボリュームのデータと3DカーネルKとの畳み込みを概略的に示す。

【図10】累積行列積命令の位相のシーケンスによって実行された行列乗算を概略的に示す。

【図11】累積行列積命令の動作をさらに示す。

【図12】畳み込みを実行するように構成された累積行列積命令のシーケンスの一連のループの例を与える。

【図13】畳み込み命令の動作を概略的に示す。

【図14】一連の畳み込み命令の例を与える。

【発明を実施するための形態】

【0049】

実施形態の詳細な説明

図1は、本開示の実施形態によるプロセッサ4の例を示す。プロセッサ4は、バレルスレッド処理装置の形態のマルチスレッド処理装置10と、ローカルメモリ11（すなわち、マルチタイルアレイの場合は同じタイル上、または、単一プロセッサチップの場合は同じチップ上）とを含む。バレルスレッド処理装置は、パイプラインの実行時間が、インターリーブされた時間スロットの繰り返しシーケンス（その各々は、所定のスレッドによって占有される）に分割されるタイプのマルチスレッド処理装置である。また、これは、間もなくさらに詳細に論じられるように、並列実行と呼ぶこともできる。メモリ11は、命令メモリ12およびデータメモリ22（異なるアドレス化可能メモリユニットまたは同じアドレス化可能メモリユニットの異なる領域において実装することができる）を含む。命令メモリ12は、処理装置10によって実行される機械語を格納し、データメモリ22は、実行コードによって処理されるデータと、実行コードによって出力されたデータ（例えば、そのような処理の結果として）の両方を格納する。

【0050】

メモリ12は、プログラムの複数の異なるスレッドを格納し、各スレッドは、所定の1つまたは複数のタスクを実行するための命令の対応するシーケンスを含む。本明細書で言及される命令は、単一のオペコードおよび0またはそれ以上のオペランドからなる機械語命令（すなわち、プロセッサの命令セットの基本命令のうちの1つのインスタンス）を意味することに留意されたい。実施形態では、プログラムは、多数のワーカースレッドと、1つまたは複数のスーパーバイザスレッドとして構造化することができるスーパーバイザサブプログラムとを含む。これらについては、間もなくさらに詳細に論じる。

【0051】

マルチスレッドプロセッサは、典型的には並列的に、互いに並んで配置された複数のプログラムスレッドの実行が可能なプロセッサである。並列実行は、スレッドが、共通の実行パイプライン（またはパイプラインの少なくとも共通部分）を共有し、異なるスレッドが、繰り返しサイクルの異なるインターリーブされた時間スロットにおいて、この同じ共有された実行パイプラインを通じてインターリーブされることを意味する。これにより、

10

20

30

40

50

パイプライン遅延を隠蔽する機会が増大するので、性能が向上する。プロセッサは、複数の異なるスレッドに共通の何らかのハードウェア（例えば、共通の命令メモリ、データメモリおよび／または実行ユニット）を含むが、マルチスレッドをサポートするため、プロセッサは、各スレッドに特有の何らかの専用ハードウェアも含む。

【0052】

専用ハードウェアは、並列に実行することができるスレッドの少なくとも各々に対するコンテキストレジスタ26の個別のセット（すなわち、サイクルの1スロット当たり1セット）を含む。マルチスレッドプロセッサについて述べる際、「コンテキスト」は、互いに並んで実行されている複数のスレッドのそれぞれのスレッドのプログラム状態（例えば、プログラムカウンタ値、ステータスおよび現在のオペランド値）を指す。コンテキストレジスタは、それぞれのスレッドのこのプログラム状態を表すためのそれぞれのレジスタを指す。レジスタファイルのレジスタは、レジスタアドレスが命令語においてビットとして固定されるという点で、一般的なメモリとは別個のものであり、メモリアドレスは、命令を実行することによって演算することができる。

10

【0053】

処理装置10内では、命令メモリ12からのスレッドの複数の異なるスレッドを、単一の実行パイプライン13を通じてインターリーブできる（ただし、典型的には、命令メモリに格納された全スレッドのサブセットのみが全プログラムの任意の所定のポイントにおいてインターリーブされ得る）。マルチスレッド処理装置10は、コンテキストレジスタ26の多数のセットを含み、各セット26は、並列に実行することができる複数のスレッドのうちの異なるそれぞれのスレッドの状態（コンテキスト）を表すように構成される。また、マルチスレッド処理装置10は、並列に実行されるスレッドに共通の共有された実行パイプライン13、及び、インターリーブ方法で（例えば、ラウンドロビン方法で）共有されたパイプラインを通じて実行するための並列スレッドをスケジュールするためのスケジューラ24も含む。処理装置10は、多数のスレッドに共通の共有された命令メモリ12と、これもまた多数のスレッドに共通の共有されたデータメモリ22とに接続される。

20

【0054】

実行パイプライン13は、フェッチステージ14と、復号ステージ16と、命令セットアーキテクチャによって定義されるような、算術および論理演算、アドレス計算、ロードおよびストア操作ならびに他の動作を実行することができる実行ユニットを含む実行ステージ18とを含む。

30

【0055】

コンテキストレジスタ26の各セットは、対応する1つまたは複数の制御レジスタを含み、制御レジスタは、それぞれのスレッドに対する少なくともプログラムカウンタ（PC）を含み（スレッドが現在実行している命令アドレスを追跡するため）、また、実施形態では、対応するスレッドの現在の状態（スレッドが現在実行されているかまたは一時停止しているかなど）を記録する1つまたは複数の制御状態レジスタ（CSR）のセットも含む。また、コンテキストレジスタファイル26の各セットは、それぞれのスレッドによって実行される命令のオペランド（すなわち、実行される際にそれぞれのスレッドの命令のオペコードによって定義された動作に基づいて操作される値または定義された動作から生じる値）を一時的に保持するためのオペランドレジスタのそれぞれのセットも含む。レジスタ26の各セットは、1つまたは複数のレジスタファイルにおいて実装することができる。

40

【0056】

フェッチステージ14は、複数のコンテキストの各々のコンテキストのプログラムカウンタ（PC）へアクセスできる。対応するスレッドのそれぞれに対し、フェッチステージ14は、プログラムカウンタによって示されたプログラムメモリ12の次のアドレスからそのスレッドの次の命令をフェッチする。プログラムカウンタは、分岐命令によって分岐されない限り、各実行サイクルを自動的にインクリメントする。次いで、フェッチステージ14は、フェッチした命令を復号するために復号ステージ16に渡し、次いで、復号ス

50

ページ16は、命令を実行するために、命令において指定されたいかなるオペランドレジスタの復号化されたアドレスと共に、復号化された命令を実行ユニット18に渡す。実行ユニット18は、オペランドレジスタおよび制御状態レジスタへアクセスできる。実行ユニットは、算術命令の場合のように、復号化されたレジスタアドレスに基づいて命令を実行するときに、そのオペランドレジスタおよび制御状態レジスタを用いてもよい（例えば、2つのオペランドレジスタの値の加算、乗算、減算または除算を行い、対応するスレッドの別のオペランドレジスタに結果を出力する）。あるいは、命令がメモリアクセス（ロードまたはストア）を定義する場合は、実行ユニット18のロード／ストアロジックは、命令に従って、データメモリからそれぞれのスレッドのオペランドレジスタに値をロードするか、または、それぞれのスレッドのオペランドレジスタからデータメモリ22に値を格納する。

10

【0057】

フェッチステージ14は、スケジューラ24の制御の下で、実行される命令を命令メモリ12からフェッチ可能に接続される。スケジューラ24は、時間スロットの繰り返しシーケンスにおいて次々と並列に実行しているスレッドのセットの各々から命令をフェッチするようにフェッチステージ14を制御するように構成され、従って、間もなくさらに詳細に論じられるように、パイプライン13のリソースを多数の時間的にインタリーブされた時間スロットに分割する。例えば、スケジューリングスキームは、ラウンドロビンまたは重み付けラウンドロビンであり得る。そのような方法で動作するプロセッサに対する別の用語は、バレルスレッドプロセッサである。

20

【0058】

スケジューラ24によって実装されるインタリーブスキームの例は、図2に示されている。ここでは、並列スレッドは、ラウンドロビンスキームに従ってインタリーブされ、それにより、スキームの各ラウンド内では、ラウンドは、時間スロットのシーケンスS0、S1、S2…SJ-1（例えば、J=4またはJ=6）に分割され、各スロットは、それぞれのスレッドを実行するためのものである。典型的には、各スロットは、1つの実行サイクルの長さであり、異なるスロットが均等にサイズ指定されるが、すべての可能な実施形態において必ずしもそうであるとは限らず、例えば、1回の実行ラウンド当たりいくつかのスレッドが他のスレッドより多くのサイクルを獲得する重み付けラウンドロビンスキームも可能である。一般に、バレルスレッドは、均一なラウンドロビンまたは重み付けラウンドロビンスケジュールを採用することができ、後者の場合、重み付けは、固定することも、適応化させることもできる。

30

【0059】

1回の実行ラウンド当たりのシーケンスが何であれ、このパターンは繰り返され、各ラウンドは、複数の時間スロットの各々の対応するインスタンスを含む。従って、本明細書で言及される場合は、時間スロットは、シーケンスにおける繰り返し割り当てられる場所を意味し、シーケンスの特定の繰り返しにおける時間スロットの特定のインスタンスを意味しないことに留意されたい。別の言い方をすれば、スケジューラ24は、パイプライン13の実行サイクルを多数の時間的にインタリーブされた（時分割多重化）実行チャンネルに分配し、各実行チャンネルは、時間スロットの繰り返しシーケンスにおけるそれぞれの時間スロットの再現を含む。示される実施形態では、4つの時間スロットが存在しているが、これは単なる例示を目的としており、他の数も可能である。例えば、好ましい実施形態では、実際に、6つの時間スロットが存在する。

40

【0060】

実施形態では、コンテキストレジスタ26は、並列に実行することができるJ個の数のスレッド（示される例では、J=3であるが、これに限定されない）の各々に対するワーカーコンテキストレジスタCX0…CX(J-1)のそれぞれのセットと、1つの追加のスーパーバイザコンテキストレジスタファイルCXSとを含む。ワーカーコンテキストレジスタファイルは、ワーカースレッドのコンテキストを保持し、スーパーバイザコンテキストレジスタファイルは、スーパーバイザスレッドのコンテキストを保持する。実施形態

50

では、スーパーバイザコンテキストは、ワーカーの各々とは異なる数のレジスタを有することに留意されたい。従って、処理装置10は、時間スロットよりも1つ多いコンテキストレジスタファイル26を含む（すなわち、処理装置10は、バレルスレッドが可能なインターリーブされた時間スロットの数よりも1つ多いコンテキストをサポートする）。

【0061】

ワーカーコンテキストC X 0 … C X Jの各々は、プログラマが望むアプリケーション特有の演算タスクは何でも実行するために、4つの実行時間スロットS 0 … S Jのうちの1つに現在割り当てられている多数のワークスレッドのそれぞれのワークスレッドの状態を表すために使用される（この場合もやはり、このワーカーコンテキストは、命令メモリ12に格納されるようなプログラムのワークスレッドの総数の単なるサブセットであり得ることに留意されたい）。追加のコンテキストC X Sは、「スーパーバイザスレッド」（S V）の状態を表すために使用され、その役割は、少なくとも実行のためにどのワークスレッドWを全プログラムのどのポイントにおいてどの時間スロットS 0、S 1、S 2 …に割り当てるかという意味で、ワークスレッドの実行を調整することである。任意に、スーパーバイザスレッドは、外部の交換またはバリア同期の実行などの他の「オーバーシーア」または調整責任を有し得る。当然ながら、J = 4の示される事例は、例示を目的とする実装形態の単なる一例であることに留意されたい。例えば、別の実装形態では、J = 6（6つの時間スロット、6つのワーカーコンテキストおよび1つのスーパーバイザコンテキスト）である。

【0062】

図3を参照すると、実施形態では、スーパーバイザスレッドS Vは、インターリーブされた実行時間スロットのスキームにそれ自体の時間スロットを有さない。また、ワークスレッドへのスロットの割り当ては柔軟に定義されるため、ワーカーも同様である。むしろ、各時間スロットは、ワーカーコンテキストを格納するためのコンテキストレジスタのそれ自体のセットを有し、そのセットは、ワーカーによって、スロットがワーカーに割り当てられる際に使用されるが、スロットがスーパーバイザに割り当てられる際には使用されない。所定のスロットがスーパーバイザに割り当てられる際は、そのスロットは代わりに、スーパーバイザのコンテキストレジスタファイルC X Sを使用する。スーパーバイザは、常に、それ自体のコンテキストへのアクセスを有し、ワーカーは、スーパーバイザコンテキストレジスタファイルC X Sを占有することはできない。

【0063】

スーパーバイザスレッドS Vは、ありとあらゆる時間スロットS 0 … S 3（またはより一般的にはS 0 … S J - 1）において動作する能力を有する。スケジューラ24は、プログラムが全体として開始されると、スーパーバイザスレッドを時間スロットのすべてに割り当てることで開始されるよう、すなわち、スーパーバイザS VがS 0 … S J - 1のすべてにおいて動作を開始するように構成される。しかし、スーパーバイザスレッドには、後続のある時点で（すぐにまたは1つもしくは複数のスーパーバイザタスクを実行した後）、実行しているスロットの各々をワークスレッドのそれぞれに（例えば、図3に示される例では、最初にワーカーW 0 … W 3（またはより一般的にはW 0 … W J - 1）に）一時的に委ねるためのメカニズムが提供される。これは、命令メモリ12のワークスレッドのアドレスをオペランドとして少なくとも取る実行（run）命令を実行するスーパーバイザスレッドによって達成される。ワークスレッドは、互いに並列に実行することができるコードの部分であり、その各々は、実行される1つまたは複数のそれぞれの演算タスクを表す。

【0064】

実行命令は、この命令自体が実行されている現在の時間スロットを、オペランドによって指定されたワークスレッドに委ねるように、スケジューラ24に作用する。実行命令では、委ねられるものが、この命令が実行される時間スロットであることは暗黙的であることに留意されたい（機械語命令の文脈では、暗黙的であるということは、オペランドがこれを指定する必要がないということ、すなわち、オペコード自体から暗黙のうちに理解

