

(19) 日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2019-139338

(P2019-139338A)

(43) 公開日 令和1年8月22日 (2019. 8. 22)

(51) Int. Cl.
G06N 3/02 (2006.01)F I
G06N 3/02

テーマコード (参考)

審査請求 未請求 請求項の数 10 O L (全 43 頁)

(21) 出願番号	特願2018-19782 (P2018-19782)	(71) 出願人	000005223
(22) 出願日	平成30年2月7日 (2018. 2. 7)		富士通株式会社
			神奈川県川崎市中原区上小田中4丁目1番1号
		(74) 代理人	100113608
			弁理士 平川 明
		(74) 代理人	100105407
			弁理士 高田 大輔
		(74) 代理人	100175190
			弁理士 大竹 裕明
		(72) 発明者	野津 隆弘
			神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(72) 発明者	伊藤 真紀子
			神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

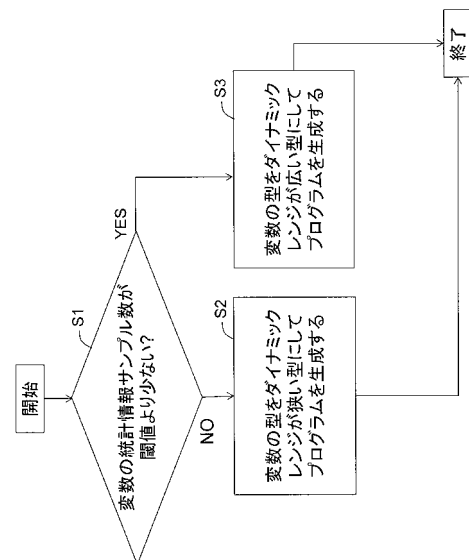
(54) 【発明の名称】 情報処理装置、情報処理方法、およびプログラム

(57) 【要約】

【課題】 深層学習等による演算を精度よく実行するとともに回路規模と消費電力を削減できるようにすることである。

【解決手段】 情報処理装置は、演算処理部で実行されるプログラムを生成する。演算処理部は、演算結果の値の分布を含む統計情報を取得可能である。情報処理装置は、演算処理部での演算で取得される統計情報のサンプル数が基準値より多いと判断した場合には、演算の対象データに使用可能なデータ型のうち、表現可能な値の最小値に対する最大値の比率がより小さいデータ型を前記対象データに設定し、サンプル数が基準値より少ないと判断した場合には、比率がより大きいデータ型を対象データに設定して、プログラムを生成する生成部を備える。

【選択図】 図35



【特許請求の範囲】**【請求項 1】**

演算処理部で実行されるプログラムを生成する情報処理装置であって、前記演算処理部は、演算結果の値の分布を含む統計情報を取得可能であり、

前記情報処理装置は、前記演算処理部での演算で取得される統計情報のサンプル数が基準値より多いと判断した場合、前記演算の対象データに使用可能なデータ型のうち、表現可能な値の最小値に対する最大値の比率がより小さいデータ型を前記対象データに設定し、前記サンプル数が前記基準値より少ないと判断した場合には、前記比率がより大きいデータ型を前記対象データに設定して、プログラムを生成する生成部を備える情報処理装置。

10

【請求項 2】

前記演算処理部は、固定小数点数の演算および浮動小数点数の演算の少なくとも一方を実行し、

前記生成部は、前記サンプル数が前記基準値より多いと判断した場合、固定小数点数のデータ型を前記対象データに設定し、前記サンプル数が前記基準値より少ないと判断した場合、浮動小数点数のデータ型を前記対象データに設定する請求項 1 に記載の情報処理装置。

【請求項 3】

前記プログラムは複数階層のニューラルネットワークによる処理を実行するものであり、

20

前記ニューラルネットワークを定義する定義情報から前記ニューラルネットワークの各層における演算回数を取得する第 1 の取得部と、

前記ニューラルネットワークの各層における演算回数と前記演算処理部における演算回数に対する前記統計情報の取得割合とを基に前記統計情報のサンプル数を取得する第 2 の取得部と、をさらに備える請求項 1 または 2 に記載の情報処理装置。

【請求項 4】

前記演算処理部による演算に基づき、前記基準値を決定する決定部をさらに備える請求項 1 から 3 のいずれか 1 項に記載の情報処理装置。

【請求項 5】

前記決定部は、浮動小数点数を用いて深層学習を実行したときの学習誤差と、固定小数点数を用いて深層学習を実行したときの学習誤差との差に基づいて前記基準値を決定する請求項 4 に記載の情報処理装置。

30

【請求項 6】

前記決定部は、所定の限定された回数の深層学習によって前記差を取得する請求項 5 に記載の情報処理装置。

【請求項 7】

前記情報処理装置が前記演算処理部を有し、前記生成部の処理が前記演算処理部で実行される請求項 1 から 6 のいずれか 1 項に記載の情報処理装置。

【請求項 8】

前記生成部の処理が前記演算処理部とは異なるプロセッサで実行される請求項 1 から 6 のいずれか 1 項に記載の情報処理装置。

40

【請求項 9】

演算処理部で実行されるプログラムを生成する情報処理装置が実行する情報処理方法であって、前記演算処理部は、演算結果の値の分布を含む統計情報を取得可能であり、前記情報処理装置が、

前記演算処理部での演算で取得される統計情報のサンプル数が基準値より多いと判断した場合、前記演算の対象データに使用可能なデータ型のうち、表現可能な値の最小値に対する最大値の比率がより小さいデータ型を前記対象データに設定し、前記サンプル数が前記基準値より少ないと判断した場合、前記比率がより大きいデータ型を前記対象データに設定して、プログラムを生成する情報処理方法。

50

【請求項 10】

演算処理部で実行されるプログラムを生成する情報処理装置に実行させるプログラムであって、前記演算処理部は、演算結果の値の分布を含む統計情報を取得可能であり、前記情報処理装置に、

前記演算処理部での演算で取得される統計情報のサンプル数が基準値より多いと判断した場合、前記演算の対象データに使用可能なデータ型のうち、表現可能な値の最小値に対する最大値の比率がより小さいデータ型を前記対象データに設定し、前記サンプル数が前記基準値より少ないと判断した場合、前記比率がより大きいデータ型を前記対象データに設定して、プログラムを生成することを実行させるプログラム。

【発明の詳細な説明】

10

【技術分野】

【0001】

本発明は、情報処理装置、情報処理方法、およびプログラムに関する。

【背景技術】

【0002】

今日、人工知能を用いた機械学習のうち、特に深層学習（DL）へのニーズが高まっている。深層学習は、多層ニューラルネットワーク（DNN）を用いた機械学習の手法といえることができる。図1は、ニューラルネットワークの構成を例示する図である。ニューラルネットワークは、神経細胞（ニューロン）をコンピュータ上にモデル化したものである。神経細胞は、細胞体と、他の細胞体から入力される信号を受ける樹状突起と、他の細胞体に信号を出力する軸索とを有する。また、信号を出力する軸索の末端と信号を受ける樹状突起との間にはシナプスと呼ばれる信号の伝達構造が形成されている。ニューラルネットワークでは、神経細胞間でのシナプスを介した情報伝達がモデル化されている。

20

【0003】

ニューラルネットワーク全体は一つの関数になっている。ニューラルネットワークにある入力が与えられれば、ニューラルネットワークは、与えられた入力にしたがって決まった値を出力する。ニューラルネットワークが既存のプログラムと異なるところは、ある入力に対する目的の値（教師信号）が与えられれば、与えられた目的の値に合わせて出力する値を目的に近づけるように調整することができる点である。すなわち、ニューラルネットワークには内部に多数のパラメータが存在する。ニューラルネットワークは、内部に存在するパラメータの値を調整することにより、目的とする関数の形を実現していくことになる。

30

【0004】

多層ニューラルネットワークは複数の階層から構成されている。それぞれの階層で行う計算の内容は異なっているが、おおよそ次のいずれかの演算に大別される。すなわち、多層ニューラルネットワークの演算としては、（a）畳み込み（Convolution）演算、（b）フルコネク（FullConnect）演算（内積演算ともいう）、（c）ReLU（ランプ関数）演算（活性化演算ともいう）、（d）プーリング（Pooling）演算、（e）ソフトマックス（Softmax）演算、（f）誤差評価演算が例示される。

【0005】

40

例えば、画像分類の場合、初めのいくつかの層は畳み込み、ReLU、Pooling演算を行い、上位の層がフルコネク、ReLUの演算を行う。最後の層がソフトマックス演算を行って入力画像に対する分類カテゴリの確率を出力する。ニューラルネットワークの計算のうち、支配的になるのが畳み込み演算とフルコネク演算である。

【0006】

図2に、ニューラルネットワークにおける畳み込み演算の処理を例示する。畳み込み演算はフィルタ演算とも呼ばれ、主に画像データに対して実行される。演算は、入力パラメータ X （ベクトル）の要素と、重み W （ベクトル）の要素の積和演算 $\sum xw$ である。入力・出力のすべてのチャンネルの組み合わせに対しフィルタが用意されている。畳み込み演算はすべての入力チャンネルの結果の合計にバイアスを加えて出力値 Z としている。

50

【0007】

図3に、フルコネクト演算の処理を例示する。フルコネクト演算は内積演算とも呼ばれる。フルコネクト演算では、入力、出力すべての組に対し重みが定義されている。フルコネクト演算は、すべての入力 $X \{x_0, x_1, \dots, x_c\}$ とその重み $W \{w_{00}, \dots, w_{cd}\}$ の積和演算を実行し、積和演算結果にバイアス $b \{b_0, b_1, \dots, b_d\}$ を加えて出力値 $Z \{z_0, z_1, \dots, z_d\}$ としている。

【0008】

多層ニューラルネットワークでは、深層学習が実行される。そして、深層学習が実行される多層ニューラルネットワークの認識性能を高めるため、多層ニューラルネットワークが大規模化される傾向にある。例えば、多層ニューラルネットワークで処理されるパラメータ数は、数百万から1億数千万にも及ぶ。多層ニューラルネットワークが人の脳に近づくためには、究極的に、パラメータ数は1千数百億にも及ぶと考えられる。したがって、今後、深層学習における学習データは増加し、多層ニューラルネットワークにおける計算負荷、メモリ負荷が増大することが予想される。そのため、増え続ける学習データに対して、認識性能と学習効率の改善が望まれている。認識性能と学習効率を改善し、負荷を軽減するため、多層ニューラルネットワークが軽量化されることが望ましい。

【0009】

ところで、深層学習においては、乗算、積和演算、ベクトル乗算を含む様々な演算が実行される。ただし、深層学習では、個々の演算精度への要求は、通常の演算処理ほど厳密ではない。例えば、通常の演算処理等では、プログラマは極力桁あふれを発生させないようにコンピュータプログラムを開発する。一方、深層学習では、大きな値がある程度飽和することは許容される。深層学習では、複数の入力データを畳み込み演算するときの係数（重み）の調整が主な処理となり、入力データのうち、極端なデータは重視されないことが多いからである。また、大量のデータを繰り返し用いて係数を調整するため、一度飽和された値も、学習の進行に合わせて桁調整を行なうことで、飽和されずに係数の調整に反映できるようになるためである。

そこで、このような深層学習の特性を考慮し、深層学習用の演算処理装置のチップ面積の削減、電力効率の向上等を図るため、浮動小数点数を用いなくて、固定小数点数による演算を用いることが考えられる。浮動小数点数演算よりも固定小数点演算の方が回路構成を簡素にできるからである。

【0010】

図4にデータ表現に用いるビットの構成を例示する。32ビット浮動小数点数、16ビット固定小数点数、8ビット固定小数点数のように、深層学習で処理するデータ（重みとパラメータ）のデータ表現に用いるビット幅を縮小することで、多層ニューラルネットワークで扱うデータ量を削減できる。扱うデータ量が削減されることで、深層学習の処理量が削減され、学習時間が短縮されることが期待できる。

【0011】

しかしながら、固定小数点数は、取りうる値のダイナミックレンジが狭いので、浮動小数点数より演算精度が劣化する場合がある。図5に、32ビット浮動小数点数、16ビット固定小数点数、8ビット固定小数点数による処理と、推論の精度の関係をモデル化して例示する。図では、「固定小数点数」が「整数」と記述されている。固定小数点数が整数に限定される訳ではない。固定小数点数は2進整数とも理解できるので、本明細書中では、固定小数点数を整数という場合がある。図のように、ビット幅を削減すると演算精度が低下することが予測される。演算精度が低下すると、深層学習がうまく実施できない場合もあり得る。すなわち、深層学習では、フォワード方向およびバックワード方向に積和演算が多数回繰り返され、演算結果が固定小数点数のダイナミックレンジを超えてしまうこともあるからである。そこで、演算精度を改善する技術により、ビット幅を削減することに起因する上記課題を克服することが望まれる。このため、固定小数点数を拡張した技術が提案されている。

【0012】

例えば、Mixed Fixed Pointによる処理では、プログラム全体として統一した小数点位置を使うのではなく、変数ごとに適した小数点位置（Qフォーマット）が用いられる。Qフォーマットとして、例えば、Q 3. 1 2 フォーマットは、符号ビット 1 桁、整数部 3 桁、小数点以下 1 2 桁の 1 6 ビットデータを定義する。Mixed Fixed Pointでは、変数ごとに、小数点位置、つまり、整数部の桁数と小数点以下の桁数が異なるものとして処理される。

【0013】

他の例として、Dynamic Fixed Point（動的固定小数点数）による処理では、実行中に変数の値域が取得され、一定のタイミングで小数点位置が見直される。したがって、Mixed Fixed Point演算、およびDynamic Fixed Point演算は、浮動小数点演算よりも処理が簡単な固定小数点演算に、浮動小数点演算の側面を加えたものと言える。

10

【0014】

さらに、Mixed Fixed Point演算、Dynamic Fixed Point演算による処理を実行するためのプログラム向けの機能を有するDigital Signal Processor（DSP）も提案されている。例えば、ブロック・シフト指定付きの演算命令を実行するDSPがある。ブロック・シフト指定付きの演算命令は、変数のビット幅より大きいビット幅で演算を実行し、演算結果から値をシフトして切り出して変数用のレジスタに格納する。この命令では、演算結果から値を切り出すときのシフト量 S （例えば、-128~127）が即値/汎用レジスタで指定可能である。例えば、DSPがResult = Saturate (((in1（演算子）in2）>> S), 16)という命令を実行すると、演算結果を S ビットシフトし、下位16bitを残し、上位ビットを飽和処理する。 $S \geq 0$ の場合には、DSPは、演算結果を算術右シフトし、すなわち、符号ビットを埋め込んで右シフトし、下位ビットを削除する。一方、 $S < 0$ の場合は、DSPは、演算結果を算術左シフト、すなわち、符号ビットを維持して左シフトし、補数の下位ビットを削除する。

20

【0015】

ブロック・カウント・リーディング・サイン（BCLS）出力を実行するDSPも提案されている。BCLS出力は、DSPが演算結果のcount leading signを取って、結果をレジスタに書き込む機能である。ここで、count leading signとは、正数最上位のビット1の位置（負数で最上位のビット0の位置）をいう。例えば、DSPが、max(block_count_leading_sign(in1（演算子）in2）-1)を実行すると、変数in1と変数in2の演算子による演算結果から、正数最上位のビット1の位置（負数で最上位のビット0の位置）がレジスタに記録される。

30

【0016】

なお、非特許文献1によるDynamic Fixed Point数による処理では、演算または変数の代入ごとに、オーバーフローの有無を記録し、オーバーフロー率（全演算回数に対するオーバーフローした演算の回数、全代入回数に対するオーバーフローした代入回数等）が算出される。そして、この処理では、所定の期間ごとに、各変数の小数点位置を以下の手順1、2で変更する。

（手順1）オーバーフロー率が規定値（ r_{max} ）より大きい場合、小数点位置を1つ下げる。

40

（手順2）オーバーフロー率の2倍の値が規定値（ r_{max} ）以下の場合、小数点位置を1つ上げる。

【先行技術文献】

【特許文献】

【0017】

【特許文献1】特開平7-84975号公報

【非特許文献】

【0018】

【非特許文献1】Courbariaux et al., "TRAINING DEEP NEURAL NETWORKS WITH LOW PRECISION MULTIPLICATIONS" Accepted as a workshop contribution at ICLR 2015, Intern

50

ational Conference on Learning Representations(ICLR), 2015年9月23日

【発明の概要】

【発明が解決しようとする課題】

【0019】

上記従来の技術においては、オーバーフロー率を基に小数点位置を1ビットずつ調整するため学習精度が劣化する。例えば、 k 個のデータの学習ごとに小数点位置を調整する場合、小数点位置が適正位置から n ビットずれている場合、小数点位置が適切な位置に調整されるまでに、 $(n * k)$ 回の学習が進む。その結果、小数点位置が適正な位置になるまでに、飽和するデータやアンダーフローするデータが増え、学習が収束しない、あるいは、学習精度の劣化につながるおそれがある。

10

1つの側面では本発明の目的は、深層学習等による演算を精度よく実行するとともに回路規模と消費電力を削減できるようにすることである。

【課題を解決するための手段】

【0020】

開示の技術の一側面は、演算処理部で実行されるプログラムを生成する情報処理装置によって例示される。この演算処理部は、演算結果の値の分布を含む統計情報を取得可能である。本情報処理装置は、前記演算処理部での演算で取得される統計情報のサンプル数が基準値より多いと判断した場合には、前記演算の対象データに使用可能なデータ型のうち、表現可能な値の最小値に対する最大値の比率がより小さいデータ型を前記対象データに設定し、前記サンプル数が基準値より少ないと判断した場合には、前記比率がより大きいデータ型を前記対象データに設定して、プログラムを生成する生成部を備える。