されていることを意味する)。従って、渡される時間スロットは、スーパーバイザが実行命令を実行する時間スロットである。

【0065】

スーパーバイザスレッドSVは、ワーカースレッドW0…WJ-1の異なるそれぞれのワーカースレッド(命令メモリ12の可能なワーカースレッドのより大きなセットから選択される)にその時間スロットのいくつかまたはすべてを渡すために、複数の時間スロットのうちの1つまたは複数の他の時間スロットの各々において同様の動作を実行する。最後のスロットに対してそのように行われた時点で、スーパーバイザは、中断される(次いで、後に、スロットのうちの1つがワーカーWから返された際に、中断したところから再開される)。従って、スーパーバイザスレッドSVは、各々が1つまたは複数のタスクを実行する異なるワーカースレッドを、インターリーブされた実行時間スロットS0…SJ-1(例えば、示されるようにJ=4、または、J=6)の異なるインターリーブされた実行時間スロットに割り当てることができる。ワーカースレッドを実行する時間であるとスーパーバイザスレッドが決定すると、スーパーバイザスレッドは、実行命令を使用して、実行命令が実行された時間スロットにこのワーカーを割り当てる。

10

【0066】

いくつかの実施形態では、命令セットは、「すべてを実行する(run-all)」という実行命令の変形形態も含み得る。この命令は、複数のワーカーのセットを共に起動するために使用され、すべてが同じコードを実行する。実施形態では、この命令は、処理装置のスロットS0…S3(またはより一般的にはS0…S(J-1))内のワーカーを1つ残らず起動する。

20

【0067】

起動された時点で、現在割り当てられているワーカースレッドW0…WJ-1の各々は、それぞれの実行命令によって指定されたコードにおいて定義された1つまたは複数の演算タスクの実行に進む。次いで、この実行の終了時には、それぞれのワーカースレッドは、実行している時間スロットをスーパーバイザスレッドに返す。このことは、それぞれのワーカースレッドにおいて終了(exit)命令を実行することによって達成される。終了命令は、この命令自体が実行されている現在の時間スロットをスーパーバイザスレッドに戻すように、スケジューラ24に作用する。それに応答して、スケジューラ24は、そのスロットにおいてスーパーバイザの実行を続ける。

30

【0068】

図4は、実行ユニット18およびコンテキストレジスタ26の詳細を含むプロセッサ4のさらなる例示的な詳細を示す。プロセッサは、並列実行が可能なM個のスレッドの各々に対するそれぞれの命令バッファ53を含む。コンテキストレジスタ26は、ワーカーMコンテキストおよびスーパーバイザコンテキストの各々に対するそれぞれのメインレジスタファイル(MRF)26Mを含む。コンテキストレジスタは、ワーカーコンテキストの少なくとも各々に対する対応する補助レジスタファイル(ARF)26Aをさらに含む。コンテキストレジスタ26は、共通の重みレジスタファイル(WRF)26Wをさらに含む。現在実行しているワーカースレッドのすべては、重みレジスタファイル(WRF)からの読み出しアクセス可能である。スーパーバイザスレッドをWRFに書き込むことができる唯一のスレッドとするよう、WRFはスーパーバイザコンテキストと関連付けられてもよい。また、コンテキストレジスタ26は、スーパーバイザおよびワーカーコンテキストの各々に対する制御状態レジスタ26CSRの対応するグループも含んでもよい。実行ユニット18は、メイン実行ユニット18Mおよび補助実行ユニット18Aを含む。メイン実行ユニット18Mは、ロード/ストアユニット(LSU)55および整数演算論理ユニット(IALU)56を含む。補助実行ユニット18Aは、少なくとも浮動小数点演算ユニット(FPU)を含む。

40

【0069】

J個のインターリーブされた時間スロットS0…SJ-1の各々では、スケジューラ24は、命令メモリ12から、現在の時間スロットに対応するJ個の命令バッファ53のそ

50

れぞれに、対応するスレッドの少なくとも1つの命令をフェッチするようにフェッチステージ14を制御する。実施形態では、各時間スロットは、プロセッサの1つの実行サイクルであるが、他のスキーム（例えば、重み付けラウンドロビン）も除外されない。プロセッサ4の各実行サイクル（すなわち、プログラムカウンタ時間を記録するプロセッサクロックの各サイクル）では、フェッチステージ14は、実装形態に応じて、単一の命令または小さな「命令バンドル」（例えば、2命令バンドルまたは4命令バンドル）をフェッチする。次いで、各命令は、復号ステージ16を介して、命令がメモリアクセス命令であるか、整数演算命令であるか、浮動小数点演算命令であるかに応じてそれぞれ（そのオペコードに従って）、メイン実行ユニット18MのLSU 55もしくはIALU 56または補助実行ユニット18AのFPUのうちの1つに発行（issue）される。メイン実行ユニット18MのLSU 55およびIALU 56は、MRF 26Mからのレジスタを使用してそれらの命令を実行し、MRF 26M内の特定のレジスタは、命令のオペランドによって指定される。補助実行ユニット18AのFPUは、ARF 26AおよびWRF 26Wのレジスタを使用して演算を実行し、ARF内の特定のレジスタは、命令のオペランドによって指定される。実施形態では、WRFのレジスタは、命令タイプにおいて暗黙的であり得る（すなわち、その命令タイプに対して事前に決定される）。また、補助実行ユニット18Aは、浮動小数点演算命令のタイプのうちの1つまたは複数の演算を実行する際に使用するためのいくつかの内部状態57を保持するための補助実行ユニット18Aの内部にある論理ラッチの形態の回路も含んでもよい。

10

【0070】

20

バンドルの命令をフェッチして実行する実施形態では、所定の命令バンドルの個々の命令は、下流の独立したパイプライン18M、18A（図4に示される）において並列して、同時に実行される。2つの命令のバンドルを実行する実施形態では、2つの命令は、下流のそれぞれの補助およびメインパイプラインにおいて同時に実行することができる。この場合、メインパイプラインは、MRFを使用するタイプの命令を実行するように構成され、補助パイプラインは、ARFを使用するタイプの命令を実行するために使用される。命令の適切な相補バンドルへのペアリング（pairing）は、コンパイラによって実現（handled）してもよい。

【0071】

各ワーカースレッドコンテキストは、メインレジスタファイル（MRF）26Mおよび補助レジスタファイル（ARF）26Aのそれ自体のインスタンス（すなわち、バレルスレッドスロットの各々に対して1つのMRFおよび1つのARF）を有する。MRFまたはARFに関連して本明細書で説明される機能は、コンテキスト別に動作するものと理解されたい。しかし、スレッド間で共有される単一の共有された重みレジスタファイル（WRF）が存在する。各スレッドは、それ自体のコンテキスト26のMRFおよびARFのみにアクセスすることができる。しかし、現在実行しているワーカースレッドはすべて、共通のWRFにアクセスすることができる。従って、WRFは、すべてのワーカースレッドによる使用のための重みの共通のセットを提供する。実施形態では、スーパーバイザのみがWRFに書き込むことができ、ワーカーは、WRFから読み取ることしかできない。

30

【0072】

40

プロセッサ4の命令セットは、少なくとも1つのタイプのロード（load）命令を含み、そのロード命令は、実行されると、そのオペコードが、データメモリ22からロード命令が実行されたスレッドのそれぞれのARF 26AへのデータのロードをLSU 55に行わせる。ARF内の送信先の場所は、ロード命令のオペランドによって指定される。ロード命令の別のオペランドは、それぞれのMRF 26Mのアドレスレジスタを指定し、それぞれのMRF 26Mのアドレスレジスタは、データをロードするためのデータメモリ22のアドレスへのポインタを保持する。また、プロセッサ4の命令セットは、少なくとも1つのタイプのストア（store）命令も含み、そのストア命令は、実行されると、そのオペコードが、ストア命令が実行されたスレッドのそれぞれのARFからデータメモリ22へのデータの格納をLSU 55に行わせる。ARF内のストアの送信元の場所は、

50

ストア命令のオペランドによって指定される。ストア命令の別のオペランドは、M R Fのアドレスレジスタを指定し、M R Fのアドレスレジスタは、データを格納するためのデータメモリ 2 2のアドレスへのポインタを保持する。一般に、命令セットは、別個のロードおよびストア命令タイプ、ならびに／あるいは、ロードおよびストア操作を組み合わせて単一の命令にした少なくとも1つのロード／ストア命令タイプを含んでいてもよい。間もなくさらに詳細に論じられるように、命令セットは、2つのロード操作および1つのストア操作をすべて単一の命令で実行する特定のタイプのロード／ストア命令を含んでいてもよい。所定のプロセッサ 4の命令セットは、複数の異なる種類のロード、ストアおよび／またはロード／ストア命令タイプを含んでもよいことに留意されたい。

【0073】

また、プロセッサの命令セットは、算術演算を実行するための1つまたは複数のタイプの算術命令も含む。本明細書で開示される実施形態によれば、これらの命令セットは、共通の重みレジスタファイルW R F 2 6 Wを利用する少なくとも1つのタイプの算術命令を含んでいてもよい。このタイプの命令は、算術命令が実行されたスレッドのそれぞれのA R F 2 6 Aの対応する算術演算の少なくとも1つの送信元を指定する少なくとも1つのオペランドを取る。しかし、算術命令の少なくとも1つの他の送信元は、共通のW R Fにあり、すべてのワーカースレッドに共通である。実施形態では、この送信元は、問題の算術命令において暗黙的である（すなわち、このタイプの算術命令に対して暗黙的である）。機械語命令の意味において、暗黙的であるということは、オペランドを指定する必要がないことを意味する。すなわち、この場合、W R Fの送信元の場合は、オペコードに固有のものである（その特定のオペコードに対して事前に決定される）。あるいは、他の実施形態では、算術命令は、W R Fの少数の異なるセットの中のどの重みレジスタのセットから重みを取り入れるかを指定するオペランドを取ることができる。しかし、重みの送信元はW R Fで見られるという事実は（例えば、汎用M R FまたはA R Fとは対照的に）、依然として暗黙的である。

【0074】

算術命令の関連タイプのオペコードに応答して、補助実行ユニット 1 8 Aの演算ユニット（例えば、F P U）は、オペコードによって指定されるように算術演算を実行し、算術演算は、スレッドのそれぞれのA R Fの指定された送信元レジスタおよびW R Fの送信元レジスタの値に基づいて演算するステップを含む。また、演算ユニットは、算術命令の送信先オペランドによって明示的に指定されるように、スレッドのそれぞれのA R Fの送信先レジスタへの算術演算の結果の出力も行う。

【0075】

共通のW R F 2 6 Wの送信元を採用することができる算術命令の例示的なタイプは、1つもしくは複数のベクトル乗算命令タイプ、1つもしくは複数の行列乗算命令タイプ、1つもしくは複数の累積ベクトル乗算命令タイプおよび／または累積行列乗算命令タイプ（命令のあるインスタンスから次のインスタンスまで乗算の結果を累計する）ならびに／あるいは1つもしくは複数の畳み込み命令タイプを含んでもよい。例えば、ベクトル乗算命令タイプは、A R F 2 6 Aからの明示的な入力ベクトルにW R Fからの既定の重みベクトルを乗じることができる。あるいは、行列乗算命令タイプは、A R Fからの明示的な入力ベクトルにW R Fからの既定の重み行列を乗じることができる。別の例として、畳み込み命令タイプは、A R Fからの入力行列にW R Fからの既定の行列を畳み込むことができる。多数のスレッドに共通の共有の重みレジスタファイルW R Fを有することにより、各スレッドは、それ自体のそれぞれのデータに共通のカーネルを乗じるまたは畳み込むことができる。これは機械学習アプリケーションにおいて、例えば、各スレッドがニューラルネットワークの異なるノードを表し、共通のカーネルが検索または訓練されている特徴（例えば、グラフィカルデータのエリアまたはボリュームのエッジまたは特定の形状）を表す場合に、多く現れるシナリオであるため、これは有益である。

【0076】

実施形態では、W R F 2 6 Wの値は、スーパーバイザスレッドによって書き込むこと

ができる。スーパーバイザ（実施形態では、すべてのスロット S 0 … S M において実行することによって開始される）は、最初に、いくつかの共通の重み値を W R F の既定の場所に書き込むために、一連のプット（put）命令を実行する。次いで、スーパーバイザは、スロット S 0 … S J - 1 のうちのいくつかまたはすべてのそれぞれのワーカーを起動するための命令を実行する（またはすべての命令を実行する）。次いで、各ワーカーは、上記で論じられるタイプの 1 つまたは複数の算術命令の 1 つまたは複数のインスタンスを含み、それにより、それらの対応する A R F 2 6 A にロードされると、それら自体の対応する入力データに対して対応する算術演算を実行する（ただし、スーパーバイザによって W R F 2 6 W に書き込まれた共通の重みを使用して）。各スレッドがそのそれぞれのタスクを終了すると、各ワーカーは、そのスロットをスーパーバイザに返すために、終了命令を実行する。起動されたすべてのスレッドがそれらの対応するタスクを終了すると、スーパーバイザは、新しい値を W R F に書き込み、新しいスレッドセットを起動することができる（または、新しいセットを起動して、W R F の既存の値の使用を続けることができる）。

10

【0077】

「メイン」、「補助」および「重み」というラベルは必ずしも制限されないことが理解されよう。実施形態では、それらのラベルは、第 1 のレジスタファイル（1 つのワーカーコンテキスト当たり）、第 2 のレジスタファイル（1 つのワーカーコンテキスト当たり）および共有された第 3 のレジスタファイル（例えば、スーパーバイザコンテキストの一部であるが、すべてのワーカーがアクセス可能である）のいずれかであってもよい。A R F

20

2 6 A および補助実行ユニット 1 8 A は、算術命令（または少なくとも浮動小数点演算）のために使用されるため、算術レジスタファイルおよび算術実行ユニットと呼ぶこともできる。M R F 2 6 M およびメイン実行ユニット 1 8 M は、それらの使用のうちの 1 つがメモリにアクセスするためのものであるため、メモリアドレスレジスタファイルおよびロード／ストアユニットと呼ぶこともできる。重みレジスタファイル（W R F）2 6 W は、間もなくさらに詳細に論じられるように、一定の 1 つまたは複数のタイプの算術命令で使用される乗法重みを保持するために使用されるため、そのように呼ばれる。例えば、これらの重みレジスタファイル（W R F）2 6 W は、ニューラルネットワークのノードの重みを表すために使用することができる。別の方法で見ると、M R F は、整数オペランドを保持するために使用されるため、整数レジスタファイルと呼ぶことができ、A R F は、浮動小数点オペランドを保持するために使用されるため、浮動小数点レジスタファイルと呼ぶことができる。2 つのバンドルの命令を実行する実施形態では、M R F は、メインパイプラインによって使用されるレジスタファイルであり、A R F は、補助パイプラインによって使用されるレジスタである。