20

【発明の効果】

【0021】

本情報処理装置によれば、深層学習等による演算を精度よく実行するとともに回路規模と消費電力を削減できる。

【図面の簡単な説明】

【0022】

【図1】ニューラルネットワークの構成を例示する図である。

【図2】ニューラルネットワークにおける畳み込み演算の処理を例示する図である。

【図3】フルコネクト演算の処理を例示する図である。

30

【図4】データ表現に用いるビットの構成を例示する図である。

【図5】32ビット浮動小数点数、16ビット固定小数点数、8ビット固定小数点数による処理と、推論の精度の関係をモデル化して例示する図である。

【図6】非符号となる最上位ビット位置の分布データを例示する図である。

【図7】プロセッサの構成を例示する図である。

【図8】プロセッサの回路ブロックを例示する図である。

【図9】ベクトルユニットの詳細を例示する図である。

【図10】スカラユニットの構成を例示する図である。

【図11】非符号最上位ビット位置を検出する統計情報取得部の真理値表を例示する図である。

40

【図12】非符号となる最上位ビット位置を取得する統計情報取得部のハードウェア回路の構成を例示する図である。

【図13】非符号最下位ビット位置を検出する統計情報取得部の真理値表を例示する図である。

【図14】非符号となる最下位ビット位置を取得する統計情報取得部のハードウェア回路の構成を例示する図である。

【図15】統計情報集約部の処理を例示する図である。

【図16】統計情報集約部のハードウェア回路の構成を例示する図である。

【図17】統計情報集約部の処理を例示する図である。

【図18】統計情報集約部のハードウェア回路の構成を例示する図である。

50

- 【図 1 9】統計情報集約部の具体的な構成例である。
- 【図 2 0】統計情報格納部のハードウェア回路の構成を例示する図である。
- 【図 2 1】統計情報格納部のハードウェア回路の構成を例示する図である。
- 【図 2 2】学習ネットワークによる学習結果の例である。
- 【図 2 3】実施形態 2 に係る情報処理システムの構成を例示する図である。
- 【図 2 4】深層学習を実行するときのモジュールの構成を例示する図である。
- 【図 2 5】ホストマシンの構成を例示する図である。
- 【図 2 6】DL実行ハードウェアの構成を例示する図である。
- 【図 2 7】DL実行プロセッサの構成を例示する図である。
- 【図 2 8】プログラム生成に係るホストマシンの論理構成を例示する図である。
- 【図 2 9】テンプレートを例示する図である。
- 【図 3 0】ニューラルネットワーク記述データの構成を例示する図である。
- 【図 3 1】prototextにおけるtopとbottomの定義を例示する図である。
- 【図 3 2】ニューラルネットワークの入力データの定義を例示する図である。
- 【図 3 3】ニューラルネットワークの畳み込み層の定義を例示する図である。
- 【図 3 4】ニューラルネットワークのフルコネクト層の定義を例示する図である。
- 【図 3 5】実施形態 2 におけるプログラム生成の簡略フローを例示する図である。
- 【図 3 6】閾値決定の処理を例示する図である。
- 【図 3 7】閾値決定の処理の第 1 の変形を例示する図である。
- 【図 3 8】閾値決定の処理の第 2 の変形を例示する図である。
- 【図 3 9】プログラム生成処理を例示する図である。
- 【図 4 0】変数型決定部の処理の詳細を例示する図である。
- 【図 4 1】プログラム生成部の処理を例示する図である。
- 【図 4 2】型に合わせた型変換パラメータの設定処理を例示する図である。
- 【図 4 3】シフト幅の設定を例示する図である。
- 【図 4 4】最上位ビット位置を決定する処理を例示する図である。
- 【図 4 5】スケールファクタの決定方法を例示する図である。
- 【図 4 6】認識結果を例示する図である。

10

20

【発明を実施するための形態】

【0023】

30

以下、図面を参照して、一実施形態に係る情報処理装置について説明する。以下の実施形態の構成は例示であり、本情報処理装置は実施形態の構成には限定されない。

〔実施形態 1〕

【0024】

以下、図 6 から図 2 1 により、実施形態 1 に係る情報処理装置 1、情報処理装置 1 が実行する情報処理方法、および情報処理装置 1 で実行されるプログラムについて説明する。本実施形態は、一例であり、情報処理装置 1 は、本実施形態の構成に限定される訳ではない。

<統計情報>

【0025】

40

本実施形態では、情報処理装置 1 のプロセッサが統計情報を取得することで、統計情報取得のためのプログラム中のオーバーヘッドが削減される。ここで、プロセッサが取得する統計情報は、例えば、以下のいずれか、または、これらの組み合わせをいう。情報処理装置 1 が実行するアプリケーションプログラムは、プロセッサから統計情報を取得することで、小数点位置を最適化する。アプリケーションプログラムの処理にしたがってプロセッサは、Dynamic Fixed Point演算のための命令を実行する。

(1) 非符号となる最上位ビット位置の分布

【0026】

図 6 に、非符号となる最上位ビット位置の分布データを例示する。図 6 は演算の途中結果が 40bit で、固定小数点数の桁あわせのために 14bit 右シフトされるデータに対する例で

50

ある。非符号となる最上位ビット位置とは、正数に対しては、ビットが1となっている最上位のビット位置をいう。また、負数に対しては、ビット0となっている最上位のビット位置をいう。非符号となる最上位ビット位置は、例えば、ビットの並びを最上位ビットであるbit[39]から最下位ビットであるbit[0]としたとき、符号ビットbit[39]と異なるbit[k]のうち、インデックスkが最大のものをいう。非符号となる最上位ビット位置の分布が得られると、絶対値としての値の分布範囲が把握可能となる。なお、統計情報は、演算結果の値の分布を含むものと言える。

【0027】

図6で、縦軸は、非符号となる最上位ビット位置の出現個数であり、横軸は最上位のビットの位置 Count Leading Sign (CLS) である。図6では、ビット0の右側に小数点があると仮定する。本実施形態では、情報処理装置1のプロセッサの演算回路および演算回路内のレジスタは、命令のオペランドで指定されるレジスタのビット数（例えば、16ビット）以上のビット幅（例えば、40ビット）を有する。ただし、情報処理装置1のプロセッサの演算回路および演算回路内のレジスタのビット幅が40ビットに限定される訳ではない。また、演算結果は、例えば、16ビットのレジスタ等、演算回路よりも少ないビット幅のレジスタ（命令のオペランドで指定されるレジスタ）に格納される。その結果、演算結果は（例えば、40ビット）、オペランドで指定されるシフト量でシフトし、ビット0未満に該当するビットは所定の丸め処理がなされ、オペランドで指定されるレジスタのビット幅を越えるデータ（ビット15を越えるデータ）は、飽和処理される。

【0028】

また、図6の横軸に付与された数値は、固定小数点で表現可能な数値を示す。例えば、情報処理装置1がこの固定小数点数を-2ビットだけシフト（右方向に2ビットシフト）すると、最上位ビットが14の位置にシフトし、飽和される領域が2ビット拡張され、アンダーフローが発生して0となる領域が2ビット減少する。すなわち、情報処理装置1が小数点位置を2ビット左にシフトすると、飽和される領域が2ビット拡張され、アンダーフローが発生する領域が2ビット減少する。また、例えば、情報処理装置がこの固定小数点数を2ビットだけ正方向にシフト（左方向に2ビットシフト）すると、最上位ビットが18の位置にシフトし、飽和される領域が2ビット減少し、アンダーフローが発生する領域が2ビット拡張される。すなわち、情報処理装置1が小数点位置を2ビット右にシフトすると、飽和される領域が2ビット減少し、アンダーフローが発生する領域が2ビット拡張される。

【0029】

情報処理装置1は、学習実行中に、非符号となる最上位のビット位置の分布を得ることで、Dynamic Fixed Point演算における適正なシフト量、つまり、適切な固定小数点位置を直ちに決定できる。例えば、情報処理装置1は、飽和されるデータが指定の割合以下になるように固定小数点位置を決定すればよい。つまり、一例としては、情報処理装置1は、データのアンダーフローが所定の程度となることよりもデータの飽和が所定の程度となること優先して、固定小数点位置を決定できる。

【0030】

非符号となる最上位ビット位置の分布は、情報処理装置1のプロセッサ10（図7参照）内の所定のレジスタ（統計情報レジスタともいう）内に積算される。プロセッサ10は、当該統計情報レジスタからの分布データの読み出し、書き込み、統計情報レジスタのクリア等の命令を実行する。したがって、統計情報レジスタには、前回のクリア命令の実行から現在までに命令実行の対象となった1以上の固定小数点数についての分布データが蓄積される。蓄積された分布データは、読み出し命令によってメモリに読み出される。なお、プロセッサ10は、クリア命令に代えて、統計情報レジスタへのロード命令の実行を可能とし、値0を統計情報レジスタにロードできるようにしてもよい。

（2）非符号となる最下位ビット位置の分布

【0031】

非符号となる最下位ビット位置の分布とは、ビットが符号と異なる値となる最下位のビ

ット位置をいう。例えば、ビットの並びを最上位ビットであるbit[39]から最下位ビットであるbit[0]までの並びとしたとき、符号ビットbit[39]と異なるbit[k]のうち、インデックスkが最小のものをいう。また、非符号となる最下位ビット位置の分布では、有効なデータが含まれる最下位ビットが把握される。

(3) 非符号となる最上位ビット位置の最大値

【0032】

非符号となる最上位ビット位置の最大値は、前回クリア命令が実行されたときから現在に至るまでに、命令実行の対象となった1以上の固定小数点数について符号ビットの値と異なる値となる最上位のビット位置のうちの最大値である。情報処理装置1は、非符号となる最上位のビット位置の最大値をDynamic Fixed Point演算における適正なシフト量、

10

【0033】

プロセッサは、上記統計情報レジスタからの上記最大値の読み出し、統計情報レジスタのクリア等の命令を実行する。したがって、統計情報レジスタには、前回のクリア命令の実行から現在までの最大値が蓄積され、読み出し命令によって最大値がメモリに読み出される。

(4) 非符号となる最下位ビット位置の最小値

【0034】

非符号となる最下位ビット位置の最小値は、前回クリア命令が実行されたときから現在に至るまでの1以上の固定小数点数について符号と異なる値となる最下位のビット位置のうちの最小値である。情報処理装置1は、非符号となる最下位ビット位置の最小値をDynamic Fixed Point演算における適正なシフト量、つまり、適切な小数点位置の決定に利用できる。

20

【0035】

プロセッサ10は、統計情報レジスタからの上記最小値の読み出し、クリア等の命令を実行する。したがって、統計情報レジスタには、前回のクリア命令の実行から現在までの上記最小値が蓄積され、読み出し命令によってメモリに読み出される。

<構成>

【0036】

図7に、情報処理装置1のプロセッサ10の構成を例示する。図7では、プロセッサ10とともに、命令用メモリ（IRAM）21およびデータ用メモリ（DRAM）22も例示されている。プロセッサ10は、single instruction multiple data（SIMD）型の演算処理装置である。

30

【0037】

プロセッサ10は、プログラムカウンタ（PC）111およびデコーダ（Decoder）112を含む制御ユニット11、レジスタファイル12、ベクトル演算用の演算器131、スカラー演算用の演算器（Arithmetic Logic Unit（ALU））141、およびベクトル演算用の演算器131の結果を加算するアキュムレータ132を有している。また、プロセッサ10は、ベクトル演算用の演算器131、スカラー演算用の演算器141、アキュムレータ132等の演算結果、および、データ用メモリ22からの読み出し結果を選択する複数のセレクタ101を有している。なお、図では、複数のセレクタを総称してセレクタ101と呼ぶ。また、複数のベクトル演算用の演算器を総称して演算器131と呼ぶ。

40

【0038】

さらに、プロセッサ10は、セレクタ101によって選択されたデータからの統計情報を取得する統計情報取得部102、統計情報取得部102によって取得された統計情報を格納する統計情報格納部105を有している。統計情報取得部102と統計情報格納部105は、固定小数点数データに対する命令実行後の固定小数点数データ中のビットの分布についての統計情報を取得する取得回路の一例である。なお、図では、複数の統計情報取得部を総称して統計情報取得部102と呼ぶ。

【0039】

50

さらに、プロセッサ 10 は、セクタ 101 によって選択されたデータの固定小数点位置を変更するデータ変換部 103 を有している。また、図では、複数のデータ変換部を総称してデータ変換部 103 と呼ぶ。

【0040】

図のように、プログラムカウンタ 111 の指す命令用メモリ 21 のアドレスから、命令がフェッチされ、デコーダ 112 がフェッチされた命令をデコードする。なお、図では、命令のフェッチを実行する命令フェッチの制御部が省略されている。

【0041】

デコーダ 112 が命令をデコードすると、デコード結果に応じて、プロセッサ 10 の各部が制御される。例えば、デコードの結果がベクトル演算命令であった場合、レジスタファイル 12 のベクトルレジスタのデータがベクトル演算用の演算器 131 に入力され、ベクトル演算が実行される。ベクトル演算用の演算器 131 の演算結果は、セクタ 101 を介して統計情報取得部 102 およびデータ変換部 103 に供給される。また、ベクトル演算用の演算器 131 の演算結果は、アキュムレータ 132 に入力され、ベクトル演算用の演算器 131 の演算結果が例えば、カスケードに加算される。アキュムレータ 132 の演算結果は、セクタ 101 を介して、統計情報取得部 102 およびデータ変換部 103 に供給される。

【0042】

また、例えば、デコードの結果、命令がスカラ演算命令であった場合には、レジスタファイル 12 のスカラ・レジスタのデータがスカラ演算用の演算器 141 に入力される。演算器 141 の演算結果は、アキュムレータ 132 の演算結果と同様、セクタ 101 を介して統計情報取得部 102 およびデータ変換部 103 に供給される。

【0043】

さらに、例えば、デコードの結果、命令がロード命令であった場合には、データ用メモリ 22 からデータが読み出され、セクタ 101 を介して、統計情報取得部 102 およびデータ変換部 103 に供給される。データ変換部 103 でデータ変換された結果は、レジスタファイル 12 のレジスタに格納される。

【0044】

また、デコードの結果、命令がDynamic Fixed Point演算を実行する命令であった場合、デコーダ 112 は、シフト量がデータ変換部 103 に供給されるように指示する。シフト量は、例えば、命令のオペランド（即値）、オペランドで指定されたレジスタ、オペランドで指定されたアドレス・レジスタの示すアドレスのデータ用メモリ 22 等から取得され、データ変換部 103 に供給される。図 7 の処理は、デコーダが命令から、固定小数点数データの小数点位置の指定（シフト量）を取得することの一例である。

【0045】

データ変換部 103 は、ベクトル演算の結果、スカラ演算の結果、アキュムレータ 132 の演算結果、あるいは、データ用メモリ 22 からの読み出し結果等で得られる固定小数点数データを、指定されたシフト量Sだけシフトする。また、データ変換部 103 は、シフトとともに、上位ビットの飽和処理および下位ビットの丸めを実行する。データ変換部 103 は、例えば入力を 40 ビットの演算結果とし、下位Sビットを小数部として丸めを行な丸め処理部と、算術シフトを実行するシフトと飽和処理を行なう飽和処理部を有する。

【0046】

丸め処理部は、下位S bitを小数部として丸めを行なう。Sが負の場合には、丸め処理部は、何も行なわない。丸めとしては、最近接丸め、0への丸め、正の無限大への丸め、負の無限大への丸め、乱数丸めなどが例示される。データ変換部 103 が固定小数点数データをシフトするときのシフト量は、例えば、図 7 に例示されるように、デコーダが命令から取得するシフト量である。

【0047】

シフト部は、Sが正の場合はS bit算術右シフト、Sが負の場合は、算術左シフト、すな

10

20

30

40

50

わち、-S bit算術左シフトを行なう。飽和部は、シフト結果が $2E15-1$ （正の最大値）以上の場合は $2E15$ 、 $-2E15$ （負の最小値）以下の場合は $-2E15$ 、それ以外は入力の下位16bitを出力する。ここで、 $2E15$ は、 2 の 15 乗を表す。

【0048】

そして、データ変換部103は、左シフト時上位ビットの符号を維持し、符号ビット以外を飽和処理、すなわち、上位ビット廃棄し、下位ビットに0を埋め込む。また、データ変換部103は、右シフト時、上位ビット（符号ビットより下位のビット）に符号ビットを埋め込む。そして、データ変換部103は、上記のように丸め、シフト、飽和処理によって得られるデータを、レジスタファイル12のレジスタ等と同一のビット幅（例えば、16ビットのレジスタ）で出力する。データ変換部は固定小数点数データの小数点位置を更新する更新回路の一例である。

10

【0049】

したがって、プロセッサ10で実行されるコンピュータプログラムがDynamic Fixed Point演算を実行する命令のオペランドにシフト量を指定することで、プロセッサ10は、プログラム実行中に、固定小数点数の小数点位置を指定されたシフト量だけ更新する。

【0050】

また、デコードの結果、命令が統計情報取得を指示する命令（統計情報取得機能付き命令という）である場合、統計情報取得部102において、統計情報が取得され統計情報格納部105に格納される。ここで、統計情報は、上記で述べたように、（1）非符号となる最上位ビット位置の分布、（2）非符号となる最下位ビット位置の分布、（3）非符号となる最上位ビット位置の最大値、（4）非符号となる最下位ビット位置の最小値、またはこれらの組み合わせである。

20

【0051】

図8に、図7のプロセッサ10の回路ブロックを例示する。プロセッサ10は、制御ユニット11、レジスタファイル12、ベクトルユニット13、スカラユニット14を含む。制御ユニット11は、プログラムカウンタ111とデコーダ112を含む。レジスタファイルは、ベクトルレジスタファイル、ベクトル演算用アキュムレートレジスタ（Vector ACC）、スカラレジスタファイル、スカラ演算用アキュムレートレジスタ（ACC）を含む。ベクトルユニット13は、ベクトル演算用の演算器131、統計情報取得部102、データ変換部103を含む。スカラユニット14は、スカラ演算用の演算器141、統計情報取得部102、データ変換部103を含む。

30

【0052】

また、図8の構成例では、複数の統計情報取得部102からの統計情報を集約する統計情報集約部104が追加されている。また、統計情報格納部105はレジスタファイル12の一部となっている。また、命令用メモリ21は、メモリインターフェース（Memory I/F）を介して制御ユニット11に接続される。また、データ用メモリ22は、メモリインターフェース（Memory I/F）を介してベクトルユニット13およびスカラユニット14に接続される。

【0053】

図9にベクトルユニット13の詳細を例示する。図では、統計情報集約部104も例示されている。ベクトルユニット13は、ベクトル・レジスタVector Reg0、Vector Reg1のデータをベクトル演算用の演算器131-1で演算する。ベクトル演算用の演算器131-1の演算結果は、積和演算用のアキュムレータ132と、ベクトル演算用の演算器131-2に入力される。

40

【0054】

積和演算用のアキュムレータ132は、ベクトル演算用の演算器131-1の演算結果をスカラ加算し、スカラ演算用アキュムレートレジスタ（ACC）に格納する。ベクトル演算用の演算器131-2は、命令によって指定される演算モードに従って、ベクトル演算用の演算器131-1の演算結果、あるいは、ベクトル演算用アキュムレートレジスタ（Vector ACC）のデータ、あるいは、それらを加算した結果を出力する。

50

【0055】

セクタ101は、ベクトル演算用の演算器131-2の出力結果と、データ用メモリ22からの読み出し結果(Read Data 0, ..., Read Data 0)のいずれかを選択し、統計情報取得部102およびデータ変換部103に入力する。統計情報取得部102で取得された統計情報は、統計情報集約部104に入力される。また、データ変換部103でデータ変換されたデータは、図示しないセクタを介して、データ用メモリ22(Write Data 0, ..., Write Data n)にストアされるか、ベクトルレジスタ(Vector Reg2)に保持される。

【0056】

図10にスカラユニット14の構成を例示する。スカラユニット14は、即値(Immediate)オペランドで得られるデータとスカラ・レジスタScalar Reg1からのデータの一方を選択するセクタ142と、セクタ142の選択結果とスカラ・レジスタScalar Reg0のデータを演算するスカラ用の演算器141を有する。スカラ用の演算器141の演算結果は、メモリインターフェース(Memory I/F)を介してデータ用メモリ22のアドレス(例えば、Addr)にストアされる。また、スカラ用の演算器141の演算結果は、セクタ101を介して、統計情報取得部102およびデータ変換部103に入力される。

【0057】

セクタ101は、スカラ用の演算器141の演算結果、スカラ・レジスタScalar Reg2のデータ、スカラ演算用アキュムレートレジスタ(ACC)のデータ、およびメモリインターフェース(Memory I/F)を介して読み出されたデータ(Read Data)の1つを選択する。セクタ101は、選択したデータを、統計情報取得部102およびデータ変換部103に入力する。統計情報取得部102は、セクタ101から入力されたデータから統計情報を取得し、統計情報集約部104に入力する。

【0058】

以下、統計情報取得部102のうち、非符号となる最上位ビット位置を取得するものを統計情報取得部102Aと呼ぶことにする。また、統計情報取得部102のうち、非符号となる最下位ビット位置を取得するものを統計情報取得部102Bと呼ぶことにする。また、統計情報集約部104のうち、統計情報取得部102によって取得されてビット位置を計数してビット位置に対するビットの分布を取得するものを統計情報集約部104Aと呼ぶことにする。また、統計情報集約部104のうち、ビット位置の最大値と最小値を取得するための前段階として統計情報取得部102によって取得されてビット位置を論理和演算するものを統計情報集約部104Bと呼ぶことにする。

【0059】

図11に非符号最上位ビット位置を検出する統計情報取得部102Aの真理値表を例示する。図11のように、統計情報取得部102Aは、正数に対しては、ビットin[38]から下位ビット方向にビットをサーチして、最初に1になったビット位置を1に他のビット0に設定した2進数のビットパターンを出力する。また、統計情報取得部102Aは、負数に対しては、in[38]から下位ビット方向にビットをサーチして、最初に0になったビット位置を1に他のビット0に設定した2進数のビットパターンを出力する。また、入力データが全ビット0、または、全ビット1の場合は、統計情報取得部102Aは、39bit目に1を、38bit以下に0を出力する。

【0060】

図12に、非符号となる最上位ビット位置を取得する統計情報取得部102Aのハードウェア回路の構成を例示する。この回路では、符号ビットin[39]と他のビット(in[38]からin[0])との排他論理和(EXOR)が実行される。そうすると、符号ビットin[39]と同じ値を有するビットによる排他論理和値は0となり、符号ビットin[39]と異なる値を有するビットによる排他論理和値は1となる。

【0061】

今、仮にin[38]がin[39]と異なる値であった場合、排他論理和によって出力データのout[38]は1となる。一方、出力データのout[37]には、in[39]とin[37]の排他論理和値がAN

10

20

30

40

50

Dゲートを介して入力される。このANDゲートの一方の入力には、in[39]とin[38]の排他論理和値を反転したビット値が入力されるため、in[39]とin[38]とが不一致の場合、in[39]とin[37]の排他論理和値によらず、ANDゲートの出力は0となる。

【0062】

以下、同様に、out[i](iは37以下)が出力されるANDゲートの論理否定付き入力、in[39]とin[j](jはi+1以上、38以下)の排他的論理和が全て0の場合に0となり、in[39]とin[i](iは37以下)の排他論理和値が1になると、out[i]には1が設定される。そのビット位置(i)より下位のビットでは、out[i]が出力されるANDゲートの論理否定付き入力が1となるため、out[i]には0が設定される。したがって、図12の回路によって、非符号となる最上位ビット位置に1が設定され、他のビットが0に設定された出力データout(40bit)が取得される。図11、図12の非符号となる最上位ビット位置を取得する統計情報取得部102Aは、命令実行後の固定小数点数データ中の符号ビットと一致しない最上位のビットの位置を取得する回路の一例である。また、非符号となる最上位ビット位置に1が設定され、他のビットが0に設定された出力データout(40bit)は、符号ビットと一致しない最上位のビットの位置を真値(1)で示すビット列の一例である。

【0063】

図13に非符号となる最下位ビット位置を検出する統計情報取得部102Bの真理値表を例示する。図13のように、統計情報取得部102Bは、正数に対しては、in[0]から上位ビット方向にビットをサーチして、最初に1になったビット位置を1に他のビット0に設定した2進数のビットパターンを出力する。また、統計情報取得部102Bは、負数に対しては、in[0]から上位ビット方向にビットをサーチして、最初に0になったビット位置を1に他のビット0に設定した2進数のビットパターンを出力する。また、入力データが全ビット0、または、全ビット1の場合には、統計情報取得部102Bは、39bit目に1を、38bit以下に0を出力する。

【0064】

図14に、非符号となる最下位ビット位置を取得する統計情報取得部102Bのハードウェア回路の構成を例示する。符号ビットin[39]が0の場合には、統計情報取得部102Bは、最下位ビットin[0]から上位側に向かって、ビットが1であるビット位置を探索すればよい。一方、符号ビットin[39]が1の場合には、データは補数となっているので、統計情報取得部102Bは、最下位ビットin[0]から上位側に向かって、ビットが0であるビット位置を探索すればよい。

【0065】

すなわち、この回路では、符号ビットin[39]と他のビット(in[0]からin[38])との排他論理和(EXOR)が実行される。そうすると、符号ビットin[39]と同じ値を有するビットによる排他論理和値は0となり、符号ビットin[39]と異なる値を有するビットによる排他論理和値は1となる。

【0066】

今、仮にin[0]がin[39]と異なる値であった場合、排他論理和によって出力データのout[0]は1となる。一方、出力データのout[1]には、in[39]とin[1]の排他論理和値がANDゲートを介して入力される。このANDゲートの一方の入力には、in[39]とin[0]の排他論理和値を反転したビット値が入力される。このため、in[39]とin[0]の排他論理和値が1の場合、in[39]とin[1]の排他論理和値によらず、ANDゲートの出力は0となる。

【0067】

以下、同様に、out[i](iは1以上)が出力されるANDゲートの論理否定付き入力、in[39]とin[j](jは0以上、i-1以下)の排他的論理和が全て0の場合に0となり、in[39]とin[i](iは1以上)の排他論理和値が1になると、out[i]には1が設定される。また、そのビットより上位の出力データout[i]には0が設定される。したがって、図14の回路によって、非符号となる最下位ビット位置に1が設定され、他のビットが0の出力データout(40bit)が取得される。図13、図14の非符号となる最下位ビット位置を取得する統計情報取得部102Bは、符号ビットと一致しない最下位のビットの位置を取得する回路の一例であ

る。また、非符号となる最下位ビット位置に1が設定され、他のビットが0に設定された出力データout(40bit)は、符号ビットと一致しない最下位のビットの位置を真値(1)で示すビット列の一例である。

【0068】

図15は、統計情報取得部102が取得したデータからビットの分布を取得する統計情報集約部104Aの処理を例示する図である。図では、40ビットのデータが8個並列に処理されるSIMDデータからビットの分布を取得する処理が例示される。図15では、ハードウェア回路である統計情報集約部104Aの処理が、擬似コードで記述されている。

【0069】

すなわち、入力データは、8(行)×40(ビット)の配列データで例示されている。また、各行の40ビットの入力データは、非符号となる最上位ビット位置(図12の統計情報取得部102Aの出力)または非符号となる最下位ビット位置(図14の統計情報取得部102Bの出力)である。この処理では、40ビットの出力データoutについて、まず、全ビットがクリアされる。次に、入力データの配列in[j][i]の各列iの要素の値が、すべての行(j=0から7)に対して加算される。したがって、図11、図13とは異なり、図15の擬似コードでは、出力データ(配列要素)out[j]は、 $\log_2(\text{SIMDデータ数})$ ビット(図15の例では3ビット)の整数である。なお、図17では、SIMDデータ数(並列処理されるデータ数)は8であると想定したが、SIMDデータ数が8に限定される訳ではない。

【0070】

図16に、統計情報取得部102が取得したデータからビットの分布を取得する統計情報集約部104Aのハードウェア回路の構成を例示する。統計情報取得部102が取得したデータ(ここでは、統計取得0から統計取得(SIMDデータ数-1))がbit population count演算によって、8個の統計情報のiビット目(i=0から39)における1の個数がそれぞれカウントされる。入力データは、統計情報取得部102A(図11、図12)によって取得された非符号となる最上位ビット位置である。したがって、統計情報集約部104Aは、統計情報取得部102Aによって取得された非符号となるSIMDデータ個数分の最上位ビット位置に対して、各ビットの'1'の発生回数をカウントすることで最上位ビット位置の発生回数を計数する。統計情報集約部104Aは、計数結果を出力データout0からout39にそれぞれ格納する。

【0071】

また、入力データは、統計情報取得部102B(図13、図14)による非符号となる最下位ビット位置とすることもできる。統計情報集約部104Aは、統計情報取得部102Bによって取得されたSIMDデータ個数分の非符号となる最下位ビット位置に対して、各ビットの'1'の発生回数をカウントすることで最下位ビット位置の発生回数を計数する。統計情報集約部104Aは、計数結果を出力データout0からout39にそれぞれ格納する。すなわち、統計情報集約部104Aは、非符号となる最上位ビット位置または非符号となる最下位ビット位置のいずれをも処理可能である。

【0072】

また、図16でセレクタ(SEL)は、bit population count演算器(Σ)と、スカラユニット14から取得されるデータを選択する。セレクタ(SEL)に選択されたデータが、出力データout0からout39に出力される。したがって、統計情報取得部102がスカラユニット14で取得したデータは、一回のスカラユニット14の演算においては、加算されることなくそのまま出力データout0からout39に出力される。out0からout39は、統計情報格納部105に引き渡すデータである。図15、図16の統計情報集約部104Aは符号ビットと一致しない最上位のビットの位置を複数の固定小数点数データに対して累積して計数する回路の一例である。図15、図16の統計情報集約部104Aは符号ビットと一致しない最下位のビットの位置を複数の固定小数点数データに対して累積して計数する回路の一例でもある。

10

20

30

40

50

【0073】

図17は、統計情報取得部102が取得したデータからビット位置の最大値と最小値を取得する前提として、論理和演算によってビット位置を集約する統計情報集約部104Bの処理を例示する図である。図17においても、図15と同様、40ビットのデータが8個並列に処理されるSIMDデータを処理が例示される。図17では、ハードウェア回路である統計情報集約部104Bの処理が、擬似コードで記述されている。

【0074】

この処理では、40ビットの出力データ $out[i]$ ($i=0, \dots, 39$)には、入力データの配列 $in[j][i]$ の各列をすべての行 ($j=0, \dots, 7$) について論理和演算 (OR演算) した結果が入力される。したがって、図17の擬似コードでは、図15とは異なり、出力データ (配列要素) $out[i]$ ($i=0, \dots, 39$) はビット列である。以上の処理の結果、出力データ $out[i]$ ($i=0, \dots, 39$) では、 $out[38]$ から下位ビット方向に向かって最初に1となるビット位置が最大ビット位置である。また、 $out[0]$ から上位ビット方向に向かって最初に1となるビット位置が最小ビット位置である。

【0075】

図18に、統計情報取得部102が取得したデータからビット位置の最大値と最小値を取得する前提として、論理和演算によってビット位置を集約する統計情報集約部104Bのハードウェア回路の構成を例示する。統計情報取得部102が取得したデータ (ここでは、統計取得0から統計取得SIMDデータ数-1) がORゲート (40ビット) によってOR演算される。また、図18でセクタ (SEL) は、論理和演算 (OR) と、スカラユニット14から取得されるデータを選択する。セクタ (SEL) されたデータが、出力データ out に出力される。したがって、統計情報取得部102がスカラユニット14で取得したデータは、一回の演算においては、論理和演算されることなくそのまま出力データ out に出力される。 out は、統計情報格納部105に引き渡すデータである。

【0076】

論理和演算によってビット位置を集約する統計情報集約部104Bは、符号ビットと一致しない最上位のビットの位置を真値で示すビット列を複数の固定小数点数データについて論理和演算によって累積する回路の一例である。論理和演算によってビット位置を集約する統計情報集約部104Bは、符号ビットと一致しない最下位のビットの位置を真値で示すビット列を複数の固定小数点数データについて論理和演算によって累積する回路の一例でもある。

【0077】

図19は、統計情報集約部104の具体的な構成例であり、統計情報の格納先がデコーダ112からのインデックスで指定される回路の構成例である。図では、例えば、 $sr[j][i]$ ($j=0, \dots, k, i=0, \dots, 39$) の領域が確保され、 $index$ によって、レジスタファイルの行 j が指定される。

【0078】

プロセッサ10は、書き込み命令によってセクタSELを介して、 $index$ によって指定されるレジスタファイルの行 j の1つあるいは複数のレジスタに初期値を書き込む。ただし、プロセッサ10はデコーダ112からの制御信号により、 $index$ によって指定されるレジスタファイルの行 j をリセットしても良い。そして、プロセッサ10は、 $in39$ から $in0$ の統計情報を加算器で蓄積し、 $index$ によって指定されるレジスタファイルの行 j に格納する。また、プロセッサ10は、デコーダ112からの制御信号により、 $index$ によって指定されるレジスタファイルの行 j から統計情報を読み出す。また、プロセッサ10は、 $index$ によって指定されるレジスタファイルの行 j のいずれかの1つあるいは複数の値を読み出し、読み出し命令で指定されたデータ用メモリに保存する、あるいは、読み出し命令で指定された汎用レジスタに格納する。

【0079】

図20は、図17、図18で例示した統計情報集約部104Bによって論理和演算されたビット位置の統計情報を蓄積するとともに蓄積された統計情報のビット位置の最大値を

読み出す統計情報格納部 105C のハードウェア回路の構成を例示する図である。統計情報格納部 105C は、統計情報集約部 104B によって論理和演算されたビット位置の統計情報を蓄積するレジスタ (sr) を有している。プロセッサ 10 は、書き込み命令 (write) によってセクタ (SEL) を介してレジスタ (sr) に初期値を書き込み可能である。ただし、プロセッサ 10 は、リセット信号によってレジスタ (sr) をリセットできるようにしてもよい。

【0080】

また、統計情報格納部 105C は、統計情報集約部 104B の論理和演算結果 (in) と、すでにレジスタ (sr) に蓄積された統計情報の論理和演算を実行し、セクタ (SEL) を介してレジスタ (sr) に論理和演算の結果を蓄積する。

10

【0081】

また、プロセッサ 10 は、読み出し命令によってレジスタ (sr) の値を読み出し、読み出し命令で指定されたデータ用メモリあるいは読み出し命令で指定された汎用レジスタに保存する。また、統計情報格納部 105C は、Priority Encoder(MSB優先)を有してもよい。Priority Encoder(MSB優先)は、レジスタ (sr) に蓄積されたビット列中の最上位のビット 1 の位置 (−1 から 38) を 2 進数で出力する。例えば、入力データ in として全ビット 0 が入力されると、Priority Encoder(MSB優先)は、"111111" (−1) を出力する。また、入力データ in として in0=1、他のビットがすべて 0 のデータが入力されると、Priority Encoder(MSB優先)は、"000000" (0) を出力する。また、入力データ in として in0=x (0 または 1)、in1=1、他のビットがすべて 0 のデータが入力されると、Priority Encoder(MSB優先)は、"000001" (1) を出力する。同様に、入力データ in として in0~in37 が x (0 または 1)、in38=1 のデータが入力されると、Priority Encoder(MSB優先)は、"100110" (38) を出力する。プロセッサ 10 は、Priority Encoder(MSB優先)を介して、統計情報集約部 104B によって論理和演算されたビット位置の統計情報からビット位置の最大値を 2 進数値で取得できる。図 17、図 18 の論理和演算によってビット位置を集約する統計情報集約部 104B (図 17、図 18) と統計情報格納部 105C の組み合わせは、符号ビットと一致しない最上位のビットの位置を真値で示すビット列を複数の固定小数点数データについて論理和演算によって累積し、累積したビット列中の最上位の真値の位置を取得する回路の一例である。

20

【0082】

30

図 21 は、図 17、図 18 で例示した統計情報集約部 104B によって論理和演算されたビット位置の統計情報を蓄積するとともに蓄積された統計情報のビット位置の最小値を読み出す統計情報格納部 105D のハードウェア回路の構成を例示する図である。統計情報格納部 105D は、統計情報格納部 105C の Priority Encoder(MSB優先)に代えて Priority Encoder(LSB優先)を有している。Priority Encoder(LSB優先)以外の統計情報格納部 105D の構成は、統計情報格納部 105C と同様であるから説明を省略する。

【0083】

Priority Encoder(MSB優先)は、レジスタ (sr) に蓄積されたビット列中の最下位のビット 1 の位置 (−1 から 38) を 2 進数で出力する。例えば、入力データ in として全ビット 0 が入力されると、Priority Encoder(LSB優先)は、"111111" (−1) を出力する。また、入力データ in として in0=1、他のビットが x (0 または 1) のデータが入力されると、Priority Encoder(LSB優先)は、"000000" (0) を出力する。また、入力データ in として in0=0、in1=1、他のビット (in2 から in38) が x (0 または 1) のデータが入力されると、Priority Encoder(LSB優先)は、"000001" (1) を出力する。同様に、入力データ in として in0~in37 が 0、in38=1 のデータが入力されると、Priority Encoder(LSB優先)は、"100110" (38) を出力する。プロセッサ 10 は、Priority Encoder(LSB優先)を介して、統計情報集約部 104B によって論理和演算されたビット位置の統計情報からビット位置の最小値を 2 進数値で取得できる。論理和演算によってビット位置を集約する統計情報集約部 104B (図 17、図 18) と統計情報格納部 105D (図 21) の組み合わせは、符号ビットと一致しない最下位のビットの位置を真値で示すビット列を複数の固定小数点数データ