30

【0078】

しかし、代替の実施形態では、レジスタ空間 2 6 は、必ずしもこれらの異なる目的のためにこれらの別個のレジスタファイルに分割されるとは限らないことに留意されたい。代わりに、メインおよび補助実行ユニットを通じて実行される命令は、同じ共有されたレジスタファイルの中から、レジスタを指定することができる場合がある（マルチスレッドプロセッサの場合、1 つのコンテキスト当たり 1 つのレジスタファイル）。また、パイプライン 1 3 は、命令のバンドルを同時に実行するために、必ずしも並列構成パイプライン（例えば、補助およびメインパイプライン）を含まなければならないわけではない。

40

【0079】

また、プロセッサ 4 は、メモリ 1 1 と 1 つまたは複数の他のリソース（例えば、プロセッサの他の例および／またはネットワークインタフェースもしくはネットワーク接続記憶装置（N A S）デバイスなどの外部のデバイス）との間でデータを交換するための交換インタフェース 5 1 も含んでいてもよい。図 5 に示されるように、実施形態では、プロセッサ 4 は、相互接続プロセッサタイルのアレイ 6 のうちの 1 つを形成することができ、各タイルは、より広いプログラムの一部を実行する。従って、個々のプロセッサ 4（タイル）は、より広いプロセッサまたは処理システム 6 の一部を形成する。タイル 4 は、それらの

50

それぞれの交換インタフェース 51 を介して接続される相互接続サブシステム 34 を介して、共に接続することができる。タイル 4 は、同じチップ（すなわち、ダイ）上、異なるチップ上またはそれらの組合せ（すなわち、アレイは、各々が複数のタイル 4 を含む複数のチップから形成することができる）で実装することができる。従って、相互接続システム 34 および交換インタフェース 51 は、内部（オンチップ）相互接続メカニズムおよび／または外部（インターチップ）交換メカニズムを相応に含み得る。

【0080】

マルチスレッドおよび／またはマルチタイルプロセッサまたはシステムの例示的なアプリケーションの 1 つでは、複数のスレッドおよび／またはタイル 4 にわたって実行されるプログラムは、ニューラルネットワークの訓練および／またはニューラルネットワークに基づく推論の実行を行うように構成されたアルゴリズムなどの機械知能アルゴリズムを含む。そのような実施形態では、各ワーカースレッド、各タイル上で実行されるプログラムの一部または各タイル上の各ワーカースレッドは、ニューラルネットワーク（グラフのタイプ）の異なるノード 102 を表すために使用され、それに従って、スレッド間および／またはタイル間の通信は、グラフのノード 102 間のエッジ 104 を表す。これは、図 6 に示されている。

【0081】

機械知能は、機械知能アルゴリズムが知識モデルを学習する学習段階から始まる。モデルは、相互接続ノード（すなわち、頂点）102 およびエッジ（すなわち、リンク）104 のグラフを含む。グラフの各ノード 102 は、1 つまたは複数の入力エッジと、1 つまたは複数の出力エッジとを有する。ノード 102 のいくつかの入力エッジのうちのいくつかは、ノード 102 の他のいくつかの出力エッジであり、それにより、ノードが互いに接続され、グラフが形成される。さらに、ノード 102 の 1 つまたは複数の入力エッジのうちの 1 つまたは複数のは、全体としてのグラフへの入力を形成し、ノード 102 の 1 つまたは複数の出力エッジのうちの 1 つまたは複数のは、全体としてのグラフの出力を形成する。さらに、所定のノードは、グラフへの入力、グラフからの出力および他のノードとの接続のすべてを有するものであってもよい。各エッジ 104 は、値又はテンソル（ n 次元行列）を伝達し、これらは、ノード 102 の入力エッジに提供される入力を形成するか、又は、出力エッジから提供される出力を形成する。

【0082】

各ノード 102 は、その 1 つまたは複数の入力エッジにおいて受信される 1 つまたは複数の入力についての関数を表し、この関数の結果は、1 つまたは複数の出力エッジにおいて提供される出力である。各関数は、1 つまたは複数の対応するパラメータ（重みと呼ばれる場合もあるが、必ずしも乗法重みである必要はない）によってパラメータ化される。一般に、異なるノード 102 によって表される関数は、関数の異なる形態であることおよび／または異なるパラメータによってパラメータ化することが可能である。

【0083】

さらに、各ノードの関数の 1 つまたは複数のパラメータの各々は、対応する誤差値によって特徴付けられる。さらに、対応する条件は、各ノード 102 のパラメータの誤差と関連付けることができる。単一のパラメータによってパラメータ化された関数を表すノード 102 の場合、条件は、簡単な閾値であってもよく、すなわち、条件は、誤差が指定閾値内にある場合に満たされるが、誤差が閾値を超える場合には満たされない。複数の対応するパラメータによってパラメータ化されたノード 102 の場合、誤差の許容レベルに達したそのノード 102 に対する条件は、より複雑なものであってもよい。例えば、条件は、そのノード 102 のパラメータの各々がそれぞれの閾値内に収まる場合にのみ満たされてもよい。別の例として、同じノード 102 に対する異なるパラメータの誤差を組み合わせた組合せ計量を定義することができ、条件は、組合せ計量の値が指定閾値内に収まるという条件によって満たされてもよいが、そうでなければ、条件は、組合せ計量の値が閾値を超える場合は満たされない（あるいは、計量の定義に応じて、その逆も同様である）。条件が何であれ、これにより、ノードのパラメータの誤差が許容度の一定のレベルまたは程

10

20

30

40

50

度を下回るかどうかの尺度が得られる。一般に、任意の適切な計量を使用することができる。条件または計量は、すべてのノードに対して同じものでも、異なるそれぞれのノードに対して異なるものでもよい。

【0084】

学習段階では、アルゴリズムは、経験データ（すなわち、グラフへの入力の変化する可能な組合せを表す複数のデータポイント）を受信する。多くの経験データが受信されるにつれて、アルゴリズムは、パラメータの誤差をでき得る限り最小化するように、経験データに基づいて、グラフの様々なノード102のパラメータを段階的に調節する。目標は、所定の入力に対してグラフの出力が所望の出力にできる限り近くなるようなパラメータの値を見出すことである。グラフが全体としてそのような状態に向かう傾向となったときに、グラフは収束したと言える。適切な収束度が得られた後、グラフは、予測または推論を実行するため（すなわち、何らかの所定の入力に対する結果を予測するかまたは何らかの所定の出力に対する原因を推論するため）に使用することができる。

10

【0085】

学習段階は、多くの異なる可能な形態を取ることができる。例えば、教師ありアプローチ（supervised approach）では、入力経験データは、訓練データ（すなわち、既知の出力に対応する入力）の形態を取る。各データポイントを用いて、アルゴリズムは、出力が所定の入力に対して既知の出力により近くなるよう、パラメータを調節することができる。後続の予測段階では、グラフは、入力クエリを近似予測出力にマッピングするために使用することができる（または、推論する場合は、その逆も同様である）。また、他の手法も可能である。例えば、教師なしアプローチ（unsupervised approach）では、1つの入力データ当たり1つの参照結果という概念は存在せず、代わりに、機械知能アルゴリズムは、出力データにおけるそれ自体の構造の識別を任される。あるいは、強化学習アプローチ（reinforcement approach）では、アルゴリズムは、入力経験データの各データポイントに対して少なくとも1つの可能な出力を試し、この出力が、正か否か（および、潜在的には、正または否の度合い）（例えば、勝つか負けるか、利益か損失かまたは同様のもの）が知られる。多くの試行にわたり、アルゴリズムは、正の結果をもたらす入力を予測することができるように、グラフのパラメータを徐々に調節することができる。グラフを学習するための様々な手法およびアルゴリズムは、機械学習の当業者に知られている。

20

【0086】

本明細書で開示される技法の例示的なアプリケーションによれば、各ワークスレッドは、ニューラルネットワークなどの機械知能グラフのノード102のそれぞれの個々のノードと関連付けられた演算を実行するようにプログラムされる。この場合、ノード102間のエッジ104の少なくともいくつかは、スレッド間のデータの交換に相当するものであってもよいし、タイル間の交換に関与するものもあってもよい。1つのタイル4当たり複数のスレッドを有するマルチタイル構成6の場合、各タイル4は、グラフのサブグラフを実行する。各サブグラフは、1つまたは複数のスーパーバイザスレッドを含むスーパーバイザサブプログラムと、それぞれのサブグラフのノード102を表すワークスレッドのセットとを含む。

30

【0087】

機械知能などのアプリケーションでは、プロセッサ4（例えば、タイル）のデータメモリ22におよびデータメモリ22からデータを効率的に流すことができることが望ましい。例えば、このことは、ベクトルドット積命令、行列積、累積ベクトルドット積、累積行列積または専用畳み込み命令（高度の算術複雑性を単一の算術命令に詰め込んだもの）などの複雑な算術命令のシーケンスを伴わせる上で特に（ただし、排他的にはない）有益であろう。

40

【0088】

そのような問題に対処するため、本開示は、プロセッサ4の命令セットにおいて、2つのロード操作および1つのストア操作を実行し、次いで、独立ストライドをロードおよびストアアドレスの各々に自動的に適用するタイプのロード／ストア命令を提供し、それら

50

の動作はすべて、命令のオペコードの単一のインスタンスに応答して行われる。その上、命令は、適切なレジスタファイル（例えば、MRF 26M）における効率的なアドレスパッキングを使用し、それにより、3つのアドレス（2つのロードアドレスおよび1つのストアアドレス）が2つのレジスタの空間にパッキングされる。これにより、単一の命令において、MRF 26MからLSUへの単に3つのポートのみを用いて、3つのメモリアドレスおよびストライドレジスタへのアクセス（3つのレジスタの各々へのアクセスに対してそれぞれのポート1つずつ）が可能になる。

【0089】

命令は、本明細書では、「ld2xst64pace」と呼ぶことができ、これは、2つの64ビット値をロードし、1つの64ビット値を格納するという事実を指し、「pace」は、ストライドと同義語である。しかし、このラベルまたは特定のビット幅は、必ずしも限定的なものとして受け止められない。命令は、より一般的には、単にロード／ストア命令（ただし、これは、必ずプロセッサの命令セットのロード／ストア命令の唯一の種類であるということを示唆するものでない）または「ldx2st」（ロード×2およびストア）と呼ぶことができる。その構文は以下の通りである。

```
ld2xst $aDst, $aSrc, $mAddr, $mStride, Strimm
```

【0090】

\$aDstは、補助（または算術）レジスタファイル（ARF）26Aの送信先を識別する1つまたは複数のオペランドを指し、送信先は、ARFのレジスタのうちの1つまたは複数の含む。実施形態では、ARFのレジスタの各々は、32ビット幅であり、送信先\$aDstは、4つのそのような32ビットレジスタ（\$aDst0:Dst0+3）からなり得る。ld2xst64paceの送信先オペランドは、ARFのこれらの送信先レジスタのうちの単に1つの送信先レジスタの場所（例えば、最も低い\$aDst0）を指定できるのみであり、他の送信先レジスタの場所は、これに関して暗黙的であってもよい（例えば、ARFの次の3つの隣接するレジスタ）。あるいは、他の実装形態では、複数の送信先オペランドの各々が、別個のオペランドによって明示的に独立して識別できることは除外されない（ただし、これには、命令幅の増加が必要になる）。

【0091】

\$aSrcは、ARF 26Aの送信元を識別する1つまたは複数のオペランドを指し、送信元は、ARFのレジスタのうちの1つまたは複数の含む。実施形態では、ARFのレジスタの各々は、32ビット幅であり、送信元\$aSrcは、2つのそのような32ビットレジスタ（\$aSrc0:Src+1）からなり得る。ld2xst64paceの送信元オペランドは、ARFのこれらの送信元レジスタのうちの単に1つの送信元レジスタの場所（例えば、最も低い\$aSrc0）を指定できるのみであり、他の送信元レジスタの場所は、これに関して暗黙的であり得る（例えば、ARFの次の隣接するレジスタ）。あるいは、他の実装形態では、複数の送信元オペランドの各々が、別個のオペランドによって明示的に独立して識別できることは除外されない（ただし、これには、命令幅の増加が必要になる）。

【0092】

\$mAddrは、メインレジスタファイル（MRF）26Mの2つのレジスタ（\$mAddr0:Addr0+1）の場所を指定する1つまたは複数のオペランドを指し、2つのレジスタ間では、3つのアドレス（2つのロードアドレスおよび1つのストアアドレス）が保持される。実施形態では、MRFのレジスタの各々は、32ビット幅である。メモリアドレスオペランドは、MRFのこれらのレジスタのうちの単に1つのレジスタの場所（例えば、最も低い\$mAddr0）を指定できるのみであり、他のレジスタの場所は、これに関して暗黙的であってもよい（例えば、MRFの次の隣接するレジスタ）。あるいは、他の実装形態では、メモリアドレスレジスタの各々が、別個のオペランドによって明示的に独立して識別できることは除外されない（ただし、これには、命令幅の増加が必要になる）。

10

20

30

40

50

【0093】

Strimmは、即値オペランドのセットであり、それぞれの即値オペランドは、2つのロードアドレスおよび1つのストアアドレスの各々に対するものである。MRF 26Mでは、多数のフィールドを含むストライドレジスタが提供される。各フィールドは、異なる可能なストライド値の既定のセットの異なる対応するストライド値を保持する。ストライド値は、メモリアドレスをどの程度インクリメントするかを示す値（すなわち、メモリアドレスステップ）であり、典型的には、一連のステップにおいて使用するものである。2つのロードおよび1つのストアの各々に対し、対応する即値ストライドオペランドは、命令の現在のインスタンスの対応するロードまたはストア操作を実行した後に、対応するロードまたはストアアドレスに適用するためにストライドレジスタのどのフィールドからストライド値を取り入れるかを指定する。この動作は、ロード／ストア命令の後続のインスタンスの利益のために、MRF 26Mのアドレスに沿って移動する。