40

50

について論理和演算によって累積し、累積したビット列中の最下位の真値の位置を取得する回路の一例である。

＜実施形態 1 の効果＞

【0084】

情報処理装置 1 は、深層学習のミニバッチ実行時、各レイヤの各変数の統計情報をレジスタあるいはレジスタファイルに累積する。そして、情報処理装置 1 は、蓄積された統計情報を基に各レイヤの各変数の小数点位置を更新することができる。すなわち、プロセッサ 10 が、ビット分布の統計上情報を取得する。ここで、統計情報は、命令実行時の（１）非符号となる最上位ビット位置の分布、（２）非符号となる最下位ビット位置の分布、（３）非符号となる最上位ビット位置の最大値、（４）非符号となる最下位ビット位置の最小値、またはこれらの組み合わせ等である。したがって、情報処理装置 1 が深層学習実行時に、データの統計情報取得のための深層学習プログラム中のオーバーヘッドがなくなるため、実用的な時間で動的固定小数点演算を実現可能となる。

【0085】

すなわち、本実施形態では、情報処理装置 1 のプロセッサ 10 が統計情報取得機能付き命令を実行するとともに、演算結果をビットシフトし、丸め・飽和を行なってレジスタに格納する命令を実行する。したがって、情報処理装置 1 はビット分布を示す統計情報を取得するオーバーヘッドを少なくできる。また、ビット分布を示す統計情報から直ちに適正なビットシフト、すなわち、小数点位置を決定できる。つまり、情報処理装置 1 のように、小数点位置を 1 ビットずつずらして適正な小数点位置を試行し、次の演算で結果を確認するという手順ではなく、ビット分布を示す統計情報から、直ちに小数点位置を決定できる。したがって、情報処理装置 1 は、小数点位置が不適切な状態で学習処理が繰り返される恐れがすくない。また、情報処理装置 1 は、学習精度の劣化につながり、深層学習の収束が遅くなる恐れが少ない。

【0086】

また、プロセッサ 10 は、統計情報取得部 102 A によって命令実行後の固定小数点数データ中の符号ビットと一致しない最上位のビットの位置を取得できる。また、プロセッサ 10 は、統計情報集約部 104 A によって符号ビットと一致しない最上位のビットの位置を複数の固定小数点数データに対して累積して計数することができる。また、プロセッサ 10 は、符号ビットと一致しない最上位のビットの位置を真値で示すビット列を複数の固定小数点数データについて論理和演算によって累積する。そして、プロセッサ 10 は、累積したビット列中の最上位の真値の位置を取得することができる。

【0087】

また、プロセッサ 10 は、統計情報取得部 102 B によって命令実行後の固定小数点数データ中の符号ビットと一致しない最下位のビットの位置を取得できる。また、プロセッサ 10 は、統計情報集約部 104 A によって符号ビットと一致しない最下位のビットの位置を複数の固定小数点数データに対して累積して計数することができる。また、プロセッサ 10 は、符号ビットと一致しない最下位のビットの位置を真値で示すビット列を複数の固定小数点数データについて論理和演算によって累積する。そして、プロセッサ 10 は、累積したビット列中の最下位の真値の位置を取得することができる。以上のような構成により、プロセッサ 10 は、上記統計情報を取得できる。

〔実施形態 2〕

【0088】

以下、図 22 から図 46 を参照し、実施形態 2 に係る情報処理システムおよびホストマシン 3 について説明する。上記実施形態 1 では、統計情報取得部 102（102 A、102 B）により統計情報を取得し、オーバーフロー回数等が所定の限度となるように固定小数点数データの小数点位置を調整することで、固定小数点数表現で深層学習を効率良く実行できるプロセッサ 10 について説明した。なお、統計情報は、演算結果の値の分布を含むものと言える。

＜実施形態 1 における課題＞

【0089】

実施形態1の構成では、以下のような課題が想定される。実施形態1のプロセッサ10の構成では、各演算器が統計情報を取得するため、演算器の数が多いと回路規模が増大する。逆にプロセッサ10において、統計情報取得機能を持つ演算器が限定できれば、回路規模の増加は抑えられる。しかし、統計情報取得機能を持つ演算器が限定されると、統計情報サンプル数が不足するため、プロセッサ10で実行されるアプリケーションプログラムが適切でない小数点位置を推定してしまう場合も想定される。適切でない小数点位置を推定した場合には、演算結果が望ましくないものとなり、例えば、深層学習の認識精度あるいは正答率等が想定範囲内に収まらない場合が増加することも生じ得る。特に、8ビット固定小数点表現など、演算ビット幅が小さい場合において想定通りに計算できないケースが増加することが予想される。したがって、回路規模の増大を抑えつつ、想定通りの計算を継続させるための手法が望まれる。

10

【0090】

図22は、VGG16という画像分類用の学習ネットワークを簡略化したVGG8というネットワークで10000回学習を行った結果の例である。この例では、16ビットの動的固定小数点数による演算を用いた方式では10000回の学習において、認識精度は40%程度になる(図でDLINT-16参照)。一方、8ビット動的固定小数点数による演算を用いた方式では、10000回の学習において、認識精度が5%に低下してしまう(図でDLINT-8参照)。

20

【0091】

深層学習では、変数(値の種類)によって変数値の生成回数(演算器による演算回数)が異なる。変数値の生成回数が少ないと小数点位置に関する統計情報の取得機会が少なくなり、正しく小数点位置を推定できなくなることが想定される。例えば、多層ニューラルネットワークの各層のノード(ニューロンともいう)に入力されるパラメータ値の計算および各ニューロンでの重みの計算では、計算回数が多く、固定小数点数の整数部桁数についての統計情報の取得機会、したがってサンプル数は多い。一方、各ニューロンにおける定数の計算および活性化関数等による分類のための確率計算は計算回数が少なく、固定小数点数の整数部桁数についての統計情報のサンプル数は少ない。

【0092】

そこで、実施形態2では、統計情報を利用するアプリケーションプログラム(以下、単にプログラム)に以下のような改善が追加される。すなわち、実施形態2では、統計情報を利用するプログラムは、統計情報の取得機会が多い変数についてはビット幅を少なくし、統計情報の取得機会が少ない変数についてはビット幅を多めに確保する。例えば、プログラムは、統計情報の取得機会が少ない変数については浮動小数点数表現を利用するようにする。統計情報の取得機会の多寡は、統計情報の取得回数の多寡、したがって、統計情報のサンプル数の多寡と考えて良い。すなわち、実施形態2の情報処理システムおよびホストマシン3は、統計情報取得回数の多寡に応じて変数のデータ型を決定し、決定されたデータ型にしたがった変数が組み込まれたプログラムを生成する。

30

【0093】

実施形態2の深層学習では、予めニューラルネットワークの構成を規定する情報が定義されているものとする。本情報処理システムおよびホストマシン3は、予め定義されたニューラルネットワークの構成を規定する情報(以下、ニューラルネットワーク記述情報)に基づいて実行するプログラムを生成する。例えば、ニューラルネットワーク記述情報としては、caffeと呼ばれるソフトウェアライブラリ(フレームワークともいう)におけるprototextファイル、tensorflowと呼ばれるソフトウェアライブラリにおけるグラフ表現、chainerと呼ばれるソフトウェアライブラリにおけるpythonスクリプト等が知られている。ニューラルネットワーク記述情報は、ニューラルネットワークを定義する定義情報の一例である。

40

【0094】

変数値の生成回数はニューラルネットワーク記述情報から取得可能である。また、統計

50

情報取得機能における間引き情報から統計情報の取得回数を得られる。間引き情報とは、統計情報取得機能において、変数値の生成N回あたり1回統計情報を取得する、という場合のNをいう。間引き情報はハードウェアの構成から決定可能である。したがって、変数値の生成回数と間引き情報とから統計情報の取得回数を計算することが可能となる。

実施形態2の情報処理システムおよびホストマシン3は、動的固定小数点数演算および固定小数点数演算の機能を持つプロセッサで実行可能な深層学習のプログラムを生成する。プログラムの生成において、本情報処理システムおよびホストマシン3は、ニューラルネットワーク記述情報に定義された深層学習のネットワーク構造を参照し、統計情報の取得回数が少ない（例：30回以下の）と判断される変数については固定小数点のビット幅を増加または浮動小数点数を使用するようにデータの型を決定する。一方、本情報処理システムおよびホストマシン3は、統計情報の取得回数が多い判断される変数については固定小数点のビット幅を削減するようにデータ型を決定する。

10

【0095】

以上のように、実施形態2の情報処理システムおよびホストマシン3は、ニューラルネットワーク記述情報に基づき統計情報の取得回数を変数毎に解析し、統計情報の取得回数の多寡に応じて変数のデータ型を決定する点で、実施形態1の情報処理装置1と相違する。したがって、実施形態2の情報処理システムおよびホストマシン3は、実行するプログラムの処理が実施形態1と異なるものの、他の構成要素および作用は実施形態1と同様である。そこで、実施形態2の構成要素のうち、実施形態1の情報処理装置1の構成が適用される構成要素には実施形態1と同一の符合を付して参照し、その説明を省略する。

20

<構成>

【0096】

図23に、実施形態2に係る情報処理システムの構成を例示する。実施形態2では、ホストマシン3とDL実行ハードウェア1Aが専用インターフェース上で接続しているものを例にとり、情報処理システムを説明する。利用者はホストマシン3に接続し、ホストマシン3がDL実行ハードウェア1Aを制御し、深層学習（以下、単にDL（Deep Learning）という）を実行する。ホストマシン3は利用者からの指示に従い、DL実行ハードウェア1Aが実行するプログラムを作成し、DL実行ハードウェア1Aに送信する。DL実行ハードウェア1Aは送信されたプログラムを実行し、結果のデータを生成する。図23におけるホストマシン3は、演算処理部の一例であるDL実行ハードウェア1A（およびDL実行プロセッサ10A）とは異なるプロセッサの一例である。実施形態2において、ホストマシン3は、プログラムを生成する生成部としての処理を実行する。

30

【0097】

図24は、ホストマシン3がDL実行ハードウェア1Aを用いて深層学習を実行するときのモジュールの構成を例示する図である。図のように、ホストマシン3は、アプリケーション51、ライブラリ52、ユーザモードドライバ53、およびカーネルモードドライバ54を有する。アプリケーション51は、利用者とのユーザインターフェースを実装する。また、アプリケーション51は、利用者からDLの設計情報を受け取り、結果を表示する。

【0098】

ホストマシン3は、ライブラリ52の機能を使ってDL実行機能を実装している。ライブラリ52は、ホストマシン3におけるアプリケーション51の実装を補佐または支援する。すなわち、DLの実行に関する機能はライブラリ52で提供される。

40

【0099】

ユーザモードドライバ53は、通常はライブラリ52から呼び出される。ただし、場合によっては、ユーザモードドライバ53は、アプリケーション51から直接呼び出される。

【0100】

ユーザモードドライバ53は、DL実行ハードウェア1Aのためのプログラムコードを作成する。ユーザモードドライバ53は、Operating System（OS）のユーザモードで実

50

行されるドライバである。

【0101】

カーネルモードドライバ54は、ユーザモードドライバから呼び出され、DL実行ハードウェア1Aと通信する。すなわち、DL実行ハードウェア1Aとの通信処理は、ハードウェアに直接アクセスするものであるため、カーネルモードドライバ54として実装される。カーネルモードドライバ54は、OSのカーネルモードで実行されるドライバである。

【0102】

図25は、ホストマシン3の構成を例示する。ホストマシン3は、プロセッサ31、高速入出力インターフェース34、ランダムアクセスメモリ(RAM32)、内部バス35、外部記憶装置としてのハードディスクドライブ(HDD33)、低速入出力インターフェース39等を有する。ホストマシン3は、プログラムを生成する情報処理装置の一例である。

10

【0103】

プロセッサ31はRAM32上に配置されているプログラムを実行する。高速入出力インターフェース34はプロセッサ31とプロセッサ31外部にあるDL実行ハードウェア1Aとを接続する。高速入出力インターフェース34の一例としてPeripheral Component Interconnect(PCI) Expressなどが挙げられる。RAM32はプロセッサ31が実行するプログラムやデータを配置する。RAM32の形式としては、DDR4-SDRAM等が例示される。内部バス35はプロセッサ31と比較して低速な周辺機器とプロセッサ31とを接続して通信を中継する。HDD33は不揮発性の補助記憶装置として動作し、プロセッサ31が実行するプログラムやデータを永続的に格納する。低速入出力インターフェース39は利用者との通信を行う。低速入出力インターフェース39は一例として、Universal Serial Bus(USB)によるキーボードあるいはマウスとの接続部が例示される。また、低速入出力インターフェース39としては、Ethernet等によるネットワークを介した通信を実行する通信部も例示できる。

20

【0104】

図26にDL実行ハードウェア1Aの構成を例示する。DL実行ハードウェア1Aは、DL実行プロセッサ10A、ハードウェア(HW)制御部1A1、高速入出力インターフェース1A4、メモリアクセスコントローラ1A3、および内部RAM1A2等を有する。

30

【0105】

DL実行プロセッサ10Aは、ホストマシン3から与えられたプログラム、データを基に深層学習の処理を実行する。DL実行プロセッサ10Aの一例は、例えば、実施形態1のプロセッサ10である。DL実行プロセッサ10Aは、演算処理部の一例である。HW制御部1A1はホストマシン3からの指令を基に、DL実行プロセッサ10Aを駆動させる。また、HW制御部1A1はホストマシン3から指令を受け、内部RAM1A2にプログラムやデータを転送する。高速入出力インターフェース1A4はホストマシン3と接続する。高速入出力インターフェース1A4の具体的なプロトコルとしては先述のPCI Express等が例示される。メモリアクセスコントローラ1A3はDL実行プロセッサ10AやHW制御部1A1からの信号を選択し、メモリアクセスのためのプロトコルにしたがって内部RAM1A2にアクセスする。内部RAM1A2はDL実行プロセッサ10Aが実行するプログラムや処理対象のデータ、処理結果のデータを格納する。内部RAM1A2のとしてはDouble-Data-Rate(DDR)4-Synchronous Dynamic Random Access Memory(SDRAM)、あるいはより高速なGraphics Double Data Rate(GDDR)5やより高帯域のHigh Bandwidth Memory(HBM)2等が挙げられる。

40

【0106】

図27に実施形態2のDL実行プロセッサ10Aの構成を例示する。図27には、メモリアクセスを介してDL実行プロセッサ10Aに接続される命令用メモリ21、およびデータ用メモリ22も記載されている。このうち、命令用メモリ21およびデータ

50

用メモリ 22 の構成は、実施形態 1 のプロセッサ 10 と同様であるので、その説明を省略する。

【0107】

DL 実行プロセッサ 10 A は、制御ユニット 11、レジスタファイル 12、ベクトルユニット 13 A、スカラーユニット 14 を含む。このうち、制御ユニット 11 は実施形態 1 と同様であるので、その説明を省略する。

【0108】

ベクトルユニット 13 A は、ベクトル演算用の固定小数点数演算器 13 B（図で INT と表記）、ベクトル演算用の浮動小数点数演算器 13 C（図で FP と表記）、統計情報取得部 102（図には「統計取得」として省略）、データ変換部 103（図には「データ変換」として省略）およびセレクタ 107 の組を複数組有している。したがって、DL 実行プロセッサ 10 A はベクトルユニット 13 A により固定小数点数の演算および浮動小数点数の演算の少なくとも一方を実行する。なお、上記の複数組は、例えば、8 組であるが、ベクトルユニット 13 A 内の演算器の組数が 8 組に限定される訳ではない。また、図では、一例として、1 組の固定小数点数演算器 13 B、浮動小数点数演算器 13 C、データ変換部 103、セレクタ 107 に符号が付されている。ベクトルユニット 13 A は、複数組の演算器によって、演算を並列に実行する。このうち、データ変換部 103 の構成は、実施形態 1 と同様であるので、その説明を省略する。

【0109】

例えば、ベクトル演算用の固定小数点数演算器 13 B は、上記の複数組用意され、並列に固定小数点数演算、例えば加算を実行する。ベクトル演算用の浮動小数点数演算器 13 C も、上記の複数組用意され、並列に浮動小数点数演算、例えば加算を実行する。以上のように、ベクトルユニット 13 A は、ベクトル演算用の浮動小数点数演算器 13 C が追加された点を除いて、実施形態 1 のベクトルユニット 13 と同様である。以下、図 27 の構成をさらに具体的に説明するとともに、DL 実行プロセッサ 10 A の演算について説明する。

（レジスタ）

【0110】

図 27 で、Scalar Register File は、32 ビットのレジスタを例えば 32 個有する。Scalar Register File は、アドレス、またはパラメータを格納する。また、図 27 で「ACC」はスカラーアキュムレートレジスタである。

【0111】

Vector Register File は、32 ビットベクトルレジスタを例えば 8 セット有する。8 セットの 32 ビットベクトルレジスタは、並列にアクセス可能なレジスタ要素を 8 要素有する。ただし、Vector Register File は、16 ビットベクトルレジスタを例えば 16 要素 8 セット有する構成として利用できる。さらに、Vector Register File は、8 ビットベクトルレジスタを例えば 32 要素 8 セット有する構成として利用できる。

【0112】

Vector Register File は、アセンブリ言語のオペランドとしては、simd0 から simd で表記され、演算データを格納する。以下、レジスタの構成が N ビットで、並列にアクセス可能な要素数 M、セット数 S のレジスタファイルを N ビット x M 要素 x S セットと表記することにする。

（Vector Acc）

【0113】

図 27 で、「Vector Acc」は、ベクトル演算用アキュムレートレジスタを示す。Vector Acc は、80 ビット x 8 要素 x 1 セット（32 ビット演算モード）、40 ビット x 16 要素 x 1 セット（16 ビット演算モード）、20 ビット x 32 要素 x 1 セット（8 ビット演算モード）で使用可能である。Vector Acc は、アセンブリ言語のオペランドとしては、例えば、acc で表記され、ベクトルレジスタ同士の乗算結果やそれらの加算結果を格納する。

(スカラ演算の種類)

プロセッサ10Aが実行可能なスカラ演算としては、四則演算(整数、浮動小数点)、シフト、分岐、ロード、ストア等が例示される。

(ベクトル演算(SIMD演算))

【0114】

プロセッサ10Aが実行可能なベクトル演算は、以下の通りである。まず、単精度浮動小数点演算としては、加減算および乗算等の演算、ロード、ストア、固定小数点表現への変換(指数部、仮数部の取得)が例示される。また、Vector Accを用いた演算(単精度浮動小数点表現)としては、アキュムレートレジスタのクリア、MAC(Multiply and Accumulate)演算が例示される。MAC演算では、加算および出力のオペランドには暗黙的にVector Accが指定される。さらに、整数演算としては、加減算、乗算、シフト演算、浮動小数点表現への生成(指数部、仮数部の設定)が例示される。

【0115】

MAC(Multiply and Accumulate)演算の処理は以下の通りである。演算は、2つの変数(レジスタ値)の積をaccに積算する処理であり、以下の式で例示される。

【0116】

$acc = 16bit \times 16bit + acc$ (16bit mode) または

$acc = 8bit \times 8bit + acc$ (8bit mode)

演算では、2つのVector Registerが指定される。また、加算および出力のオペランドには暗黙的にVector Accが指定される。

累積加算の処理は以下の通りである。演算は、1つの変数(レジスタ値)をaccに積算する処理であり、以下の式で例示される。

【0117】

$acc = 16bit + acc$ (16bit mode) または

$acc = 16bit + acc$ (8bit mode) または

$acc = 8bit + acc$ (8bit mode)

【0118】

演算では、1つのVector Registerと1つのScalar Registerが指定される。ただし、 $16bit + acc$ (8bit mode) の場合は2つのVector Registerが読み込まれる。DL実行プロセッサ10Aは、16要素の16bit Vector Registerを有しており、accは8bit modeでは、32要素利用可能であるため、2つの16bit Vector Registerの処理が可能となるのである。また、Scalar Registerは入力オペランドのシフト量が指定される。

【0119】

Vector Registerへの転送命令では、1つのVector Register、1つのScalar Registerが指定される。Scalar RegisterにはVector Accのシフト量が指定される。

(ロード命令)

【0120】

ロード命令では、オペランドに、読み込みアドレス、ストライド、型変換、書き込み先Vector Registerが指定される。ストライド=0ならば読み込みアドレスの1要素をすべてのVector Registerの要素に格納することを意味する。また、ストライド=1は読み込みアドレスから1要素ずつ読みこんで、Vector Registerの各要素に格納する。ストライド=N(任意の整数)は読み込みアドレス+エレメント幅×N×エレメント番号のアドレスで指定される要素を順番に読み込んでVector Registerの各エレメントに格納する。型変換の命令には読み込み元の型(8bit整数 or 16bit整数)とレジスタに格納する型(8bit整数 or 16bit整数)が指定される。

(ストア命令)

書き込みアドレス、ストライド、読み込み元Vector Registerが指定される。ストライドに関してはロードと同様である。

(統計情報取得部の処理)

【0121】

10

20

30

40

50

統計情報取得部 102 の構成は、実施形態 1 の統計情報取得部 102 (102A、102B) と同様である。すなわち、統計情報取得部 102 は、アキュムレートレジスタに格納されている各要素を解釈し、非符号となる最上位ビットの位置を取得して、その分布を統計情報集約部 104 に累積させる。なお、実施形態 2 において、LSB の位置を 0 として、ビット位置が定義される。

【0122】

統計情報は、複数系統格納可能で、系統は命令のオペランドに即値で指定される。例えば、演算結果が 0x00_1000_0000 という値の場合、符号ビットが 0 で、符号ビットと異なる値 1 が現れる最上位ビット位置は 28 となる。また、例えば、演算結果が 0xFE_FFFF_FFFF という値の場合、符号ビットが 1 で、符号ビットと異なる値 0 が現れる最上位ビット位置は 32 となる。また、複数系統とは、統計情報の複数の格納先という意味である。複数の格納先があるため、例えば、第 1 の統計情報を第 1 の格納先に、第 2 の統計情報を第 2 の格納先に格納するようにオペランドの指定が可能となる。

＜プログラム生成処理＞

【0123】

実施形態 2 のホストマシン 3 は、ニューラルネットワークの計算に登場する処理のうち、積和演算と総和演算とを対象に変数の型を決定し、プログラム生成を行う。ホストマシン 3 は、プログラム生成を行うために、それぞれの処理のテンプレートを用意しておき、与えられたパラメータに従ってテンプレートの内容を修正することで、目的のプログラム (コードともいう) を生成する。

【0124】

図 28 に、プログラム生成に係るホストマシン 3 の論理構成を例示する。ホストマシン 3 は、RAM 32 に実行可能に配置されたコンピュータプログラムにより図 28 の処理を実行する。図のように、ホストマシン 3 は、論理構成としては、変数ごとの生成回数取得処理部 311、変数ごとのサンプル数の取得処理部 312、変数型決定部 313、およびプログラム生成部 314 を有する。ここで、プログラム生成時には、ホストマシン 3 は、ニューラルネットワーク記述情報と、統計情報の間引き情報と、プログラム生成パラメータを入力から取得する。入力例えば、プログラム起動に指定されるパラメータ、OS から引き渡されるパラメータ、ファイルシステムの所定のファイルに記述されたパラメータ等である。

【0125】

ニューラルネットワーク記述データは、層数、それぞれの層の入力値の数、それぞれの層の重みの数、それぞれの層の定数加算値の数等を含む。統計情報の間引き情報 N は一般には 1 つの整数であり、N 個に 1 個の割合で統計情報を取得する回路が実装されることを表わす。すなわち、統計情報の間引き情報 N は、N 回の演算に対して 1 回の割合で統計情報が取得されることを意味する。それぞれの変数の生成個数はニューラルネットワーク記述データから取得可能である。それぞれの変数の統計情報の取得回数 (サンプル数) は、生成個数 / 間引き情報 (ここで、/ は除算) で求めることができる。間引き情報 N は、演算処理部の一例である DL 実行プロセッサ 10A における演算回数に対する統計情報の取得割合ということができる。

【0126】

プログラム生成パラメータは、例えば、各変数のスケールファクタの初期値、あるいは変数のデータ型を決定するための変数ごとの取得回数 (サンプル数) の閾値等を含む。スケールファクタの初期値は、処理開始時の固定小数点数の小数点位置を指定する情報である。

【0127】

変数ごとの生成回数取得処理部 311 は、ニューラルネットワーク記述データを入力として、変数ごとに変数値の生成回数を取得し、変数ごとのサンプル数の取得処理部 312 に引き渡す。変数ごとに変数値の生成回数は、ニューラルネットワークの各層における演算回数ということもできる。変数ごとの生成回数取得処理部 311 は第 1 の取得部の一例

である。

【0128】

変数ごとのサンプル数の取得処理部312は、変数ごとの生成回数取得処理部311から引き渡された変数ごとの変数値の生成回数と、プログラムに入力される統計情報の間引き情報に基づいて、変数ごとの統計情報の取得回数（サンプル数）を取得し、変数型決定部313に引き渡す。変数ごとのサンプル数の取得処理部312は、第2の取得部の一例である。

【0129】

変数型決定部313は、ニューラルネットワーク記述データと、変数ごとのサンプル数の取得処理部312から引き渡された変数ごとの統計情報の取得回数（サンプル数）とに基づいて、変数の型情報を決定する。プログラム生成部314は、ニューラルネットワーク記述データと、プログラム生成パラメータと、変数型決定部313から引き渡された変数の型情報とに基づいて、テンプレートを修正することで深層学習等の処理を実行するプログラムを生成し、DL実行プロセッサ10Aに出力する。変数型決定部313とプログラム生成部314との組み合わせがプログラムを生成する生成部の一例である。

【0130】

図29にフルコネクト層のプログラムを生成するためのテンプレートを例示する。このテンプレート例の処理は、データ数D／16回のループと、C個のベクトルの積和演算を実行するループを含む。この処理では、データ数D／16回のループ中で、アキュムレートレジスタacc_cir_reg()の配列をクリアする処理と、C回のループ中で、入力値xをベクトルレジスタにロードする処理と、重みwをロードする処理と、入力値xと重みwの積和演算を実行する処理を含む。さらに、この処理は、D／16回のループ中で、バイアスbをロードする処理と、積和演算結果にバイアスbを加算する処理と、統計情報（非符号最上位ビット位置分布）を取得する処理と、積和演算結果をメモリにストアする処理を含む。

【0131】

このうち、例えば、forループのループ回数を指定する値D／16、C、およびi16_to_i16は、パラメータで設定可能な値である。したがって、プログラム生成部314は、これらのパラメータを変更することで、テンプレートを変更してプログラムを生成できる。なお、i16_to_i16の代り、i16_to_i8、i8_to_i16、i16_to_f32、f32_to_i16等のパラメータが使用可能である。i16_to_i16は、16ビット固定小数点数維持（型変更なし）、i16_to_i8は16ビット固定小数点数から8ビット固定小数点数への変更、i8_to_i16は8ビット固定小数点数から16ビット固定小数点数への変更を指定する。また、i16_to_f32は、16ビット固定小数点数から32ビット浮動小数点数への変更、f32_to_i16は32ビット固定小数点数から16ビット浮動小数点数への変更を指定する。図のプログラムで、8、16等の数値は、プロセッサのアーキテクチャによって決まるパラメータである。Cは、入力データ数、Dは出力データ数である。D／16は、16データ並列に出力されることを意味する。

【0132】

図30に、ニューラルネットワーク記述データの構成を例示する。ここで、Caffeと呼ばれるライブラリが使用する形式（prototext）での記述データの例を示す。この記述データは各層それぞれの内容をlayer {} 内に定義する。例えば、nameに層の名前が定義され、typeに層の種類が指定される。さらに、層の種類ごとの独自のデータが定義される。さらに、層数分だけlayer {} が設定される。

【0133】

図31に、prototextにおけるtopとbottomの定義を例示する。prototext独特の表現として、topとbottomがある。topとはある層が出力する先の層のことであり、bottomとはある層が入力を得る層のことである。例えば、処理がLayer1, Layer2, Layer3の順に実行される場合、Layer2のbottomがLayer1になり、Layer2のtopがLayer3になる。

【0134】

図 3 2 に、prototext によるニューラルネットワークの入力データの定義を例示する。prototext では、入力データも層の一つとして定義される。この時の type は "Data" となる。上述のように、データの出力先を top で指定する。図 3 2 の例は、conv1 と label という層に出力することを示す。conv1 は畳み込み層への出力を示し、label は、最終段階の正解判定への出力を示す。また、data_param にさらに詳細な情報が記述される。この例では、入力ファイルの名前 (source で指定)、バッチサイズ (batch_size で指定)、形式 (backend で形式「LMBD」) を格納する。

【0135】

なお、プログラム生成部 3 1 4 は、入力情報を格納したファイルを読むことにより、図 3 2 に例示した情報以外のさらに詳しい情報を得ることができる。たとえば、画像ファイルの大きさ、画像ファイルの入力チャンネル数 (カラーチャンネルの数) が取得される。これらの情報はサンプル取得数の計算に用いられる。

10

図 3 3 は、prototext によるニューラルネットワークの畳み込み層の定義を例示する。

【0136】

畳み込み層は、"Convolution" という文字列が type に指定される。また、入力先は、図 3 2 の入力データ (name が "input") となるので、"input" という文字列が top に指定される。convolution_param [] は畳み込み層のパラメータであり、num_output で出力チャンネル数を指定する。図 3 3 の場合は、64 チャンネルが指定されている。また、kernel_size で畳み込みのフィルタサイズを指定する。図 3 3 の場合は 3 x 3 フィルタが指定されている。

【0137】

さらに、前段の input 層から画像サイズ、入力チャンネル数、入力画像数 (バッチサイズ) を得ることにより、この層での変数値の生成回数を求めることができる。例えば、入力画像の大きさを 96 x 96 とする。また、バッチサイズは 32 とする。つまり 32 枚分の入力画像に対する画像の統計情報を収集する。この層ではフィルタサイズ 3 x 3 で 64 チャンネル出力する。簡単のため、畳み込みの際、ゼロパディングを用いると仮定すると入力画像の大きさと出力画像の大きさは等しくなる。したがって、畳み込み層の変数値の生成回数は次の通りである。

20

回数 = $96 \times 96 \times 64 \times 32 = 18874368$

【0138】

図 3 4 は、prototext によるニューラルネットワークのフルコネクト層の定義を例示する。フルコネクト層は "InnerProduct" という文字列が type に指定される。inner_product_param [] はフルコネクト層のパラメータであり、num_output で出力チャンネル数が指定される。図 3 4 の場合は 4096 が指定されている。フルコネクト層の場合、1 チャンネルにつき 1 データの生成であるため、4096 個のデータを生成する。バッチサイズが 32 であるとする 4096 x 32 = 131072 個のデータが生成される。

30

【0139】

<プログラム生成の処理フロー>

(プログラム生成の簡略フロー)

図 3 5 に、実施形態 2 におけるプログラム生成の簡略フローを例示する。図 3 5 は、実施形態 2 の処理を簡易に説明するために、詳細を省略して説明する処理フロー図である。この処理では、変数型決定部 3 1 3 が、変数の統計情報サンプル数が閾値より少ないか否かを判定する。変数の統計情報サンプル数が閾値に達している場合 (S 1 で NO)、変数型決定部 3 1 3 は、選択可能なデータ型のうち、ダイナミックレンジが狭い型を選択し、変数の型に設定する。そして、プログラム生成部 3 1 4 が決定された型にしたがってプログラムを生成する (S 2)。ここで、ダイナミックレンジは、変数値の取り得る範囲を示す特性情報ということができ、例えば、表現可能な値の最小値に対する最大値の比率によって定義できる。S 1 の判定が NO の場合は、統計情報のサンプル数が基準値より多いと判断した場合の一例である。また、S 2 の処理は、生成部が、ダイナミックレンジがより小さいデータ型を対象データに設定することの一例である。

40

【0140】

50

一方、変数の統計情報サンプル数が閾値より少ない場合（S 1でYES）、変数型決定部313は、選択可能なデータ型のうち、ダイナミックレンジが広い型を選択し、変数の型に設定する。そして、プログラム生成部314が決定された型にしたがってプログラムを生成する（S 3）。S 1の判定がYESの場合は、統計情報のサンプル数が基準値より少ないと判断した場合の一例である。また、S 3の処理は、生成部が、ダイナミックレンジがより大きいデータ型を対象データに設定することの一例である。

（閾値決定の処理）

【0141】

上述のように、実施形態2においては、変数型決定部313は、統計情報のサンプル数が閾値を超えるかどうかで変数の型を決定する。ここで、閾値の決定方法は以下の方法が考えられる。

【0142】

第1の方法は、DL実行プロセッサ10A用のプログラムを生成する生成プログラム内部のパラメータとして閾値を実装する方法である。すなわち、閾値自体を生成プログラム内部にハードコーディングする方法が例示できる。

第2の方法は、DL実行プロセッサ10A用のプログラムを生成する生成プログラムがファイル等から読み込む設定情報の1つに閾値を含める方法である。

【0143】

第3の方法として、ホストマシン3が試験的に学習を実行して閾値を決定してもよい。例えば、ホストマシン3は、32ビット浮動小数点表現と、閾値初期値を用いて型を決定した型の固定小数点表現との両方で、目的とする回数よりも少ない回数Nで試験的に学習を実行する。そして、固定小数点表現での学習誤差が32ビット浮動小数点表現を用いた場合の学習誤差に所定値 δe を加えた値よりも大きいならば、それぞれの型決定にかかわる閾値を、閾値の種類ごとに予め決められた差分ずつ増やす処理を実行する。この処理は、型決定の閾値を増加することで、よりダイナミックレンジが広い型が選ばれやすくする。ホストマシン3は、このような閾値の変更後、再度型を決定し、固定小数点の試験学習を再実行する。そして、固定小数点での学習誤差が32ビット浮動小数点表現を用いた場合の学習誤差に所定値 δe を加えた値以下となる条件が充足されたときに、ホストマシン3は、試験的な学習を終了し、本来の処理に切り替えて学習を続行することで閾値を正式に設定すればよい。

【0144】

図36に、閾値決定の処理を例示する。ここでは、仮決めされた閾値を基に、目的とする回数よりも少ない回数Nで試験的に学習を実行し、閾値を決定する処理を例示する。この処理では、ホストマシン3の変数型決定部313は、仮決めした閾値にしたがって型決定を行い、深層学習のプログラムを生成する（S10）。次に、ホストマシン3は、DL実行プロセッサ10Aで32ビット浮動小数点表現を用いて、N回事前学習を実行し、学習誤差を結果として受け取り、変数 $e f$ に格納する（S11）。変数 $e f$ の値は、浮動小数点数を用いて深層学習を実行したときの学習誤差の一例である。次に、ホストマシン3は、DL実行プロセッサ10Aを用いて固定小数点表現でN回事前学習をし、学習誤差を結果として受け取り、変数 $e i$ に格納する（S12）。変数 $e i$ の値は、固定小数点数を用いて深層学習を実行したときの学習誤差の一例である。S11、S12の処理は、所定の限定された回数の深層学習を実行することの一例である。

【0145】

次に、ホストマシン3は、変数 $e i$ の誤差が変数 $e f$ の誤差と事前に設定した基準値（ δe ）との和（判定値という）よりも小さいか否かを判定する（S13）。変数 $e i$ の誤差が上記判定値よりも小さい場合、ホストマシン3は、閾値を減少するための調整を実行する（S14）。一方、変数 $e i$ の誤差が上記判定値以上の場合、ホストマシン3は、閾値を増加するための調整を実行する（S15）。個々で、閾値を減少することは、統計情報のサンプル数が閾値以上の値になりやすくなる。その結果、変数型決定部313がダイナミックレンジの狭い変数を選びやすくなる結果となる。一方、閾値を増加することは、

10

20

30

40

50

統計情報のサンプル数が閾値より小さい値になりやすくなる。その結果、変数型決定部 313 がダイナミックレンジの広い変数を選びやすくなる結果となる。S13 から S15 の処理は、浮動小数点数を用いて深層学習を実行したときの学習誤差と、固定小数点数を用いて深層学習を実行したときの学習誤差との差に基づいて基準値を決定することの一例である。

【0146】

なお、図36は、S13の判定で、変数 e_i の誤差が単一の判定値より小さいか否かで閾値を変更するが、閾値を増加する場合と、閾値を減少する場合とで、異なる判定値を用いてもよい。例えば、変数 e_f の誤差と事前に設定した基準値 (δe) との和を第1の判定値し、変数 e_f の誤差と事前に設定した基準値 (δe) の $1/2$ 倍の値との和を第2の判定値としてもよい。そして、変数 e_i の誤差が第1の判定値より小さい場合には、閾値を維持し、変数 e_i の誤差が第2の判定値より小さい場合には、ホストマシン3は、閾値を減少するための調整を実行すればよい。また、ホストマシン3は、変数 e_i の誤差が第2の判定値と第1の判定値の間に収まるまで図36の処理を繰り返し実行してもよい。図36の処理は、基準値を決定する決定部の処理の一例といえることができる。ホストマシン3は、決定部の一例として図36の処理を実行すると言える。