10

【0094】

実施形態では、ロード／ストアユニット55は、ある特徴をサポートし、その特徴により、即値ストライドオペランドの1つの特別な値は、レジスタフィールドを指し示すというよりむしろ、1のストライド値（すなわち、使用されるアドレス空間における1アドレス単位の増加）を直接指定することができる。すなわち、即値ストライドオペランドが取りうる値のうち、1つは、1のストライドを指定し、他の値のいくつかまたはすべては、プログラム可能なストライド値を保持しているストライドレジスタの異なる可能なフィールドを指定する。ここでの1単位は、データアクセスのアトミックサイズを意味する。例えば、オペランドが16ビット浮動小数点値の4要素ベクトルである場合は、インクリメントは、1単位当たり1アトムであり、それは、8バイト（64ビット）に等しい。

20

【0095】

ストライドレジスタは、MRF 26Mのレジスタである。実施形態では、ワーカー自体は、それ自体のストライドレジスタ（それ自体のMRF）のフィールドのストライド値を書き込むことに対する責任を有する。あるいは、他の実装形態では、ストライドレジスタのフィールドのストライド値がスーパーバイザスレッドSVによって書き込めること、または、手法の組合せを使用できることは除外されない。

【0096】

オペランドによる\$mstrideの指定に対する要件は、実装形態に応じたプロセッサ4の任意の特徴である。実施形態では、ロード／ストア命令は、MRF 26Mのストライドレジスタ\$mstrideの場所を指定するオペランドを取る。従って、プログラムは、多数の可能なストライドレジスタの中からストライドレジスタを選択し、ストライドの選択におけるさらなる一層の柔軟性を提供することができる。しかし、代替の実装形態では、\$mstrideの場所を固定するかまたは暗黙的にし、ロード／ストア命令のオペランドを必要としないようにすることができることは除外されない。

30

【0097】

図7は、3つのアドレスをMRF 26Mの2つの32ビットレジスタにパッキングする例を示し、各アドレスは、21ビットである。また、図7は、3つのストライド値をMRFの1つの32ビットレジスタにパッキングする例も示し、各ストライド値は、10ビットである（この事例では、レジスタの2ビットは未使用である）。従って、ロード／ストア命令は、単一の命令において、MRFの単に2つの32ビット幅ポートを介して、3つのメモリアドレスにアクセスすることができる。第3の32ビット幅ポートは、同じ命令によって、ストライドレジスタにアクセスするために使用することができ、ストライドレジスタは、例えば、MRFの32ビット幅ストライドレジスタにパッキングされた3つの10ビット幅フィールドの3つの10ビットストライド値（アドレスデルタ）を含む（そのレジスタの2ビットは未使用のままである）。図7に示される特定のパッキングは単なる例であることに留意されたい。例えば、別の実装形態では、addr1というよりむしろ、addr2が2つのレジスタをまたぐことができる。

40

【0098】

50

図8は、MRF 26Mにおける複数のストライドレジスタ \$mstrideA\$、\$mstrideB\$、\$mstrideC\$…の構成を示す。これらはいずれも、ロード／ストア命令の対応するオペランドによって、ストライドレジスタ \$mstride\$ として指定することができる。可能なストライドレジスタの各々は、多数のフィールドを含み、例えば、実施形態では、3つのフィールドを含む（ただし、実装形態に応じて、必ずしもロードおよびストアアドレスのものと同一数とは限らないフィールドやストライドレジスタが存在する）。2つのロードアドレスおよび1つのストアアドレスの各々に対し、即値 \$strimm\$ のセットのそれぞれの即値オペランドは、それぞれのストライド値（それぞれのロードまたはストアに続いて適用されるアドレスデルタ）を取り入れるためのストライドレジスタ \$mstride\$ の可能なフィールドのうちの1つを指定することができる。例えば、実施形態では、ストライドオペランドが2つのロードおよび1つのストアの各々に対して2ビットである場合は、可能な値のうちの3つは、3つのストライドフィールドのうちの異なるストライドフィールドを指定し、他の可能な値は、レジスタに保持されている値を参照することなく、単に、1のストライドを指定するのみである。例えば、00は、1（アトム）のストライドを指定し、01は、ストライドレジスタの第1のフィールドを指定し、10は、ストライドレジスタの第2のフィールドを指定し、11は、ストライドレジスタの第3のフィールドを指定することになる。

10

【0099】

実施形態では、即値ストライドの可能な値のうちの1つは、ストライドレジスタ \$mstride\$ のフィールドというよりむしろ、1のデフォルトインクリメントを指定することに留意されたい。

20

【0100】

動作の際、実行ユニット18によって実行されると、ロード／ストア命令のオペコード（復号ステップ16によって復号された後）は、以下の動作を実行するようにLSU 55をトリガする。LSU 55は、MRF 26Mの \$mAddr\$ に保持されるアドレスによって指定されるメモリ22の2つのロードアドレスから、\$aDsc\$ によって指定されるARF 26Aの送信先に、値をロードする。さらに、LSU 55は、ARFの \$aSrc\$ から、MRFの \$mAddr\$ に保持されたアドレスによって指定されるメモリ22のストアアドレスに、値をロードする。次いで、LSU 55は、3つの即値オペランド \$strimm\$ のそれぞれの即値オペランドによって指定されるようなMRFのそれぞれのストライドレジスタ \$mstride\$ のフィールドからのそれぞれのストライド値の分だけ、2つのロードアドレスおよび1つのストアアドレスの各々を独立してポストインクリメントする。

30

【0101】

注：ロード／ストア命令の現在のインスタンスの2つのロードアドレスおよび1つのストアアドレスの各々に対し、現在の命令のストライド操作は、現在のロード／ストア操作のロードおよびストア操作の各々に続いてそれぞれ、それぞれのストライドの分だけそれぞれのアドレスをインクリメントする。実装形態に応じて、このことは、すべてのインクリメントがロードとストアの両方の後に共に適用されること（例えば、ロード、ロード、ストア、インクリメント、インクリメント、インクリメント）を意味し得る。あるいは、ストライド操作は、各ロードおよびストアの直後に、それぞれのアドレスをインクリメントすることができる（例えば、ロード、インクリメント、ロード、インクリメント、ストア、インクリメント）。実際に、ストアは、ロードのうちの一方または両方の前に来ることでもできる。重要なことは、単に、各ロードアドレスのインクリメントがそのそれぞれのロードの後に行われること、そして、ストアアドレスのインクリメントがストアの後に行われることである。肝心なことは、ロード／ストア命令の後続のインスタンスに備えて、MRFのロードおよびストアアドレスに沿って移動することである。

40

【0102】

実施形態では、メモリ11（または少なくともデータメモリ22）は、メモリからデータをロードするための単に2つの64ビットポートと、メモリにデータを格納するための

50

単に1つの64ビット幅ポートとを有するのみである。実施形態では、MRF 26M（所定のコンテキストの）は、ロード／ストアユニット55への単に3つの32ビット幅ポートを有するのみであり、ARF 26A（所定のコンテキストの）は、ロード／ストアユニット55への単に1つの64ビット幅ポートを有するのみである（注：示される例では、IALU 56は、MRF 26Mからポインタを回収し、LSU 55に渡すためにこれらのポインタからアドレスを演算するために使用され、従って、事実上、IALU 56は、LSU 55の一部として動作する。従って、この例では、MRF 26MからLSU 55への3つのポートは、MRF 26MからIALU 56への3つのポートを含む。また、他の実装形態では、LSU 55は、MRF 26Mから直接、ポインタを回収し、そのポインタに基づいてそれ自体のアドレスを演算することも除外されない）。

10

【0103】

実施形態では、ARFの4つの32ビット送信先レジスタ\$aDst0:Dst0+3（合計で128ビット）は、メモリ22から、例えば、16ビット浮動小数点（f16）値の4要素ベクトルおよび32ビット浮動小数点（f32）値の2要素ベクトルをロードするために使用することができる。ARFの2つの32ビット送信元レジスタ\$aSrc0:Src+1（合計で64ビット）は、メモリ22に、例えば、f32値の32ビットの2要素ベクトルを格納するために使用することができる。

【0104】

プログラムにおいて使用するため、ロード／ストア命令は、算術命令などの他のタイプの命令の中に散在し、算術命令は、ロード命令の送信先から入力を取り入れ、その入力に基づいて演算を実行し、ロード／ストア命令の送信元に結果を出力する。すなわち、プログラムは、ロード／ストア命令のインスタンスと、少なくとも1つの算術命令（以前に言及されたベクトルドット積命令、行列積命令、累積ベクトルドット積命令、累積行列積命令または畳み込み命令など）のインスタンスとを含み、ロード／ストア命令のうちの少なくともいくつかのロード／ストア命令の送信先は、算術命令のうちの少なくともいくつかの算術命令の送信元であり、算術命令のうちの少なくともいくつかの算術命令の送信先は、ロード／ストア命令のうちの少なくともいくつかのロード／ストア命令の送信元である。ロード／ストア命令の高い意味的密度に起因して、その2つのロードおよびストア機能により、プログラムは、ロードおよびストア操作に費やされるコードオーバーヘッドが小さな状態で、効率的にデータを処理することができる。

20

30

【0105】

例えば、プログラムは、一連の複数対の命令を含んでもよく、命令の各対は、ロード／ストア命令のインスタンスと、ロード／ストア命令のインスタンスに続くかまたは同じバンドルにおいて同時に伴われる算術命令の対応するインスタンスとからなる。命令の各対では、ロード／ストア命令の送信元は、先行する対からの算術命令の送信先として設定され、ロード／ストア命令の送信先は、現在の対または後続の対の算術命令の送信元として設定される。例えば、以下の構文を有する「arith」という算術命令を考慮する。

```
arith $aDst, $aSrcA, $aSrcB
```

構文中、\$aDstは、ARF 26Aの送信先を指定するオペランドを指し、\$aSrcA、\$aSrcBは、ARFの2つの送信元を指定するオペランドを指す（「arith」は、少なくともこの構文を有する算術命令のためにここで使用される一般名であることが理解されよう）。次いで、プログラムは、例えば、以下のような一連の複数対の命令によってプログラムすることができる。

40

...

...

```
ldx2st Xin-Pin, Pout, Tripacked, Strides; arith Pout, Xin, Pin;
```

```
ldx2st Xin-Pin, Pout, Tripacked, Strides; arith Pout, Xin, Pin;
```

```
ldx2st Xin-Pin, Pout, Tripacked, Strides; ar
```

50

```

i t h  P o u t , X i n , P i n ;
l d x 2 s t  X i n - P i n , P o u t , T r i p a c k e d , S t r i d e s ; a r
i t h  P o u t , X i n , P i n ;
. . .

```

または

```

. . .
l d x 2 s t  X i n - P i n , P o u t _ A , T r i p a c k e d , S t r i d e s ;
a r i t h  P o u t _ A , X i n , P i n ;
l d x 2 s t  X i n - P i n , P o u t _ B , T r i p a c k e d , S t r i d e s ;
a r i t h  P o u t _ B , X i n , P i n ;
l d x 2 s t  X i n - P i n , P o u t _ A , T r i p a c k e d , S t r i d e s ;
a r i t h  P o u t _ A , X i n , P i n ;
l d x 2 s t  X i n - P i n , P o u t _ B , T r i p a c k e d , S t r i d e s ;
a r i t h  P o u t _ B , X i n , P i n ;
. . .

```

など。

【0106】

実施形態では、各対は、命令バンドルであり、すなわち、下流のそれぞれのパイプラインにおいて同時に実行される。例えば、実施形態では、ロード／ストア命令は、MRF 26Mを使用してメインパイプラインによって実行され、算術命令は、ARF 26Aを使用して補助パイプラインによって並列して実行される。しかし、代替の実現形態では、単一のパイプライン処理装置を通じてロード／ストアおよび算術命令を次々と実行できることは除外されない。

【0107】

ロード／ストア命令 (l d x 2 s t) のインスタンスでは、X i n - P i n は、ARF 26Aの2つのロード操作の送信先 \$ a D s c を指し (例えば、4つの32ビットレジスタ \$ a D s c 0 : D s c 0 + 3 にロードされる $4 \times f 16$ 入力ベクトル X i n および $2 \times f 32$ 入力ベクトル P i n)、P o u t は、ARFのストアの送信元を指す (例えば、2つの32ビットレジスタ \$ a S r c 0 : S r c 0 + 1 から格納される $2 \times f 32$ 値)。「T r i p a c k e d」は、MRF 26Mのトリプルパッキングアドレスレジスタ対 \$ m A d d r 0 : A d d r 0 + 1 を指す (例えば、2つの32ビットレジスタに3つの21ビットアドレスポインタを保持する)。「S t r i d e s」は、3つの即値 S t r i m m によって指定されたMRFのストライドレジスタ \$ m S t r i d e 内のフィールドを参照することによって2つのロード操作および1つのストア操作の各々に対するストライドを指定するストライドオペランド \$ m S t r i d e および S t r i m m を指す (再び図8を参照)。算術命令のインスタンスでは、X i n、P i n は、算術命令の入力を取り入れる対象となる送信元レジスタを指し、その送信元レジスタは、現在のまたは以前の命令対のロード／ストア命令によって実行されたロードの送信先と同じになるように設定される。P o u t は、ARFの算術命令の出力の送信先を指し、その送信先は、後続の命令対のロード／ストア命令のストア操作の送信元と同じになるように設定される。2つの同時命令のバンドルとして各対を実行する実施形態では、X i n、P i n は、以前のバンドルのロード／ストア命令によって実行されたロードの送信先と同じになるように設定され、P o u t は、次のバンドルの l d x 2 s t 命令のストア操作の送信元と同じになるように設定される。この実施形態は、同じレジスタを再利用するが、2つの命令が並列して実行されるため、現在の算術命令に対する入力データは、以前のロード命令によって読み取られたものである。次いで、現在の l d x 2 s t は、次の算術命令に対する入力データを準備するために、それらの同じレジスタを再利用する。

【0108】

従って、各対では、ロード／ストア (l d x 2 s t) は、以前の命令の以前の対 (例えば、バンドル) の以前の算術命令から現在見出されている出力をレジスタ P o u t に格納

し、また、メモリからレジスタ X_{in} および P_{in} に値をロードする。同じ対（バンドル）の次の算術命令は、以前にロードされた値に基づいて算術演算を実行し、後続の対の後続のロード／ストア命令のストア操作によって格納するためにレジスタ P_{out} に出力する。

【0109】

実施形態では、算術命令は、共有された WRF_{26W} の少なくとも第3の暗黙的な送信元を有することに留意されたい。これらの送信元は、 WRF に書き込むスーパーバイザスレッドによって設定することができる。従って、スーパーバイザは、すべての実行ワーカースレッドの算術演算によって暗黙的に使用される共通の重みを設定する。

【0110】

また、 MRF_{26M} の $\$mAddr0:Addr0+1$ を指定するメモリアドレス「 $Tripacked$ 」の値は、それぞれのワーカーによっていくつかの初期値に設定される。また、 MRF の可能なストライドレジスタ $\$mStride$ の各々のフィールドのストライド値は、それぞれのワーカーによって設定される。ロード／ストア命令が実行される度、メモリアドレスレジスタ $\$mAddr0:Addr0+1$ 内の3つのアドレス（2つのロードアドレスおよび1つのストアアドレス）の各々は、それぞれのストライド値の分だけポストインクリメントされる。再び図7および8を参照すること。例えば、仮に、ロード／ストア命令が $\$mStride = \$mStrideA$ および $StrideImm = 011000$ （第1の即値ストライドオペランド = 01、第2の即値ストライドオペランド = 10 および第3の即値ストライドオペランド = 00）と指定するとする。これは、第1のロードを実行した後、 $\$mStrideA$ のフィールド0のストライド値の分だけ第1のロードアドレスがインクリメントされ、第2のロードを実行した後、 $\$mStrideA$ のフィールド1のストライド値の分だけ第2のロードアドレスがインクリメントされ、格納後、アトミックインクリメントの分だけストアアドレスがインクリメントされる（使用されているアドレス空間の1メモリアドレスステップずつインクリメントされる）ことを意味する。ロード／ストア命令の後続のインスタンスは、 $\$mStride$ の同じまたは異なる値および3つの即値オペランドの各々の同じまたは異なる値を指定するように設定することができる。従って、一連の命令の各ロード／ストア命令では、ロードおよびストアの場所は制御可能な方法で移動することができる。従って、コードは、新しいアドレスを演算するための追加の整数演算命令を必要とすることなく、入力データ空間を通じて柔軟な方法で効率的に走査することができる。

【0111】

図9は、上記の実施形態が有益であり得る例示的なアプリケーション、すなわち、入力データ X_{in} の多次元部分とカーネル K との畳み込みの実行を示す。例えば、入力データは、3Dボリュームの入力データであってもよく、カーネル K は、3Dカーネルであってもよい。また、入力データは、入力チャンネルと呼ぶこともでき、出力データは、出力特徴チャンネルと呼ぶことができる。データが多次元であると言える場合は、これは、データが二次元以上の座標系の座標の多数の組合せの各々における値を取ることを意味し、各次元（すなわち、各軸）は、異なる独立変数を表す（そして、座標の所定の組合せにおけるデータの値は、従属変数である）。

【0112】

この例は、各データ値がピクセルを表す画像処理におけるものである。例えば、典型的には、赤、緑および青の各々に対して8ビットが使用され、次いで、それらの8ビット整数値は、畳み込み演算の前に $f16$ 値に変換される。各 R 、 G 、 B 値は、別個の入力平面（すなわち、別個の入力チャンネル）として扱われる。従って、各2D入力画像は、 x 、 y 、 z 次元を有し、 R 、 G 、 B 平面は、 z 軸に存在する。データのボリュームは、例えば、フレームのシーケンスを含んでもよく、ボリュームの2つの次元は、フレームエリアの x 、 y 空間軸を表し、第3の次元は、シーケンスの異なるフレームに対応する時間軸を表す。別の例では、ボリュームの3つの次元は、空間または物体の3Dモデルまたは画像の3つの x 、 y 、 z 空間次元を表してもよい。別の例では、データは、静止画像の x 、 y 空

間軸を表す単なる二次元であってもよく、z 軸は、同時に処理される入力画像の数を表す。開示される技法の適用性は、画像またはグラフィックス処理に限定されず、より高い次元のデータも除外されないことに留意されたい。例えば、二次元、三次元またはそれ以上の次元は、ユーザ挙動などを学習するアプリケーションにおいてユーザまたはそれらの状況の異なる態様を表してもよい。

【0113】

カーネルKは、既定の重みの行列である。カーネルKは、ニューラルネットワークが認識のために訓練されているかまたは後続の推論において検索されているある特徴を表してもよい。例えば、画像データの場合、カーネルは、エッジまたは他の何らかの特定の種類の形状を表してもよい。機械学習アプリケーションでは、ニューラルネットワークの各ノードは、それぞれのカーネルに相当してもよい（ただし、後にさらに論じられるように、多くのニューラルネットワークでは、同じカーネル値を使用するが、単に異なる接続104を有する複数のノード102が存在する）。実施形態では、各ノード102は、それぞれのワークスレッドによって表され、従って、各スレッドは、そのスレッドのそれぞれの入力データとカーネルKとのそれぞれの畳み込みを実行する。ワークスレッドの少なくともいくつかは、共有の重みレジスタファイル（WRF）26Wにその値が格納されている同じカーネルKを使用することができるが、そのようなスレッドの各々は、それ自体のそれぞれの補助（または算術）レジスタファイル（ARF）26Aを介してメモリ22からおよびメモリ22に流れるスレッド自体のそれぞれの入力データにそのカーネルを畳み込む。

【0114】

三次元の離散畳み込み（*）は、以下のように表すことができる。

$$X_{in} * K[x, y, z] = \sum_{x', y', z'} (K[x', y', z'] X_{in}[x - x', y - y', z - z'])$$

【0115】

従って、畳み込みは、入力データにカーネルを重ね合わせることができる可能な位置のいくつかまたはすべてにわたってカーネルKをシステマチックに走査することを含む。そのような位置の各々では、カーネルと重なり合う入力データの部分の各ピクセル（またはデータポイント）と、対応するオーバーレイポイントにおけるカーネルの値とは乗算され、これらの個々の乗算の各々の結果は加算され、その特定のオーバーレイ位置における入力データとカーネルとの畳み込みを表すスカラ出力が得られる。次いで、この動作は、出力データ（1つのカーネル位置当たり1つのスカラ値）の対応するボリュームを得るために、入力データX_{in}に対するカーネルの異なる走査位置で繰り返される。例えば、図9に示される例では、カーネルKを入力データX_{in}のボリューム内の一隅に置いて開始することができ、各カーネル値とそのカーネルがその位置にあるときにそのカーネルが重なり合うデータポイントとが乗じられる。次いで、カーネルはボリュームの長さに沿って1ステップ移動し、この新しい位置におけるカーネルを用いてポイント乗算および加算が再び実行されるなど、以下同様である。ボリュームの全長に沿ってカーネルが走査されると、カーネルは新しい座標位置にシフトされ、全ボリュームが操作されるまで、長さ方向走査が再び実行されるなど、以下同様である。所定の各位置におけるカーネルは、出力データの1つのピクセル（1つのスカラ値）を生成する。従って、出力データX_{out}は、畳み込み走査のステップサイズに応じて、入力データと同じまたは同様のサイズを有することになる。

【0116】

実際には、この動作は、より小さな動作に分割する必要がある。このことは、所定の位置におけるデータとカーネルとの畳み込みを多くの個々のベクトルドット積および部分積に分けることによって行うことができる。このことは、図9の中央および右側に示されている。仮に、例えば、カーネルのサイズが3×3×16ピクセルであるとする（「ピクセル」は、本論考のために、必ずしも画像を指すとは限らない）。すなわち、カーネルは、3×3×16の既定の重み値からなる。このカーネルは、9つの16要素ベクトルCW（

本明細書では、カーネルKの構成カーネルとも呼ばれる)に分割することができる。最初に、これらの9つのベクトルのうちの1つと対応する位置における入力データとの間でドット積(すなわち、スカラ積)が取られ、部分積P(スカラ)が得られる。次いで、9つのベクトルのうちの別のものとの間でドット積が取られ、第1の部分積と累計して、第2の部分積(この場合も、スカラ)が得られる。次いで、9つのベクトルのうちの別のものとの間でドット積が取られ、この結果を第2の部分積と累計して、第3の部分積が得られるなど、以下同様である。この動作が終了し、9つのベクトルのすべてを累計した時点で、総累計結果は、出力データXoutにおける単一のポイントまたはピクセルの値を与える。次いで、この動作は、入力データXinのボリュームに対する異なる位置にわたってカーネルKが走査されるにつれて、各カーネル位置で繰り返され、従って、出力データXout(1つのカーネル位置当たり1つの出力データポイント)の対応するボリュームが得られる。

10

【0117】

実施形態では、畳み込みなどの演算を効率的な方法で実行することを支援するため、プロセッサ4の命令セットは、累積積(「amp」)命令の形態の算術命令を含む。その命令は、以下の構文を有する。

amp \$aDst, \$aSrcA, \$aSrcB, Phase

構文中、\$aDstは、この場合もやはり、ARF 26Aの送信先を指定するオペランドを指し、\$aSrcA、\$aSrcBは、ARFの2つの送信元を指定するオペランドを指す。「Phase」は、累積の位相を指定する即値オペランドである。例えば、実施形態では、位相オペランドは、4つの可能な位相0・・・3のうちの1つを指定する2ビットで形成される。実施形態では、amp命令は、f16v4sisoampと呼ぶことができ、これは、第1の入力として4つの半精度(16ビット)浮動小数点値の1つのベクトルを取り入れ、第2の入力として1つの単精度(32ビット)浮動小数点値を取り入れ、1つの単精度(32ビット)浮動小数点値を出力するという事実を指す(「siso」は、「単精度入力、単精度出力」を指す)。実際に、実施形態では、第2の入力は、2つの単精度浮動小数点値(2×32ビット)の対であり、出力も同様である。しかし、これらの特定の精度の値および要素の数は、すべての可能な実施形態に制限されないことが理解されよう。

20

【0118】

30

ampなどの1つまたは複数のタイプの命令に対し、実施形態では、WRFの重みの場所は、完全に暗黙的である。その代替としてまたはそれに加えて、1つまたは複数の他のタイプの命令は、WRFの少数の異なるセットの中からどの重みのセットを使用するかを指定する追加のオペランド(図示せず)を取ることができる。例えば、その例は、後に言及されるスリムな畳み込み(slim convolution)(「slic」)命令であり得る。

【0119】

以前に説明されたロード/ストア命令(ld2xst)と組み合わせると、amp命令などの命令は、メモリからデータを効率的に流し、ドット積および部分積を実行し、部分積をメモリに戻すために使用することができる。例えば、以下のような4つの命令対のループを経るシーケンスを含むプログラムを考慮する。

40

```
Loop {
ldx2st Xin-Pin, Pout, Tripacked, Strides; am
p Pout, Xin, Pin, Phase=0;
ldx2st Xin-Pin, Pout, Tripacked, Strides; am
p Pout, Xin, Pin, Phase=1;
ldx2st Xin-Pin, Pout, Tripacked, Strides; am
p Pout, Xin, Pin, Phase=2;
ldx2st Xin-Pin, Pout, Tripacked, Strides; am
p Pout, Xin, Pin, Phase=3;
```

50

}

【0120】

別の例示的なバージョンは、以下の通りである。

```
Loop {
ldx2st Xin-Pin, Pout_A, Tripacked, Strides;
amp Pout_A, Xin, Pin, Phase=0;
ldx2st Xin-Pin, Pout_B, Tripacked, Strides;
amp Pout_B, Xin, Pin, Phase=1;
ldx2st Xin-Pin, Pout_A, Tripacked, Strides;
amp Pout_A, Xin, Pin, Phase=2;
ldx2st Xin-Pin, Pout_B, Tripacked, Strides;
amp Pout_B, Xin, Pin, Phase=3;
}
```

10

など。

【0121】

この場合もやはり、実施形態では、各対は、命令バンドルであり、すなわち、下流のそれぞれのパイプライン（例えば、メインおよび補助パイプライン）において同時に実行される。

【0122】

命令の各対は、ロード／ストア命令のインスタンスおよびそれに続く累積積（amp）命令のインスタンスを含む。amp命令の送信元は、同じまたは先行する命令対のロード／ストア命令によって実行された2つのロードの送信先である。2つの同時命令のバンドルとして各対を実行する実施形態では、ampの送信元（Xin、Pin）は、以前のバンドルにおいて実行されたロードの送信先と同じになるように設定される。amp命令の送信先は、後続の命令対（例えば、次のバンドル）においてロード／ストア命令によって実行される1つのストアの送信元である。amp命令の送信元\$aSrcA、\$aSrcBのうちの一方は、入力データXinから入力ベクトルxを取り入れるために使用される。他方は、部分和を取り入れるために使用される。amp命令は、実行されると、その入力ベクトルxと重みレジスタファイルWRF 26Wからの重みCWの対応するベクトル（このベクトルCWは、全体または3DカーネルKの構成カーネルである）とのドット積を実行する。位相オペランドは、ドット積の結果を累計するための位相を指定する。ループの所定のインスタンスにおけるシーケンスの命令の各対では、位相オペランドは、シーケンスの異なる連続位相を指定する異なるそれぞれの値に設定される。位相オペランドによって指定されるようなシーケンスのこれらの連続位相にわたり、amp命令の効果は、連続ドット積の各々の結果を累計することである。入力部分和との累計は、シーケンスの第1の位相において始まる。

20

30

【0123】

ロード／ストア命令のストライド機能により、プログラムは、MRF 26Mに含めるための新しいメモリアドレスを演算するための別個の整数演算命令を必要とすることなく、メモリアドレスに沿って、命令の各対を有する次のデータ片に自動的にシフトすることができる。ストライドレジスタ\$mStrideから多くの事前にプログラムされたストライドのうちのいずれかを選択できるとの上記ロード／ストア命令の能力は、図9において例示として示されるものなどの多次元データを取り扱う上で特に有益である。メモリ空間は、一次元であるが、データは、二次元、三次元またはそれ以上の次元を有し得る。多次元データ空間への1Dメモリ空間のマッピングに応じて、カーネルKが入力データXinを通じて走査されるにつれて、メモリ空間の対応するメモリアドレスを通じて異なるサイズのステップを取る必要がある場合がある。例えば、カーネルKが入力データXinの一次元に沿って長さ方向に走査されるにつれて、各走査位置におけるデータのメモリアドレスは、アトミックステップでインクリメントするが、垂直平面において走査をシフトする必要がある際は、入力データXinの次のデータ片またはサブセットに到達するために