【0147】

図37は、閾値決定の処理の第1の変形を例示する。この処理では、仮決めした閾値 T と浮動小数点表現で学習した時の学習誤差 e_f が入力として与えられる。また、この処理では、仮決めした閾値 T は、想定より大きめの値として設定される。この場合に、当初は統計データのサンプル数が少ないと判定され、変数のダイナミックレンジが広く設定され、エラーが小さくなると想定される。ホストマシン3はエラーがある程度大きくなるまで閾値を減少させる。

【0148】

ホストマシン3の変数型決定部313は、閾値 T から一定値 (δt) を減少する (S21)。次に、ホストマシン3の変数型決定部313は、閾値にしたがって型決定を行い、深層学習のプログラムを生成する (S22)。次に、ホストマシン3は、DL実行プロセッサ10Aを用いて固定小数点表現でN回事前学習を実行し、学習誤差を結果として受け取り、変数 e_i に格納する (S23)。S22の処理は、所定の限定された回数の深層学習を実行することの一例である。

【0149】

次に、ホストマシン3は、変数 e_i の誤差が変数 e_f の誤差と事前に設定した基準値 (δe) との和 (判定値) よりも大きいかなんかを判定する (S24)。変数 e_i の誤差が上記判定値以下の場合、ホストマシン3は、処理をS21に戻し、閾値を減少するための調整を実行する。一方、ホストマシン3は、変数 e_i の誤差が上記判定値より大きい場合、閾値 T を調整後の閾値として決定する (S25)。S25の処理は、浮動小数点数を用いて深層学習を実行したときの学習誤差と、固定小数点数を用いて深層学習を実行したときの学習誤差との差に基づいて基準値を決定することの一例である。

【0150】

この処理では、処理の初期段階では、仮決めした閾値 T は、想定より大きめの値として設定されるので、統計情報のサンプル数が閾値以下となりやすく、ダイナミックレンジの広い型の変数が選択される。その結果、S24の判定はNOになりやすく、処理が進むと、S24の判定はYESになる。図37の処理は、基準値を決定する決定部の処理の一例といえることができる。ホストマシン3は、決定部の一例として図37の処理を実行すると言える。

【0151】

図38は、閾値決定の処理の第2の変形を例示する。この処理では、仮決めした閾値 T と浮動小数点表現で学習した時の学習誤差 e_f が入力として与えられる。また、この処理では、仮決めした閾値 T は、想定より小さめの値として設定される。ホストマシン3の変数型決定部313は、閾値 T に一定値 (δt) を加算する (S31)。次に、ホストマシン

3の変数型決定部313は、閾値にしたがって型決定を行い、深層学習のプログラムを生成する(S32)。次に、ホストマシン3は、DL実行プロセッサ10Aを用いて固定小数点表現でN回事前学習を実行し、学習誤差を結果として受け取り、変数 e_i に格納する(S33)。S33の処理は、所定の限定された回数の深層学習を実行することの一例である。

【0152】

次に、ホストマシン3は、変数 e_i の誤差が変数 e_f の誤差と事前に設定した基準値(δe)との和(判定値)よりも小さいか否かを判定する(S34)。変数 e_i の誤差が上記判定値以上の場合、ホストマシン3は、処理をS21に戻し、閾値を増加するための調整を実行する。一方、ホストマシン3は、変数 e_i の誤差が上記判定値より小さい場合、閾値Tを調整後の閾値として決定する(S35)。S35の処理は、浮動小数点数を用いて深層学習を実行したときの学習誤差と、固定小数点数を用いて深層学習を実行したときの学習誤差との差に基づいて基準値を決定することの一例である。

【0153】

この処理では、処理の初期段階では、仮決めした閾値Tは、想定より小さめの値として設定されるので、統計情報のサンプル数が閾値より大きくなりやすく、ダイナミックレンジの狭い型の変数が選択される。その結果、S34の判定はNOになりやすく、処理が進むと、S34の判定はYESになる。図38の処理は、基準値を決定する決定部の処理の一例といえることができる。ホストマシン3は、決定部の一例として図38の処理を実行するといえる。

【0154】

図39にプログラム生成処理を例示する。処理では、ホストマシン3は、生成対象のプログラムのニューラルネットワーク記述データを取得する(S41)。次に、ホストマシン3は、入力から統計情報の間引き数を取得する(S42)。次に、ホストマシン3は、それぞれの変数について生成回数と間引き数から、統計情報のサンプル数を取得する(S43)。そして、ホストマシン3の変数型決定部313は、それぞれの変数の統計情報サンプル数に基づいて型を決定する(S44)。次に、ホストマシン3のプログラム生成部314は、決定された変数の型に基づいてプログラムを生成する(S45)。

【0155】

図40は、変数型決定部313の処理(図39のS44)の詳細を例示する。この処理では、変数型決定部313は、変数の統計情報サンプル数が第1の基準値(sample1)以下であるか否かを判定する(S441)。変数の統計情報サンプル数が第1の基準値(sample1)以下である場合、変数型決定部313は、変数の型を32ビット浮動小数点数にする(S442)。また、変数の統計情報サンプル数が第1の基準値(sample1)より大きい場合、変数型決定部313は、変数の統計情報サンプル数が第2の基準値(sample2 > sample1)以下であるか否かを判定する(S443)。変数の統計情報サンプル数が第2の基準値(sample2)以下である場合、変数型決定部313は、変数の型を16ビット固定小数点数にする(S444)。また、変数の統計情報サンプル数が第2の基準値(sample2)より大きい場合、変数型決定部313は、変数の型を8ビット固定小数点数にする(S445)。S442の処理は、サンプル数が基準値より少ないと判断した場合には、浮動小数点数のデータ型を対象データに設定することの一例といえる。S444、S445の処理は、サンプル数が基準値より多いと判断した場合には、固定小数点数のデータ型を対象データに設定することの一例といえる。

【0156】

図41は、プログラム生成部314の処理を例示する。この処理では、プログラム生成部314は、複数階層のニューラルネットワーク中の各層の依存関係を整理する。すなわち、プログラム生成部314は、ニューラルネットワークの各層を順伝播の順に並びかえ、Layer[0], Layer[1], ..., Layer[L-1]として管理する(S451)。次に、プログラム生成部314は、Layer[0], Layer[1], ..., Layer[L-1]それぞれについて、順伝播と逆伝播のプログラムを生成する(S452)。プログラム生成部314は、図29に例示した

テンプレート中の入力数、変数の型等を変更することでプログラムを生成する。さらに、プログラム生成部 314 は、Layer[0], Layer[1], ..., Layer[L-1] の順伝播と逆伝播を呼び出すコードを生成する (S453)。以上の処理によって、深層学習を実行するプログラムが生成される。

<生成されたプログラムの実行>

【0157】

ホストマシン 3 は、生成したプログラムを DL 実行プロセッサ 10A に転送し、プログラムの実行を起動する。DL 実行プロセッサ 10A は、実施形態 1 と同様の手順で、深層学習等の処理を実行し、処理結果をホストマシン 3 に返す。

【0158】

図 42 は、DL 実行プロセッサ 10A で実行される処理のうちの型に合わせた型変換パラメータの設定処理を例示する図である。型に合わせた型変換パラメータの設定処理は、DL 実行プロセッサ 10A において演算が実行されるときに、演算対象のデータの型を合わせる処理である。型に合わせた型変換パラメータの設定処理では、DL 実行プロセッサ 10A は、基本的にはビット幅の大きい変数に合わせて型変換を実行する。また、DL 実行プロセッサ 10A は、浮動小数点数と固定小数点数の演算では、浮動小数点数に合わせて演算を実行するように型変換を実行する。ここで、演算の種類は、図 27 の DL 実行プロセッサ 10A に関して、(スカラ演算の種類) と (ベクトル演算 (SIMD 演算)) で述べたものである。

【0159】

図 42 では、これらのうち、16 ビット固定小数点数と 8 ビット固定小数点数の積和演算の一部である乗算を変数 (ベクトル) X と W に対して行う場合の型変換の処理が例示される。また、この処理では、DL 実行プロセッサ 10A で演算可能な固定小数点数は 8 ビットか 16 ビットであると想定する。この処理では、DL 実行プロセッサ 10A は、変数 X、W の型が両方とも 8 ビット固定小数点数であるか否かを判定する (S101)。変数 X、W の型が両方とも 8 ビット固定小数点数である場合、DL 実行プロセッサ 10A は、変数 X、W のレジスタ格納型を 8 ビット固定小数点数にする (S102)。一方、変数 X、W の型が両方とも 8 ビット固定小数点数でない場合、プロセッサ 10A は、変数 X、W のレジスタ格納型を 16 ビット固定小数点数にする (S103)。そして、DL 実行プロセッサ 10A は、型変換後の変数に対して演算を実行する。

【0160】

図 43 は、シフト幅の設定を例示する図である。いま、演算が例えば、ベクトル X とベクトル W の内積と、バイアス b の加算として、 $X * W + b$ であり、演算結果がアキュムレートレジスタ acc に積算されるとする。また、ベクトル X およびベクトル W の各要素のスケールファクタ、すなわち、小数点位置のシフト量がそれぞれ S_x 、 S_w であると仮定する。すると、アキュムレートレジスタ Acc のスケールファクタは、ベクトル X およびベクトル W の各要素の乗算の結果であるため、 $S_x + S_w$ となっている。したがって、バイアス b のスケールファクタを S_b とすると、DL 実行プロセッサ 10A は、 $\Delta Wb = S_x + S_w - S_b$ だけ左シフトさせる必要がある。

【0161】

また、図 44 は、非符号となる最上位ビットの位置の分布からオーバーフロー率を設定し、最上位ビット位置 η (eta) を決定する処理を例示する図である。図 44 は、実施形態 1 同様の構成によって得られるプログラムのループ中での非符号である最上位ビットの位置分布のうち、オーバーフロー率が r_{max} となる点の桁位置を η とする。

【0162】

図 45 は、演算 $Z = X * W + b$ による演算結果 Z の次のミニバッチで使用するスケールファクタの決定方法を例示する図である。DL 実行プロセッサ 10A は、ビット 0 から η (図 44 で決定された桁位置) までを次のミニバッチにおける有効ビット幅とみなす。演算 $Z = X * W + b$ による演算結果 Z をレジスタに格納する際の右シフト量 ΔS_z は、 $\Delta S_z = \eta + 2 - \beta$ で求められる。ここで、 β は、格納先のレジスタのビット幅であり、例え

10

20

30

40

50

ば、16ビット、8ビット等である。したがって、次のミニバッチで使用する演算結果格納先の変数Zのスケールファクタは、 $S_z = S_x + S_w - \Delta S_z$ で求めることができる。DL実行プロセッサ10Aは、このように決定されるスケールファクタを用いて、各固定小数点数の少数位置を設定し、次のミニバッチの各演算を実行すればよい。

【0163】

また、例えば、深層学習での各層の入力Xは、前の層（順方向伝播の下位層）の出力Zと同じであるので、DL実行プロセッサ10Aは、前の層の出力Zのスケールファクタをそのまま使用すればよい。また、DL実行プロセッサ10Aは、重み係数Wとバイアスbのスケールファクタを逆方向伝播時に更新すればよい。

<実施形態2の効果>

【0164】

図46は、インターネット上の画像データベースImageNetの8階層ネットワークVGG8による認識結果を例示する。この例では、統計情報は1/16に間引かれている。図のように、大半の変数を8ビット固定小数点数表現で処理し、一部の変数（例えば、1つのミニバッチでのサンプル数62以下、あるいはサンプル数32以下の変数）を16ビット固定小数点数のように、ビット幅を拡張して処理すると、認識精度が向上する。一部の変数を浮動小数点数表現で処理する場合も同様である。したがって、実施形態2の手順および情報処理方法によれば、ホストマシン3は、DL実行プロセッサ10Aのように回路規模を抑制した固定小数点数を用いて少ないビット幅でニューラルネットワーク等の学習処理を含む演算を行い、認識精度を向上できる。

【0165】

また、ホストマシン5は、実際の学習回数よりも少ない学習回数N回の学習の実行結果により、閾値Tを決定し、当該閾値Tを基に統計情報のサンプル数が十分か否かを判定し、変数の型を選択する。したがって、ホストマシン5は、学習結果を反映して、統計情報のサンプル数が十分か否かの閾値を決定できる。

【0166】

また、ホストマシン3は、DL実行プロセッサ10Aが実行した学習のエラーに基づいて閾値を決定するので、実際の学習に適した閾値で学習に使用する変数の型を決定できる。また、ホストマシン3は、型を決定する際、通常の学習より少ない回数NでDL実行プロセッサ10Aに学習させるので、短時間低負荷で閾値を決定できる。

【0167】

また、ホストマシン5は、浮動小数点数を用いて学習したときの学習誤差（ef）と、固定小数点数を用いて学習したときの学習誤差（ei）を比較する。そして、学習誤差（ei）が学習誤差（ef）に十分近づくまで処理を繰り返し、閾値を決定するので、固定小数点数を用いる場合でも、固定小数点数での学習誤差（ei）が動小数点数での学習誤差（ef）に近づくように閾値T設定できる。

【0168】

また、実施形態2のホストマシン3は、ニューラルネットワーク記述データを取得し、統計情報のサンプル数を算出するので、精度良くサンプル数を取得し、変数の型を決定できる。また、実施形態2のホストマシン3は、変数の型を決定後、テンプレートを用いて変数の型を変更し、プログラムを生成するので、簡易にプログラムを生成できる。

<変形例>

【0169】

上記実施形態2では、ホストマシン3とDL実行ハードウェア1Aを含む情報システムにおいて、ホストマシンが3学習プログラムを生成した。しかし、ホストマシン3とDL実行ハードウェア1Aとが一体のコンピュータであってもよい。すなわち、実施形態1で述べたプロセッサ10あるいは図27のDL実行プロセッサ10Aが学習プログラムを生成するとともに、生成した学習プログラムを実行するようにしてもよい。図27のDL実行プロセッサ10Aが学習プログラムを生成するとともに、生成した学習プログラムを実行する場合には、演算処理部の一例であるDL実行プロセッサ10Aが同一のプロセッサ

10

20

30

40

50

でプログラムを生成する生成部としての処理を実行するといえる。なお、図 23 で、ホストマシン 3 と DL 実行ハードウェア 1A とは単一の筐体に収納されていてもよいし、異なる筐体に収納されていてもよい。

《コンピュータが読み取り可能な記録媒体》

【0170】

コンピュータその他の機械、装置（以下、コンピュータ等）に上記いずれかの機能を実現させるプログラムをコンピュータ等が読み取り可能な記録媒体に記録することができる。そして、コンピュータ等に、この記録媒体のプログラムを読み込ませて実行させることにより、その機能を提供させることができる。

【0171】

ここで、コンピュータ等が読み取り可能な記録媒体とは、データやプログラム等の情報を電氣的、磁氣的、光学的、機械的、または化学的作用によって蓄積し、コンピュータ等から読み取ることができる記録媒体をいう。このような記録媒体のうちコンピュータ等から取り外し可能なものとしては、例えばフレキシブルディスク、光磁気ディスク、CD-ROM、CD-R/W、DVD、ブルーレイディスク、DAT、8mmテープ、フラッシュメモリなどのメモリカード等がある。また、コンピュータ等に固定された記録媒体としてハードディスク、ROM（リードオンリーメモリ）等がある。さらに、SSD（Solid State Drive）は、コンピュータ等から取り外し可能な記録媒体としても、コンピュータ等に固定された記録媒体としても利用可能である。

【符号の説明】

【0172】

- 1 情報処理装置
- 1A DL実行ハードウェア
- 3 ホストマシン
- 10 プロセッサ
- 10A DL実行プロセッサ
- 11 制御ユニット
- 12 レジスタファイル
- 13、13A ベクトルユニット
- 13B ベクトル演算用の整数演算器
- 13C ベクトル演算用の浮動小数点数演算器
- 14 スカラユニット
- 21 命令用メモリ
- 22 データ用メモリ
- 101 セレクタ
- 102 統計情報取得部
- 103 データ変換部
- 104 統計情報集約部
- 105 統計情報格納部
- 111 プログラムカウンタ
- 112 デコーダ
- 131 ベクトル演算用の演算器
- 132 積和演算用のアキュムレータ
- 141 スカラ用の演算器