40

50

、メモリアドレスの異なるサイズのステップが必要とされる場合がある（そして、独立して、第2の入力および出力アドレスに対して同じプロセスが行われる）。1つのストライドレジスタ $\$mstride$ または複数のストライドレジスタ（例えば、 $\$mstrideA$ 、 $\$mstrideB$ 、・・・）の異なるフィールドは、有利には、メモリ空間の異なるサイズのジャンプを事前にプログラムすることができる。次いで、プログラムまたはコンパイラは、ロード／ストア命令別にどのジャンプサイズを使用するかを選択するために、即値オペランド $Strimm$ を設定することができる（次のアドレスまたは次のステップサイズを再計算するための別個の整数演算命令を必要とすることなく）。

【0124】

実施形態では、 amp 命令は、累積行列積命令である。この命令の動作については、図10～12を参照して説明する。図10に示されるように、 amp 命令は、FPUによって実行されると、入力データ Xin からの N 要素入力ベクトルと重みの $M \times N$ 行列（ WRF_{26W} から）との乗算を実行する。実施形態では、 $M=8$ および $N=16$ である。行列の M 個の行の各々は、異なる対応するカーネル $K_0, K_1 \dots K_{M-1}$ からの構成ベクトルに相当することに留意されたい。従って、 amp 命令は、入力データと M 個の異なるカーネル K との並列畳み込みのコンポーネントを実行するために使用される。例えば、これらのカーネルの各々は、入力データと異なるそれぞれの特徴（例えば、異なるエッジまたは形状）との畳み込みに相当してもよい。畳み込みニューラルネットワークの場合、これらのカーネルは、異なる特徴フィルタであり、 M 個のカーネルの各々は、入力層における異なる特徴の存在の検出を試みる。所定のカーネル K に対し、 amp 命令は、1つのドット積および累積のみを実行する。しかし、行列は、データと複数のカーネルとの効率的な並列畳み込みを行えるようにする。

【0125】

N 要素入力ベクトルは、 N_1 個のセグメントに分けられ、その各々は、 N_2 個の要素のサブベクトルを含む（従って、 $N=N_1 \times N_2$ ）。実施形態では、 $N_1=4$ および $N_2=4$ である。 amp 命令の位相オペランドは、 N_1 個の異なるセグメントに対応する N_1 個の異なる可能な値 $0 \dots N_1-1$ のうちの1つを取り入れる。ループを経るシーケンスの各インスタンスでは、各連続対の amp 命令は、位相 $0 \dots N_1-1$ のうちの異なる連続位相を指定する位相オペランドの異なる値を取る。各位相は、 N 要素入力ベクトルの N_1 個のサブベクトルのうちの対応するものと行列の M 個の行の各々とのドット積（従って、 M 個の異なるカーネル K の各々に対する演算の一部）を実行する。実行ユニット18（実施形態では、補助または算術実行ユニット18A）のFPUは、 M 個のアキュムレータ状態 $\$AACC$ （1スレッド当たり）を有する。実施形態では、これらのアキュムレータ状態 $\$AACC$ は、FPUの内部状態57として実装される。しかし、代替の実装形態では、これらのアキュムレータ状態 $\$AACC$ を複数のレジスタファイル26（例えば、それぞれのスレッドの ARF_{26A} ）のうちの1つのレジスタファイル26のレジスタとして実装できることは除外されない。示される例では、これらのアキュムレータ状態 $\$AACC$ は、偶数番号が付けられた状態 $\$AACC[0]$ 、 $\$AACC[2] \dots \$AACC[14]$ であり、奇数番号が付けられた状態 $\$AACC[1]$ 、 $\$AACC[3] \dots \$AACC[15]$ は、シーケンスの異なるループ間で値を伝播するために使用することができるが、これは、実装形態の詳細であり、これに限定されない。

【0126】

図11は、 amp 命令の動作を図に示す。図12は、一連の amp 命令の例を与える。各々は、上記で示されるように、命令対の対応するロード／ストア命令が先行すると理解されている。図12では、 P_n は、単精度入力部分 n であり、 x_n は、 $f16v4$ 入力ベクトルであり、 $CW_{m,n}$ は、共通の重み状態 $\$CWEI_m_n$ であり、 R_n は、 P_n で始まる連続ドット積累積の最終的な単精度結果である。

【0127】

動作の際、 amp 命令のオペコード（復号済みの）に応答して、実行ユニット18（実施形態では、補助実行ユニット18A）のFPUは、以下の動作を実行する。

・ `amp` 命令の送信元オペランドのうちの1つによって指定された `ARF 26 A` のレジスタから、 M/N 1 個の数の部分和を取り入れ、これらの部分和を次のループに対するプロパゲータ状態に一時的に置く。このプロパゲータ状態は、`FPU` の内部状態 57 で実装することができるか、または、代替の実装形態では、レジスタファイル 26（例えば、`ARF`）のうちの1つのレジスタファイル 26 の別のレジスタであり得る。各位相は、 M/N 1 個の部分和の異なるサブセットを取り入れる。 N 1 個の位相にわたり、 M 個のそのような部分和は、プロパゲータ状態にされる。従って、 $M=8$ および $N=4$ の場合は、各 `amp` 命令は、入力として2つの部分和（例えば、2つの `f32` 値）を取り入れ、これらをプロパゲータ状態に保持し、すべての4つの位相にわたり、8つの部分和が受信される。

10

【0128】

・ `amp` 命令の送信元オペランドのうちの1つによって指定された `ARF 26 A` のレジスタから、入力ベクトル `Xin` の N 2 要素セグメント（サブベクトル）を取り入れる。行列の M 個の行の各々（従って、 M 個のカーネル `K0` … `KM-1` の各々）に対し、この入力ベクトル `Xin` の N 2 要素セグメントと `WRF` からの重み `CW` の対応するサブベクトルとのドット積を実行する。どの列から重みのサブベクトルが取り入れられるかは、位相オペランドの値に依存する。さらに、位相がシーケンスの第1の（すなわち、初期の）位相である場合は、部分和のうちの対応するものが M 個のドット積の各々に加えられる。以前のループの各 `amp` 命令はそれぞれの M/N 1 個（ N 1 は、位相の数）の部分和を取り入れたため、以前のループは、現在のループに備えてすべての M 個の部分 and をプロパゲータ状態に置いたことになることに留意されたい。

20

【0129】

・ M 個の行の各々に対し、上記の計算の結果は、 M 個のアクキュムレータ状態 `$AAC` のうちの対応する状態に既にある任意の値と累計され（総和が求められ）、それによって、 M 個の結果 `R0` … `RM-1` のうちの対応するものを生成する。従って、 N 1 個の位相にわたり、`amp` 命令は、完全な N 要素入力ベクトルと `WRF` からの重みの対応する N 要素ベクトルとのドット積を実行する。

【0130】

・ 現在のループでは、以前のループからの M/N 1 個の数の結果 R は、`amp` 命令の送信先オペランドによって指定された `ARF` の送信先レジスタに出力される。各位相は、 M/N 1 個の部分 and の異なるサブセットを出力し、 N 1 個の位相にわたり、 M 個のそのような部分 and がプロパゲータ状態にされる。従って、 $M=8$ および $N=4$ の実施形態では、1つの `amp` 命令当たり2つの結果（例えば、2つの `f32`）が出力され、すべての4つの位相にわたってすべての8つの結果 `R0` … `R7` が出力される。

30

【0131】

・ 現在のループの最後の位相の後（または結果が上書きされる前の次のループの第1の位相の開始時）、現在のループの結果 R は、次のループの出力に備えて、プロパゲータ状態に一時的に置かれる。このプロパゲータ状態は、`FPU` の内部状態 57 で実装することができるか、または、代替の実装形態では、レジスタファイル 26（例えば、`ARF`）のうちの1つのレジスタファイル 26 の別のレジスタであってもよい。

40

【0132】

要約すると、単一の部分 and は、（1）第1のドット積結果に加えられ、アクキュムレータ状態（以前に保持された値を上書きする）で格納され、（2）さらなる3つのドット積結果と累計され、（3）一時的なプロパゲータ状態にコピーされる（その場合、その値は、今後の `amp` 命令の送信先オペランドに書き込まれる）。ステップ（3）の代替手段は、アクキュムレータからそのまま得られた結果として生じた部分 and を `amp` 命令の送信先オペランドに書き込むことである。いずれの方法にせよ、ステップ（1）および（3）は、ハードウェアにおいて重複してもよく、4サイクルループを提供する。

【0133】

実施形態では、アクキュムレータ状態は、明示的にリセットする必要はない。各行列ベク

50

トル乗算の出発点はメモリから読み取られた入力部分和であるため、出力ボリュームは、畳み込みの開始時に、アキュムレータ状態をリセットするというよりむしろ、すべての要素がゼロに設定される。そうは言っても、実施形態では、タイル4は、実際に、1から0にアキュムレータ状態のすべてを初期化することができる。

【0134】

現在のamp命令の入力プロパゲータ状態は、次のループでの使用のために保持されることに留意されたい。また、amp命令によってARFの送信先レジスタに出力される結果は、現在のampの累積演算が適用される前の既存のアキュムレータ状態である（すなわち、累積の以前の短いシーケンスの結果であり、現在の入力を含まない）。

【0135】

プログラムは、結果Rが後続のループに対する部分和入力Pになるように構成される（プログラマまたはコンパイラによって）。その正確な関係は、プログラマまたはコンパイラにかかっており、データおよびカーネルの構造に依存する。典型的には、部分和は、0の初期値（メモリ内の）を有する。

【0136】

ループ間の時間差により、シーケンスは、ウォームアップ期間を必要とすることに留意されたい。提示されている特定の入力ベクトルとamp命令によって返されている4位相計算の結果との間には待ち時間がある。その待ち時間は固定されており、これは、最初のM個の結果が無効であるか、関心のないものであるかまたは意味のないものであり、ウォームアップコードにおいて処分されることを意味する。有益な結果が生み出されないこのウォームアップ期間の後、コードは繰り返しループに入り、その繰り返しループは有意な結果を生み出し、その有意な結果は必要なメモリ場所に格納される。また、内側のループの後にはクールダウン期間もあり、クールダウン期間では、新しい入力は提供されないが、最終的な出力値が格納される。図12は、f16v4sisoampに対するウォームアップ期間および「結果待ち時間」を示す。

【0137】

実施形態では、カーネルの数は $M=8$ であり、入力ベクトルの要素の数は $N=16$ であり、位相の数は $N1=4$ である。実施形態では、入力ベクトルの各要素は、f16値であり、各入力部分和および各結果は、f32値である。そのような実施形態では、上記のプロセスは、1つの命令対当たり2つのf32部分和および入力ベクトルの4つのf16要素のレートで、データをロードおよび処理することができ、1つの命令対当たり2つのf32値のレートで、データを出力および格納することができる。それは、1つの命令対当たり128ビットの入力および64ビットの出力である。これは、LSU55とデータメモリ22との間の2つの64ビット幅ロードポートおよび1つの64ビット幅ストアポートに相当する（再び図4を参照）。また、アドレスのトリプルバッキングのため、プロセスは、MRF26MからLSUへの2つの32ビット幅ポートおよびストライドに対する別のポートを介して処理することができる。2つの64ビット幅ポートは、ARF26Aから補助実行ユニット18AのFPUへと使用される。

【0138】

重みCWは、共有されたWRF26Wから取り入れられ、各スレッドに対して同じである。ニューラルネットワークなどのアプリケーションでは、異なるデータと重みの同じセットとを乗じる（畳み込みの一部など）必要がある多くのシナリオがあるということが分かる。例えば、ニューラルネットワークのいくつかのノードは、全く同じ重みを含むが、異なる接続を含む場合がある。例えば、各カーネルK（ $m=0 \dots M$ ）は、入力データ（例えば、グラフィック画素のエリアまたはボリューム）と特徴（例えば、エッジまたは形状）との畳み込みを表すことができる。従って、これらのカーネルに対応するワークスレッドには、別個の重みは不要である。代わりに、スーパーバイザが所有する重み状態の単に1つのコピーが提供されるのみである。従って、共有の重みレジスタファイルからの共有されたスレッドオペランドの使用により、有利には、重みに対するレジスタ空間が少なく済む。図12は、各16要素入力ベクトルに対しループが8個の出力値（1つ

10

20

30

40

50

のカーネル当たり1つの出力値)を生成することを示す。この特定のシナリオでは、それらの出力値は、各々が32ビットである。上記のループは、4つのサイクルごとに16個の入力値(命令ごとに4×f16の入力値)を処理する。従って、その入力レートを維持するため、プロセッサ4は、4サイクルループごとに8×32ビット値を生成し、それは、1つのループ当たり4×64ビットストアまたは1つのサイクル当たり1×64ビットストアを意味する。

【0139】

実施形態では、奇数アキュムレータ\$AACC[1]、\$AACC[3]・・・\$AACC[15]は、結果(Rx)をずらすために使用され、部分和入力(Px)にも同じことをするために使用される。部分入力は、一度に2つずつアキュムレータレジスタ\$AACC[13]および\$AACC[15]に供給される。各サイクルでは、それらの値は、\$AACC[11]および\$AACC[9]へと前方に進み、以前に\$AACC[11]および\$AACC[9]内にあった値は、\$AACC[7]および\$AACC[5]に進むなど、以下同様である。それらの値がそれらの送信先に到達すると、それらの値は、累積演算に備えて、偶数番号が付けられたアキュムレータに伝播される。

【0140】

図13および14は、本明細書で説明されるタイプのロード/ストア命令(すなわち、それぞれのスレッドのARF 26Aを介して流された入力データに共通の重みレジスタファイル(WRF) 26Wからの重みの2Dセットを畳み込む畳み込み命令)から恩恵を得ることができる別のタイプの算術命令を示す。実施形態では、この命令は、半精度浮動小数点ベクトルのスリムな畳み込み(slic命令)の形態を取ることができ、例えば、「f16v4slic」は、f16値の4要素サブベクトルに基づいて演算するという事実を指す。この命令は、1つの半精度および1つの単精度入力を有し、単精度出力を有する。slicはampと同様であり、ampと同じハードウェアを使用する。しかし、この命令は、8つのカーネルの各々に対する1×1×16累積ドット積というよりむしろ、2つのカーネルの各々に対する1×4×4畳み込み(真の畳み込みである)を実行する。実際に、実施形態では、この命令は、1×N×4畳み込み(Nは、2、3または4である)を実行するように構成することができる。

【0141】

累積行列積(「amp」)およびスリムな畳み込み(「slic」)命令は、ワーカコンテキストにわたる重み共有が適切なシナリオに対して、高性能の乗累算シーケンスを容易にする。実施形態では、ampおよびslic命令は、単精度数フォーマットと半精度数フォーマットの両方をサポートし、同じ基本的な方法で動作することができる。共通の演算構成状態(WRF 26Wの共有重みを含む)は、最初に、スーパーバイザコンテキストによって初期化される。次いで、入力データの2つのストリーム、すなわち、入力起動(ピクセル)データおよび後続の乗累算シーケンスに対する初期値を指定する部分和价值(畳み込みの場合、これらは、部分的に演算された起動前のものである)が処理される。出力データ(結果として生じた累計済み部分和价值)の単一のストリームが生成される。各部分和价值は、最終的な部分和价值が出力として示される前に、固定長シーケンスの乗累算の対象となる。多くのドット積および累算は、演算エンジンによって実行され、並列して発生する。第1の被乗数は、入力データストリームによって提供され、第2の被乗数は、共通の演算構成状態によって提供される。

【0142】

以下の表は、いくつかの例示的なampおよびslic命令の変形形態をリストし、実施形態では、それらのいずれかまたはすべては、プロセッサの命令セットに含めることができる。

【0143】

【表 1】

命令	入力データフォーマット	入力部分和 フォーマット	出力部分和 フォーマット
f16v4sisoamp	半精度 (v4)	単精度 (v2)	単精度 (v2)
f16v4hihoamp	半精度 (v4)	半精度 (v2)	半精度 (v2)
f16v4sihoamp	半精度 (v4)	単精度 (v2)	半精度 (v2)
f32sisoamp	単精度 (スカラ)	単精度 (v2)	単精度 (v2)
f16v4sisoslic	半精度 (v4)	単精度 (v2)	単精度 (v2)
f16v4hihoslic	半精度 (v4)	半精度 (v2)	半精度 (v2)
f16v4sihoslic	半精度 (v4)	単精度 (v2)	半精度 (v2)
f32sisoslic	単精度 (スカラ)	単精度 (v2)	単精度 (v2)

10

注:いくつかのタイプの命令では、重みの場所は完全に暗黙的であるが、他のタイプでは、多くの異なるセットのうちのどれをW R Fにおいて使用するかを選択するオペランドを取る。f16v4sisoslic命令は、例えば、f16v4sisoampの重み状態の4分の1のみを必要とする。その結果、ソフトウェアは、最大で4セットの重みをプリロードすることができる。重みセット選択は、即値オペランドの2ビットによって指定される。他方では、f16v4sisoampの場合は、完全な重み状態を取り入れ、従って、その中での選択はない。

20

【0 1 4 4】

上記の実施形態は単なる例示として説明されていることが理解されよう。

【0 1 4 5】

例えば、本開示の範囲は、上記で説明されるアーキテクチャであり、そのアーキテクチャでは、スーパーバイザスレッドに対して別個のコンテキストが提供されるか、または、スーパーバイザスレッドは、スロットで実行し、次いで、そのスロットをワーカーに委ねる。スーパーバイザは、代わりに、汎用コンテキストを使用することができる。あるいは、別の構成では、例えば、スーパーバイザは、それ自体の専用スロットで動作できる。さらに、実装形態は、スーパーバイザの役割さえも有するスレッドのうちの特定のものに限定されず、実際に、マルチスレッドには全く限定されない。本明細書で開示される技法は、非マルチスレッド動作を採用しないプロセッサ上で使用することもできる。さらに、本開示の範囲は、タイルのレイにおけるタイルであるプロセッサ4に限定されない。代替の実施形態では、プロセッサ4は、例えば、スタンドアロンプロセッサでも、単一チッププロセッサであってもよい。

30

【0 1 4 6】

実行ユニット18の実装形態は、別個のメイン実行ユニット18 Mおよび補助実行ユニット18 Aにも、別個のM R FおよびA R Fにも限定されない。また、別個のW R Fの使用にも限定されない。一般に、レジスタは、1つまたは複数のレジスタファイルのいずれからのものであってもよく、異なるメモリアクセスと算術演算との間で共有することも、分離することもできる。

40

【0 1 4 7】

開示されるロード／ストア命令と共に使用することができる算術命令は、行列積または畳み込み命令に限定されない。より一般には、ロード／ストア命令のインスタンスは、少なくとも2つの入力オペランドを取り、かつ、少なくとも1つの結果を出力する任意のタイプの算術命令または命令の組合せに対して、メモリからデータを流すためおよびメモリ

50

に結果を戻すために使用することもできる。例えば、ロード／ストア命令タイプは、別個の命令または汎用コードにおいて累積が行われる非累積行列積命令と共に使用することも、以前の積の結果が必ずしも後続の積の入力であるとは限らない行列乗算（畳み込み以外）の他のシーケンスと共に使用することもできる。また、ロード／ストアは、ベクトルドット積または累積ベクトルドット積命令（すなわち、M個の異なるカーネルを並列して適用する必要がない算術命令）と共に使用することもできる。

【0148】

さらに、本開示の範囲は、例示として上記で開示される、特定のレジスタサイズ、ポートビット幅、ポートの数、値精度、ベクトルサイズまたは行列サイズに限定されない。他のレジスタおよびポートビット幅が可能であり、他の値の精度および他のベクトルまたは行列サイズ（例えば、要素の数の観点から）も同様に可能である。さらに、他の実装形態では、2つのレジスタへの2つのロードアドレスおよび1つのストアアドレスの他のパッキング（例えば、2つの16ビットレジスタへの3つの10ビットアドレス、または、2つの64ビットレジスタへの3つの42ビットアドレス）が可能である。また、ロードおよびストアアドレスは、必ずしも同じ長さである必要はない。さらに、代替の実装形態では、ストライドレジスタへのストライド値の他のパッキング（例えば、32ビットレジスタへの4つの8ビットストライドフィールドなど）も可能である。

【0149】

さらに、さらなる変形形態では、本開示の適用性は、画像処理に限定されない。「カーネル」という用語は、本明細書で使用される場合は、ベクトル乗算、行列乗算、畳み込みまたは他のもの（例えば、相関）などの演算を含む任意のプロセスの一部として適用されるいかなる重みの行列も意味してもよい。また、アプリケーション能力は、3Dボリュームのデータに限定されない。入力データ（およびカーネル）は、線形でも、2Dでも、より高い多次元性（3より大きい独立変数または自由度）を有するものでもあってもよい。さらに、本開示の範囲は、機械学習アプリケーションに限定されない。ベクトル乗算、行列乗算および／または畳み込みなどを含むものなどのプロセスを実行することが望ましい他の多くのアプリケーションが存在する。

【0150】

開示される技法の他の変形形態または使用事例は、本明細書の本開示が与えられた時点で当業者に明らかになるであろう。本開示の範囲は、説明される実施形態による制限は受

【符号の説明】

【0151】

- 4 プロセッサ
- 6 タイルのアレイ
- 10 マルチスレッド処理装置
- 11 ローカルメモリ
- 12 命令メモリ
- 13 実行パイプライン
- 14 フェッチ
- 16 復号
- 18 実行ユニット
- 18A 補助実行ユニット
- 18M メイン実行ユニット
- 22 データメモリ
- 24 スケジューラ
- 26 コンテキストレジスタ
- 26A 補助レジスタファイル
- 26CSR 制御状態レジスタ
- 26M メインレジスタファイル

10

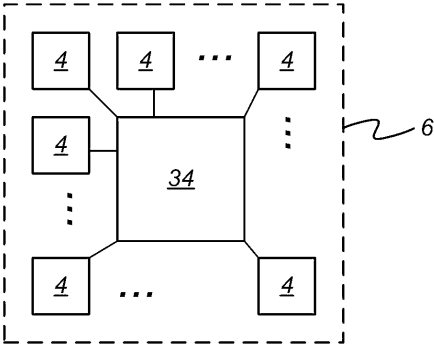
20

30

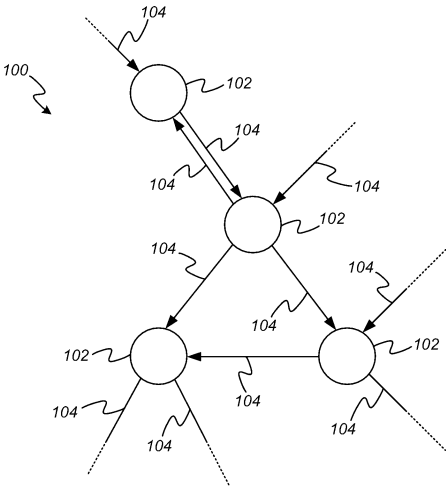
40

50

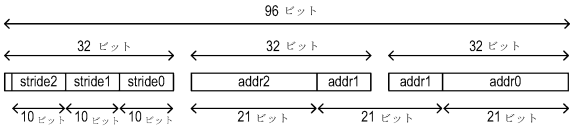
【図 5】



【図 6】



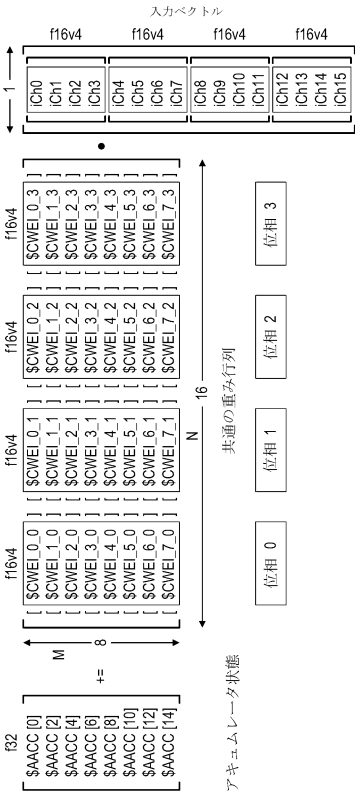
【図 7】



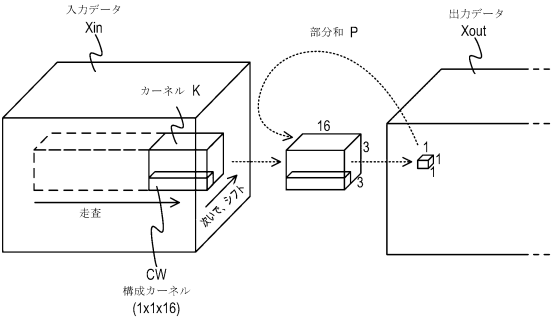
【図 8】

\$mStrideA	strideA2	strideA1	strideA0
\$mStrideB	strideB2	strideB1	strideB0
\$mStrideC	strideC2	strideC1	strideC0
⋮	⋮	⋮	⋮

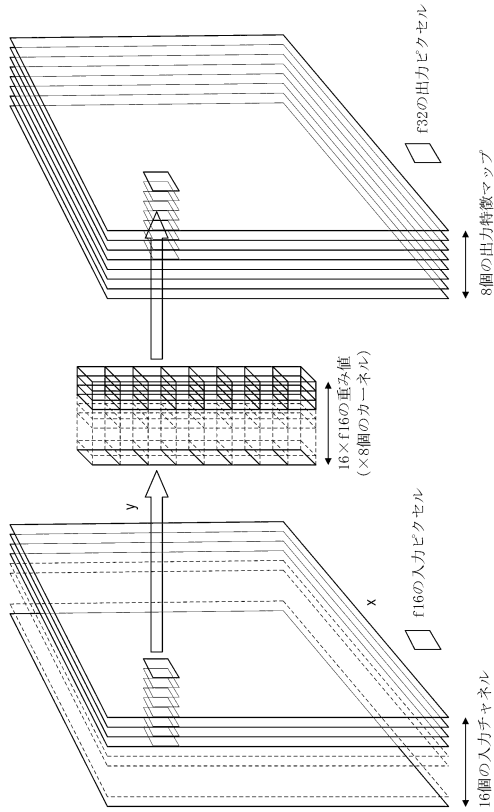
【図 10】



【図 9】



【図 1 1】



【図 1 2 - 2】

\$AACC[6]	\$AACC[4]	\$AACC[2]	\$AACC[0]	\$aDsIO
-	-	-	-	-
PERIOD	-	-	-	-
-	-	-	-	-
$R_3 = x_0 \cdot CW_{3,0} + P_3$	$R_2 = x_0 \cdot CW_{2,0} + P_2$	$R_1 = x_0 \cdot CW_{1,0} + P_1$	$R_0 = x_0 \cdot CW_{0,0} + P_0$	-
$R_3 = x_1 \cdot CW_{3,1}$	$R_2 = x_1 \cdot CW_{2,1}$	$R_1 = x_1 \cdot CW_{1,1}$	$R_0 = x_1 \cdot CW_{0,1}$	-
$R_3 = x_2 \cdot CW_{3,2}$	$R_2 = x_2 \cdot CW_{2,2}$	$R_1 = x_2 \cdot CW_{1,2}$	$R_0 = x_2 \cdot CW_{0,2}$	-
$R_3 = x_3 \cdot CW_{3,3}$	$R_2 = x_3 \cdot CW_{2,3}$	$R_1 = x_3 \cdot CW_{1,3}$	$R_0 = x_3 \cdot CW_{0,3}$	-
$R_{11} = x_4 \cdot CW_{3,0} + P_{11}$	$R_{10} = x_4 \cdot CW_{2,0} + P_{10}$	$R_9 = x_4 \cdot CW_{1,0} + P_9$	$R_8 = x_4 \cdot CW_{0,0} + P_8$	$R_0 \cdot R_1$
$R_{11} = x_5 \cdot CW_{3,1}$	$R_{10} = x_5 \cdot CW_{2,1}$	$R_9 = x_5 \cdot CW_{1,1}$	$R_8 = x_5 \cdot CW_{0,1}$	$R_2 \cdot R_3$
$R_{11} = x_6 \cdot CW_{3,2}$	$R_{10} = x_6 \cdot CW_{2,2}$	$R_9 = x_6 \cdot CW_{1,2}$	$R_8 = x_6 \cdot CW_{0,2}$	$R_4 \cdot R_5$
$R_{11} = x_7 \cdot CW_{3,3}$	$R_{10} = x_7 \cdot CW_{2,3}$	$R_9 = x_7 \cdot CW_{1,3}$	$R_8 = x_7 \cdot CW_{0,3}$	$R_6 \cdot R_7$
$R_{19} = x_8 \cdot CW_{3,0} + P_{19}$	$R_{18} = x_8 \cdot CW_{2,0} + P_{18}$	$R_{17} = x_8 \cdot CW_{1,0} + P_{17}$	$R_{16} = x_8 \cdot CW_{0,0} + P_{16}$	$R_8 \cdot R_9$
$R_{19} = x_9 \cdot CW_{3,1}$	$R_{18} = x_9 \cdot CW_{2,1}$	$R_{17} = x_9 \cdot CW_{1,1}$	$R_{16} = x_9 \cdot CW_{0,1}$	$R_{10} \cdot R_{11}$
$R_{19} = x_{10} \cdot CW_{3,2}$	$R_{18} = x_{10} \cdot CW_{2,2}$	$R_{17} = x_{10} \cdot CW_{1,2}$	$R_{16} = x_{10} \cdot CW_{0,2}$	$R_{12} \cdot R_{13}$
$R_{19} = x_{11} \cdot CW_{3,3}$	$R_{18} = x_{11} \cdot CW_{2,3}$	$R_{17} = x_{11} \cdot CW_{1,3}$	$R_{16} = x_{11} \cdot CW_{0,3}$	$R_{14} \cdot R_{15}$
$R_{27} = x_{12} \cdot CW_{3,0} + P_{27}$	$R_{26} = x_{12} \cdot CW_{2,0} + P_{26}$	$R_{25} = x_{12} \cdot CW_{1,0} + P_{25}$	$R_{24} = x_{12} \cdot CW_{0,0} + P_{24}$	$R_{16} \cdot R_{17}$