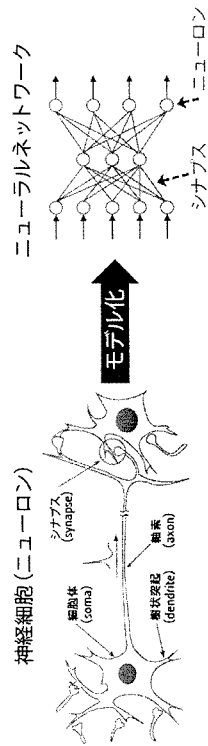
10

20

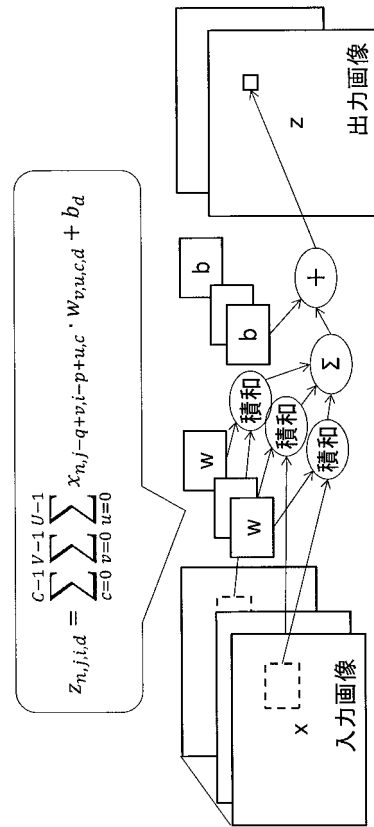
30

40

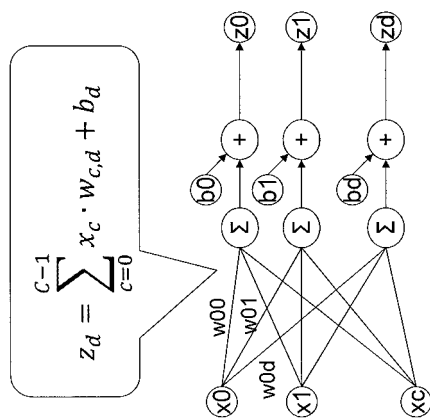
【図 1】



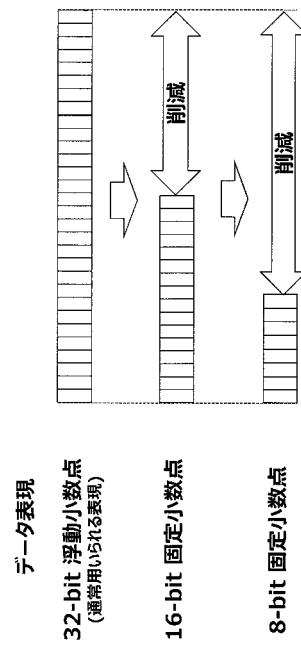
【図 2】



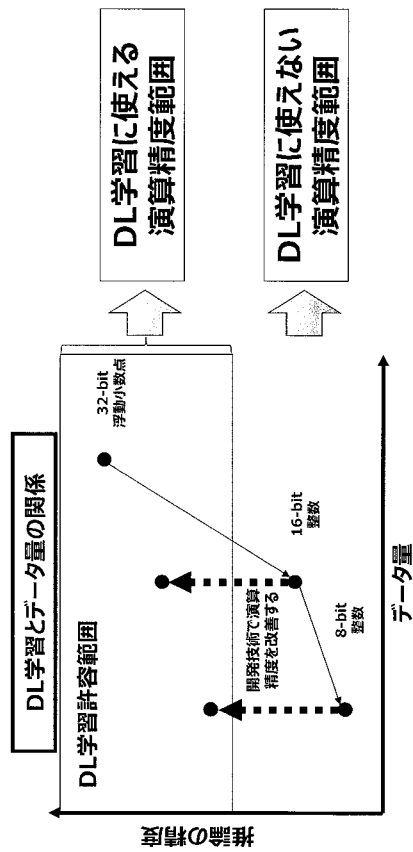
【図 3】



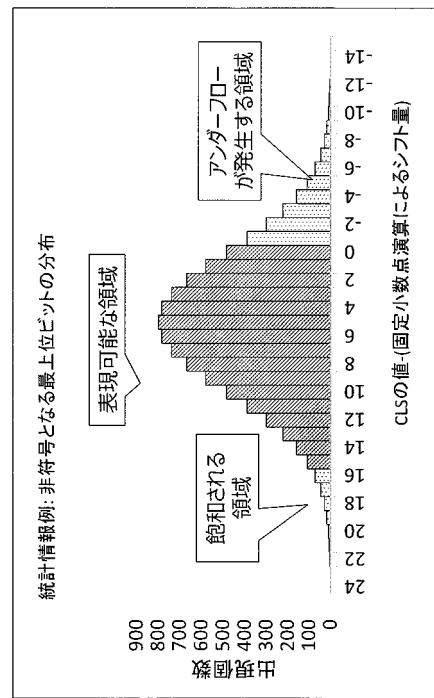
【図 4】



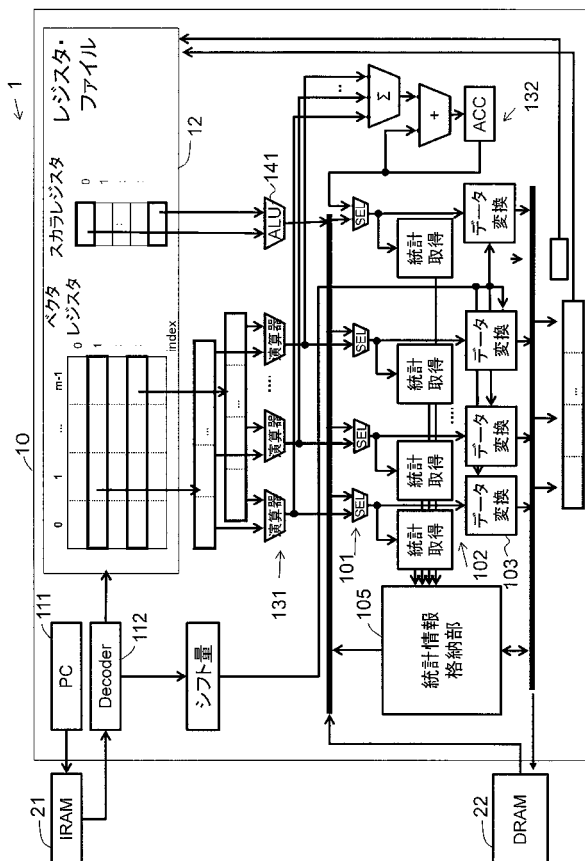
【 図 5 】



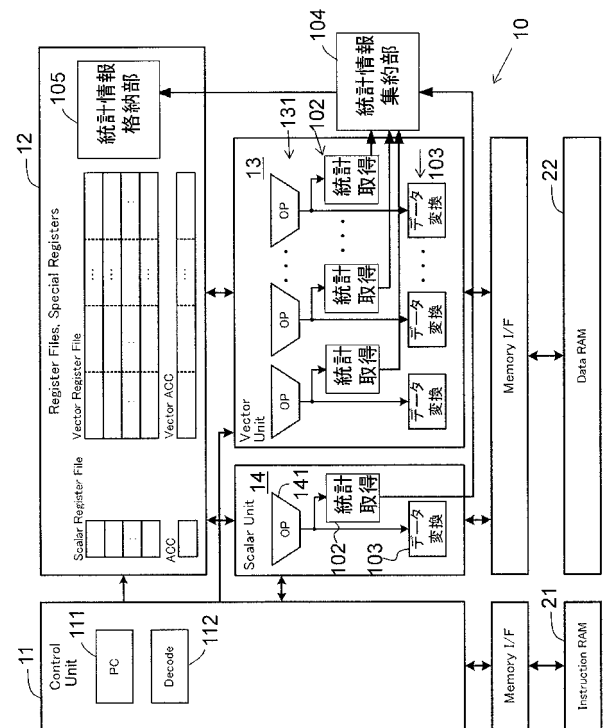
【図 6】



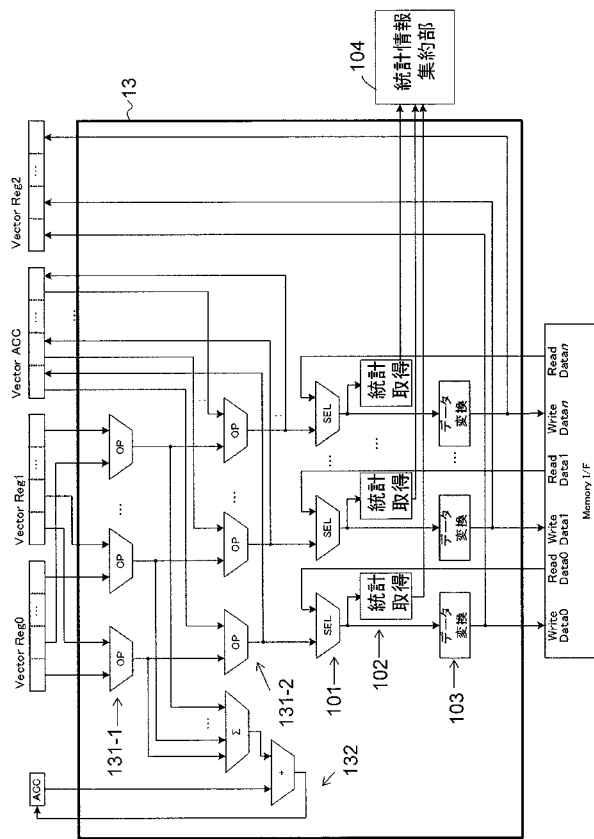
【图 7】



【图 8】



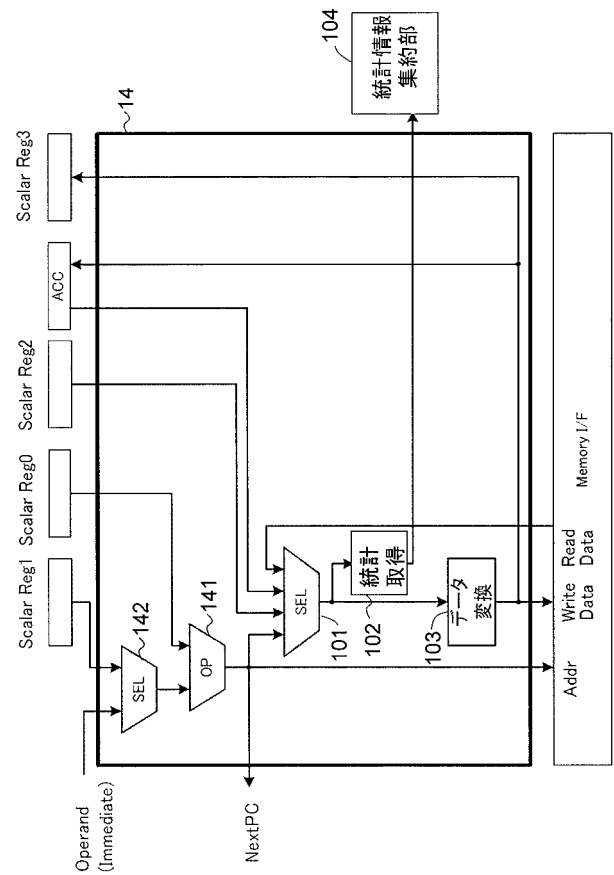
【図 9】



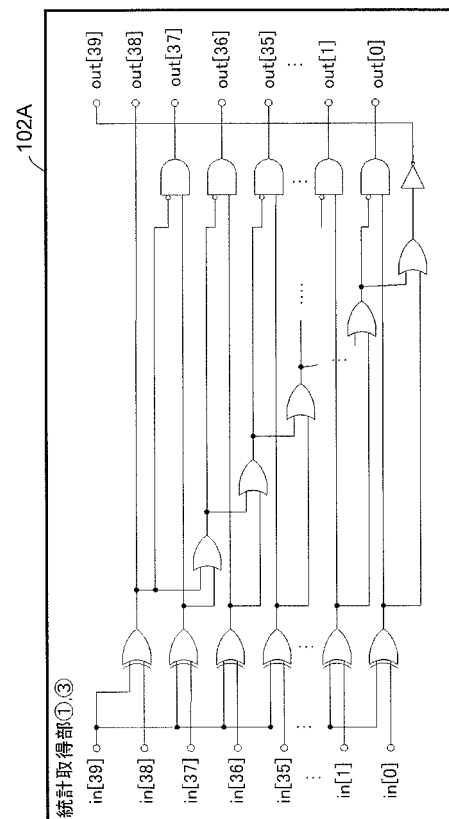
【図 11】

input				output	
in[39]	in[38]	in[37]	in[36]	in[1]	out
0	0	0	0	0	0x8000000000
1	1	1	1	1	0x8000000000
0	1	b	b	b	0x4000000000
1	0	b	b	b	0x4000000000
0	0	1	b	b	0x2000000000
1	1	0	b	b	0x2000000000
0	0	0	1	b	0x1000000000
1	1	1	0	b	0x1000000000
:	:	:	:	:	:
0	0	0	0	1	0x0000000002
1	1	1	1	0	0x0000000002
0	0	0	0	0	0x0000000001
1	1	1	1	1	0x0000000001

【図 10】



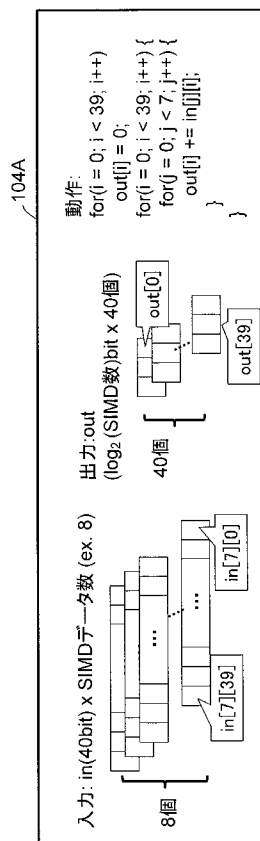
【図 12】



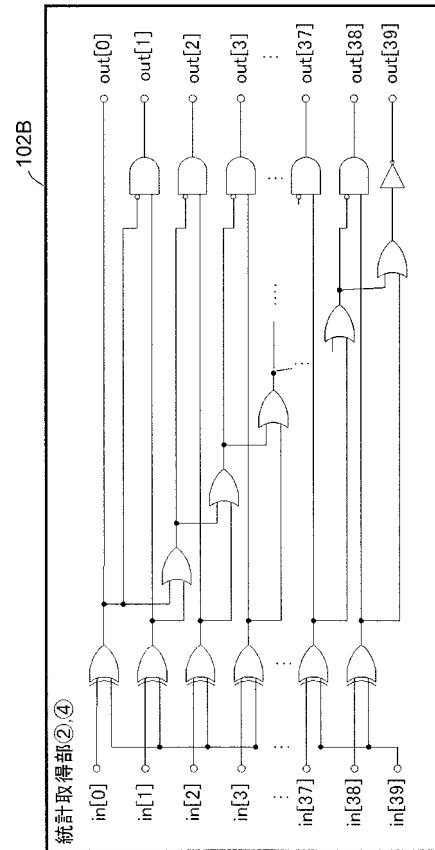
【図 1 3】

input				output			
in[39]	in[38]	in[37]	in[36]	in[1]	in[0]	out	out
0	b	b	b	b	1	0x000000001	0x000000001
1	b	b	b	b	0	0x000000001	0x000000001
0	b	b	b	1	0	0x000000002	0x000000002
1	b	b	b	0	1	0x000000002	0x000000002
:	:	:	:	:	:	:	:
0	b	b	1	0	0	0x100000000	0x100000000
1	b	b	0	1	1	0x100000000	0x100000000
0	b	1	0	0	0	0x200000000	0x200000000
1	b	0	1	1	1	0x200000000	0x200000000
0	1	0	0	0	0	0x400000000	0x400000000
1	0	1	1	1	1	0x400000000	0x400000000
0	0	0	0	0	0	0x800000000	0x800000000
1	1	1	1	1	1	0x800000000	0x800000000