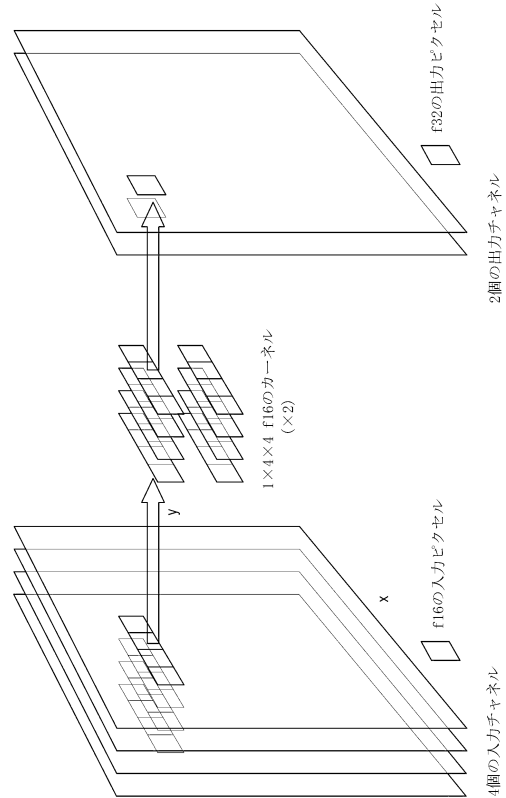
図 1 2 続き

【図 1 2 - 1】

\$aSc011	\$AACC[14]	\$AACC[12]	\$AACC[10]	\$AACC[8]
$0 \cdot P_0 \cdot P_1$	-	-	-	-
$0 \cdot P_2 \cdot P_3$	-	-	-	[WARM UP]
$0 \cdot P_4 \cdot P_5$	-	-	-	-
$0 \cdot P_6 \cdot P_7$	-	-	-	-
$x_0 \cdot P_8 \cdot P_9$	$R_7 = x_0 \cdot CW_{7,0} + P_7$	$R_6 = x_0 \cdot CW_{6,0} + P_6$	$R_5 = x_0 \cdot CW_{5,0} + P_5$	$R_4 = x_0 \cdot CW_{4,0} + P_4$
$x_1 \cdot P_{10} \cdot P_{11}$	$R_7 = x_1 \cdot CW_{7,1}$	$R_6 = x_1 \cdot CW_{6,1}$	$R_5 = x_1 \cdot CW_{5,1}$	$R_4 = x_1 \cdot CW_{4,1}$
$x_2 \cdot P_{12} \cdot P_{13}$	$R_7 = x_2 \cdot CW_{7,2}$	$R_6 = x_2 \cdot CW_{6,2}$	$R_5 = x_2 \cdot CW_{5,2}$	$R_4 = x_2 \cdot CW_{4,2}$
$x_3 \cdot P_{14} \cdot P_{15}$	$R_7 = x_3 \cdot CW_{7,3}$	$R_6 = x_3 \cdot CW_{6,3}$	$R_5 = x_3 \cdot CW_{5,3}$	$R_4 = x_3 \cdot CW_{4,3}$
$x_4 \cdot P_{16} \cdot P_{17}$	$R_{15} = x_4 \cdot CW_{7,0} + P_{15}$	$R_{14} = x_4 \cdot CW_{6,0} + P_{14}$	$R_{13} = x_4 \cdot CW_{5,0} + P_{13}$	$R_{12} = x_4 \cdot CW_{4,0} + P_{12}$
$x_5 \cdot P_{18} \cdot P_{19}$	$R_{15} = x_5 \cdot CW_{7,1}$	$R_{14} = x_5 \cdot CW_{6,1}$	$R_{13} = x_5 \cdot CW_{5,1}$	$R_{12} = x_5 \cdot CW_{4,1}$
$x_6 \cdot P_{20} \cdot P_{21}$	$R_{15} = x_6 \cdot CW_{7,2}$	$R_{14} = x_6 \cdot CW_{6,2}$	$R_{13} = x_6 \cdot CW_{5,2}$	$R_{12} = x_6 \cdot CW_{4,2}$
$x_7 \cdot P_{22} \cdot P_{23}$	$R_{15} = x_7 \cdot CW_{7,3}$	$R_{14} = x_7 \cdot CW_{6,3}$	$R_{13} = x_7 \cdot CW_{5,3}$	$R_{12} = x_7 \cdot CW_{4,3}$
$x_8 \cdot P_{24} \cdot P_{25}$	$R_{31} = x_8 \cdot CW_{7,0} + P_{23}$	$R_{22} = x_8 \cdot CW_{6,0} + P_{22}$	$R_{21} = x_8 \cdot CW_{5,0} + P_{21}$	$R_{20} = x_8 \cdot CW_{4,0} + P_{20}$
$x_9 \cdot P_{26} \cdot P_{27}$	$R_{23} = x_8 \cdot CW_{7,1}$	$R_{22} = x_8 \cdot CW_{6,1}$	$R_{21} = x_8 \cdot CW_{5,1}$	$R_{20} = x_8 \cdot CW_{4,1}$
$x_{10} \cdot P_{28} \cdot P_{29}$	$R_{23} = x_{10} \cdot CW_{7,2}$	$R_{22} = x_{10} \cdot CW_{6,2}$	$R_{21} = x_{10} \cdot CW_{5,2}$	$R_{20} = x_{10} \cdot CW_{4,2}$
$x_{11} \cdot P_{30} \cdot P_{31}$	$R_{23} = x_{11} \cdot CW_{7,3}$	$R_{22} = x_{11} \cdot CW_{6,3}$	$R_{21} = x_{11} \cdot CW_{5,3}$	$R_{20} = x_{11} \cdot CW_{4,3}$
$x_{12} \cdot P_{32} \cdot P_{33}$	$R_{31} = x_{12} \cdot CW_{7,0} + P_{31}$	$R_{30} = x_{12} \cdot CW_{6,0} + P_{30}$	$R_{29} = x_{12} \cdot CW_{5,0} + P_{29}$	$R_{28} = x_{12} \cdot CW_{4,0} + P_{28}$

P_i は、単精度入力部分 $\ln$ である。
 x_i は、f16v4入力ベクトルである。
 $CW_{m,n}$ は、共通の重み状態 $CWEL_{m,n}$ である。
 R_i は、 P_i で始まる連続ドット縦列の最終的な単精度結果である。

【図 1 3】



【図 14 - 1】

\$aScd[1]\$	\$aAcc[14]\$	\$aAcc[10]\$	\$aAcc[6]\$	\$aAcc[2]\$
$x_0P_0P_1$	-	$R_1=x_0CW_{5,0}+P_1$	-	-
$x_1P_2P_3$	-	$R_3=x_1CW_{5,0}+P_3$	$R_1=x_1CW_{3,0}$	-
$x_2P_4P_5$	-	$R_5=x_2CW_{5,0}+P_5$	$R_3=x_2CW_{3,0}$	$R_1=x_2CW_{1,0}$
$x_3P_6P_7$	-	$R_7=x_3CW_{5,0}+P_7$	$R_5=x_3CW_{3,0}$	$R_3=x_3CW_{1,0}$
$x_4P_8P_9$	-	$R_9=x_4CW_{5,0}+P_9$	$R_7=x_4CW_{3,0}$	$R_5=x_4CW_{1,0}$
$x_5P_{10}P_{11}$	-	$R_{11}=x_5CW_{5,0}+P_{11}$	$R_9=x_5CW_{3,0}$	$R_7=x_5CW_{1,0}$
$x_6P_{12}P_{13}$	-	$R_{13}=x_6CW_{5,0}+P_{13}$	$R_{11}=x_6CW_{3,0}$	$R_9=x_6CW_{1,0}$

表3.156: f16v4sisoslic、1x4のシーケンス例。

\$aScd[1]\$	\$aAcc[14]\$	\$aAcc[10]\$	\$aAcc[6]\$	\$aAcc[2]\$
$x_0P_0P_1$	$R_1=x_0CW_{7,0}+P_1$	-	-	-
$x_1P_2P_3$	$R_3=x_1CW_{7,0}+P_3$	$R_1=x_1CW_{5,0}$	-	-
$x_2P_4P_5$	$R_5=x_2CW_{7,0}+P_5$	$R_3=x_2CW_{5,0}$	$R_1=x_2CW_{3,0}$	-
$x_3P_6P_7$	$R_7=x_3CW_{7,0}+P_7$	$R_5=x_3CW_{5,0}$	$R_3=x_3CW_{3,0}$	$R_1=x_3CW_{1,0}$
$x_4P_8P_9$	$R_9=x_4CW_{7,0}+P_9$	$R_7=x_4CW_{5,0}$	$R_5=x_4CW_{3,0}$	$R_3=x_4CW_{1,0}$
$x_5P_{10}P_{11}$	$R_{11}=x_5CW_{7,0}+P_{11}$	$R_9=x_5CW_{5,0}$	$R_7=x_5CW_{3,0}$	$R_5=x_5CW_{1,0}$
$x_6P_{12}P_{13}$	$R_{13}=x_6CW_{7,0}+P_{13}$	$R_{11}=x_6CW_{5,0}$	$R_9=x_6CW_{3,0}$	$R_7=x_6CW_{1,0}$

P_n は、単精度入力部分 m である。
 x_n は、f16v4入力ベクトルである。
 $CW_{m,n}$ は、共通の重み状態\$CWEI_{m,n}\$である。
 R_n は、 P_n で始まる連続ドット積累積の最終的な単精度結果である。

【図 14 - 2】

\$aAcc[12]\$	\$aAcc[8]\$	\$aAcc[4]\$	\$aAcc[0]\$	\$aDis0
-	$R_0=x_0CW_{4,0}+P_0$	-	-	-
-	$R_2=x_1CW_{4,0}+P_2$	$R_0=x_1CW_{2,0}$	-	-
-	$R_4=x_2CW_{4,0}+P_4$	$R_2=x_2CW_{2,0}$	$R_0=x_2CW_{0,0}$	-
-	$R_6=x_3CW_{4,0}+P_6$	$R_4=x_3CW_{2,0}$	$R_2=x_3CW_{0,0}$	R_0R_1
-	$R_8=x_4CW_{4,0}+P_8$	$R_6=x_4CW_{2,0}$	$R_4=x_4CW_{0,0}$	R_2R_3
-	$R_{10}=x_5CW_{4,0}+P_{10}$	$R_8=x_5CW_{2,0}$	$R_6=x_5CW_{0,0}$	R_4R_5
-	$R_{12}=x_6CW_{4,0}+P_{12}$	$R_{10}=x_6CW_{2,0}$	$R_8=x_6CW_{0,0}$	R_6R_7

\$aAcc[12]\$	\$aAcc[8]\$	\$aAcc[4]\$	\$aAcc[0]\$	\$aDis0
$R_0=x_0CW_{6,0}+P_0$	-	-	-	-
$R_2=x_1CW_{6,0}+P_2$	$R_0=x_1CW_{4,0}$	-	-	-
$R_4=x_2CW_{6,0}+P_4$	$R_2=x_2CW_{4,0}$	$R_0=x_2CW_{2,0}$	-	-
$R_6=x_3CW_{6,0}+P_6$	$R_4=x_3CW_{4,0}$	$R_2=x_3CW_{2,0}$	$R_0=x_3CW_{0,0}$	-
$R_8=x_4CW_{6,0}+P_8$	$R_6=x_4CW_{4,0}$	$R_4=x_4CW_{2,0}$	$R_2=x_4CW_{0,0}$	R_0R_1
$R_{10}=x_5CW_{6,0}+P_{10}$	$R_8=x_5CW_{4,0}$	$R_6=x_5CW_{2,0}$	$R_4=x_5CW_{0,0}$	R_2R_3
$R_{12}=x_6CW_{6,0}+P_{12}$	$R_{10}=x_6CW_{4,0}$	$R_8=x_6CW_{2,0}$	$R_6=x_6CW_{0,0}$	R_4R_5

図 14 続き

フロントページの続き

- (72)発明者 アラン グラハム アレクサンダー
イギリス国, ジーエル12 8アールワイ, ウォットン-アンダー-エッジ, キングスウッド, テ
ィンデル ビュー 24
- (72)発明者 サイモン クリスチャン ノウルズ
イギリス国, ビーエー2 9エーエヌ, コーストン, ザ パートン, ヒル ハウス
- (72)発明者 マルドゥラ ゴア
イギリス国, ビーエー1 3ビーダブリュ, バス, セントジョーンズ ロード 14

審査官 井上 宏一

- (56)参考文献 再公表特許第2010/044242 (JP, A1)
特表2014-526758 (JP, A)

- (58)調査した分野(Int.Cl., DB名)
G06F 9/30 - 9/355