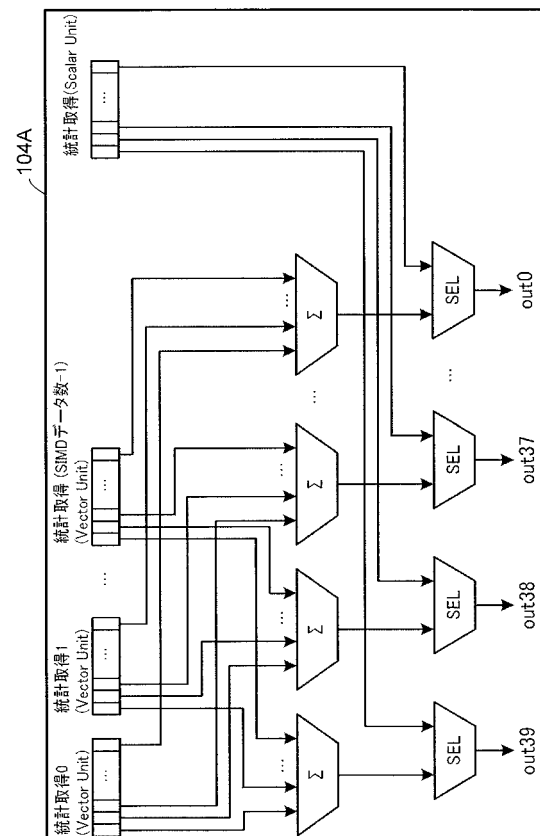
【図 1 5】



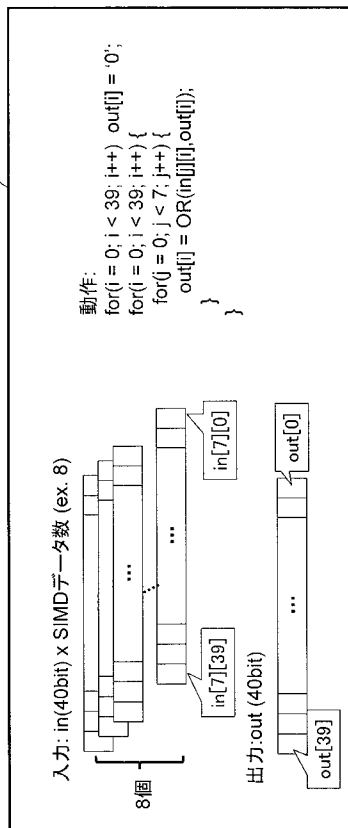
【図 1 4】



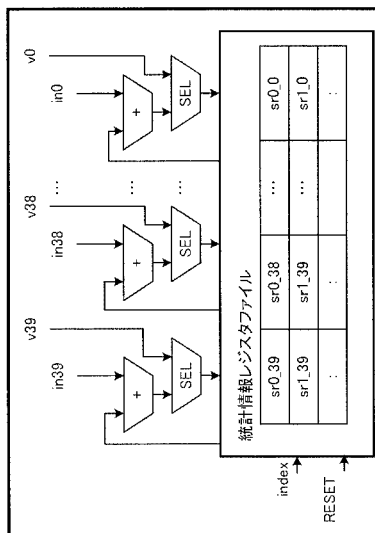
【図 1 6】



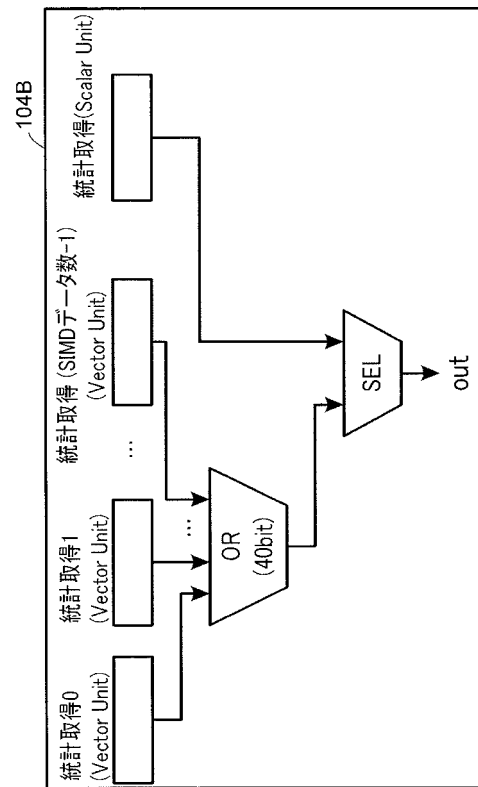
【図 17】



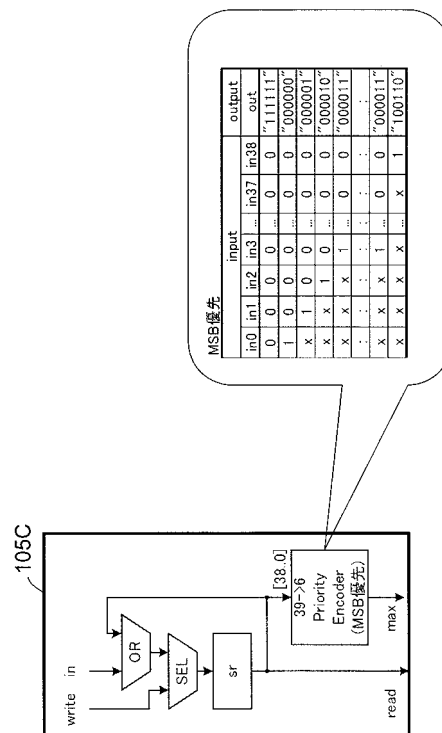
【図 19】



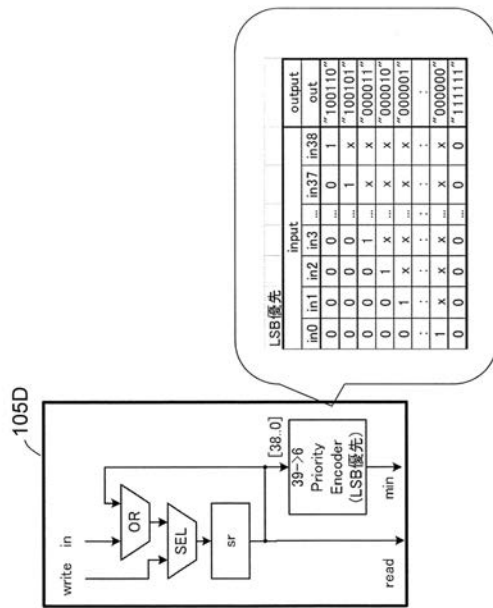
【図 18】



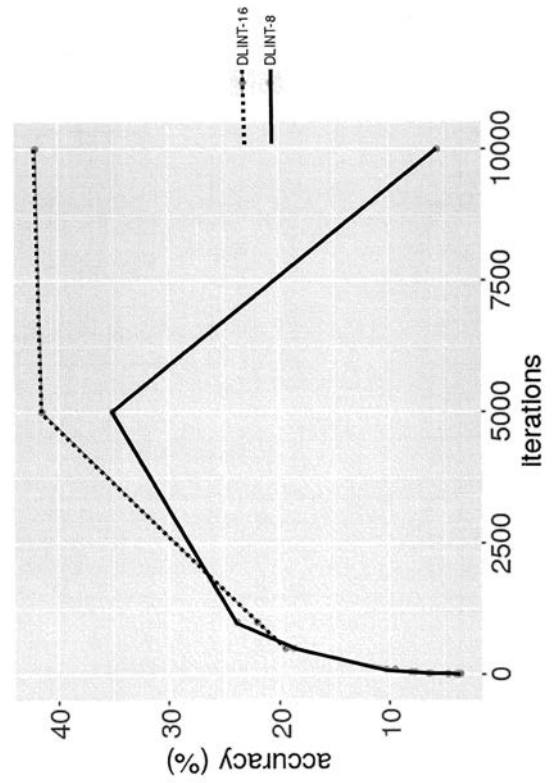
【図 20】



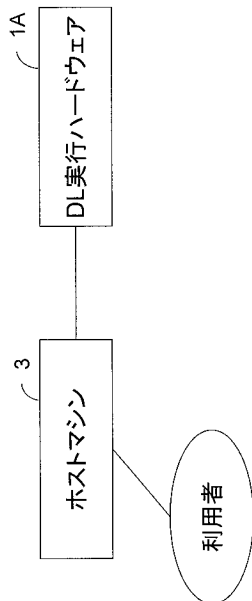
【図 2 1】



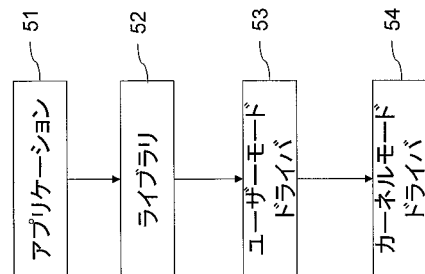
【図 2 2】



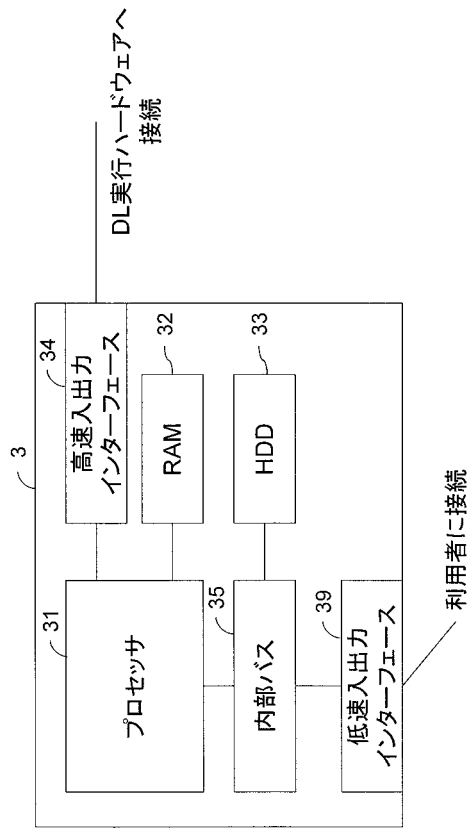
【図 2 3】



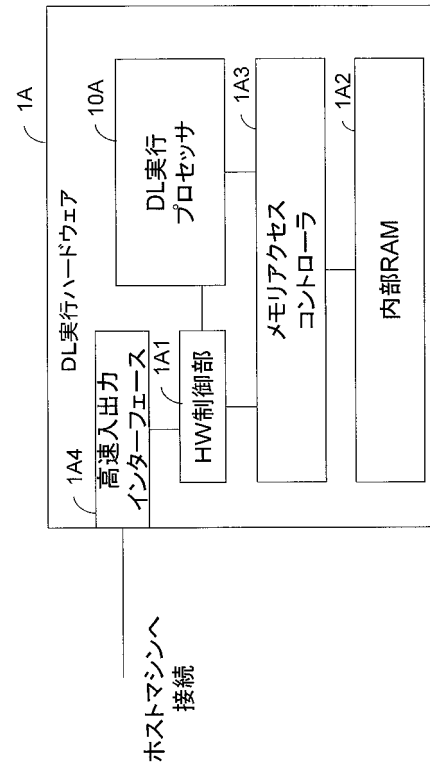
【図 2 4】



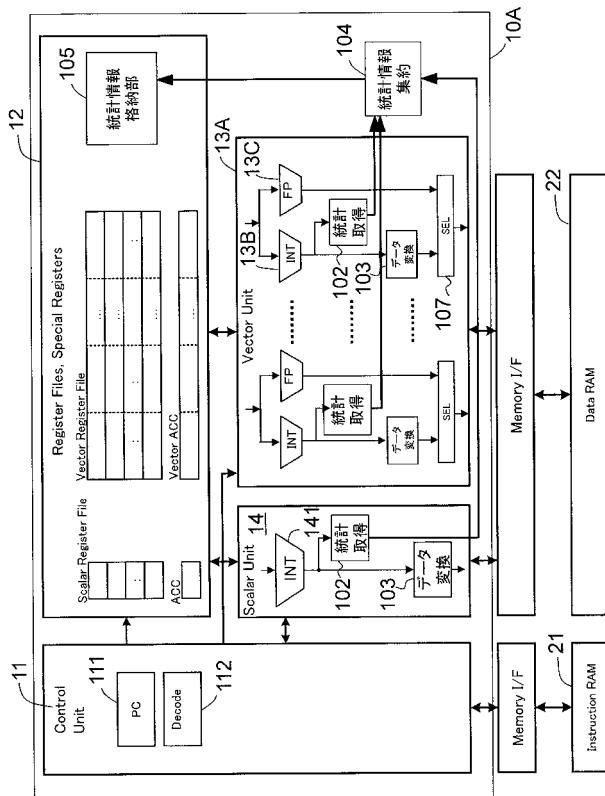
【図 25】



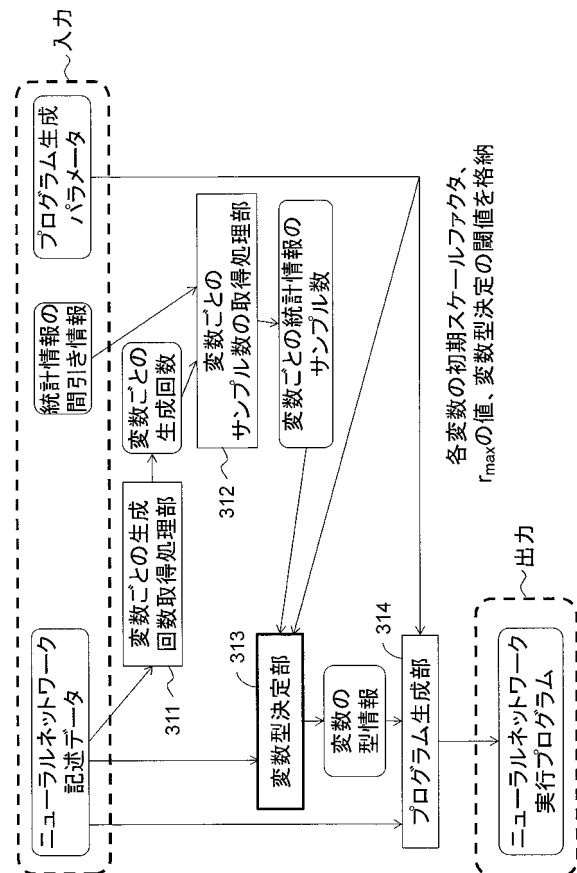
【図 26】



【図 27】



【図 28】



【図 29】

フルコネクト層のテンプレート例

$$Z_d = \sum_{c=0}^{C-1} x_c \cdot w_{c,d} + b_d$$

D/16、C、i16_to_i16、および16は、パラメータ設定可能な値。

```

for (d = 0; d < D / 16; d++) {
  acc_clr_reg(); // アキュムレートのレジスタのクリア
  for (c = 0; c < C; c++) {
    vector_load(simd1, i16_to_i16, addr(x(c)), stride=0); // 入力値 x のロード
    vector_load(simd2, i16_to_i16, addr(w(c, d*16)), stride=1); // 重み w のロード
    acc_mac(simd, simd2); // x と w の積和演算
  }
  vector_load(simd1, i16_to_i16, addr(b(d*16)), stride=1); // バイアス b のロード
  acc_add(simd1, Δω_b); // 積和演算結果にバイアス b の加算
  acc_store_mbs_pos(0); // 統計情報 (非符号最上位ビット位置分布) の取得
  acc_move(simd1, Δω_z);
  vector_store(simd1, i16_to_i16, addr(z(d*16)), stride=1); // 積和演算結果のストア
}
set_next_wb_wz(); // 次の演算に使用する Δω_b と Δω_z を算出する

```

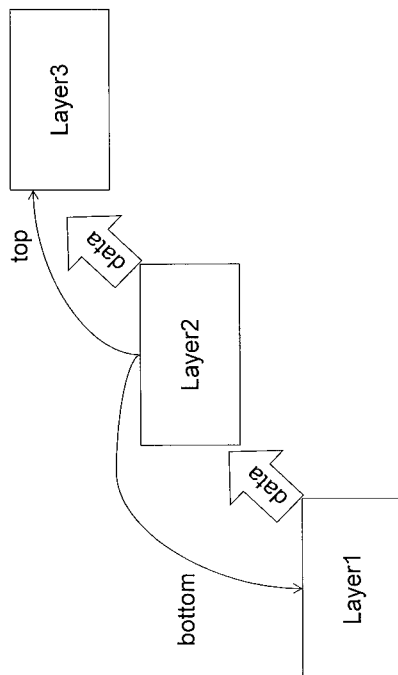
【図 30】

```

# # 行末まではコメント
name: "VG68" # ネットワーク全体の名前をここに記述する
layer {
  name: "layerName1"
  type: XXXX
  # typeの内容により記述内容が変化する
}
layer {
  name: "layerName2"
  type: YYYY
  # typeの内容により記述内容が変化する
}

```

【図 31】



【図 32】

```

layer {
  name: "input"
  type: "Data"
  top: "conv1"
  top: "label"
  data_param {
    source: "vgg16_train_lmdb"
    batch_size: 32
    backend: LMDB
  }
}

```

【図 3 3】

```

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "input"
  top: "relu1"
  convolution_param {
    num_output: 64
    kernel_size: 3
  }
}

```

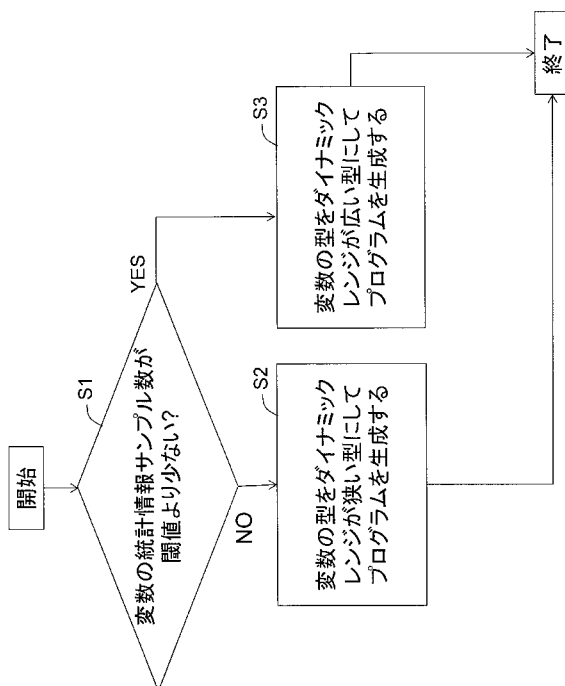
【図 3 4】

```

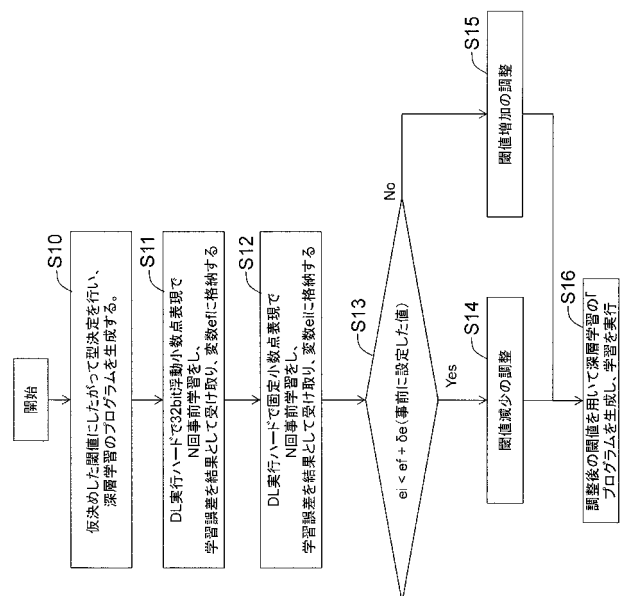
layer {
  name: "fc6"
  type: "InnerProduct"
  bottom: "pool5"
  top: "relu6"
  inner_product_param {
    num_output: 4096
  }
}

```

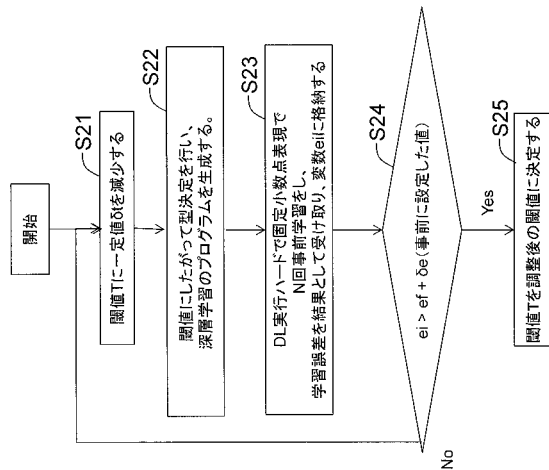
【図 3 5】



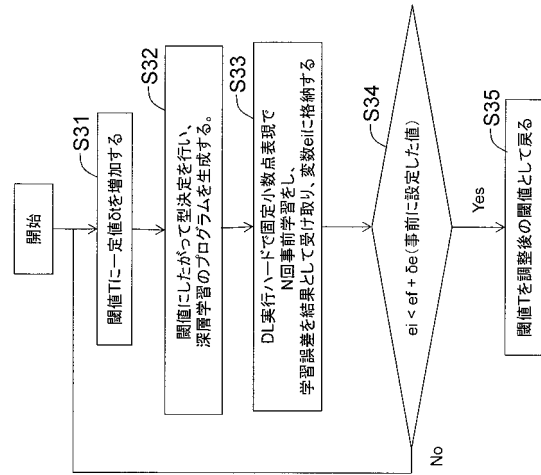
【図 3 6】



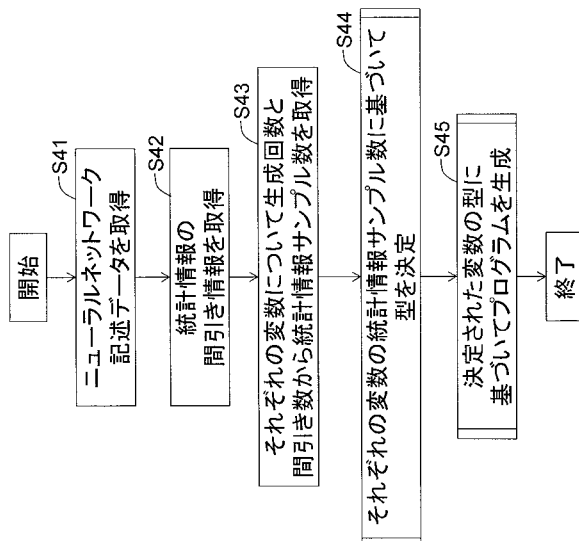
【図 37】



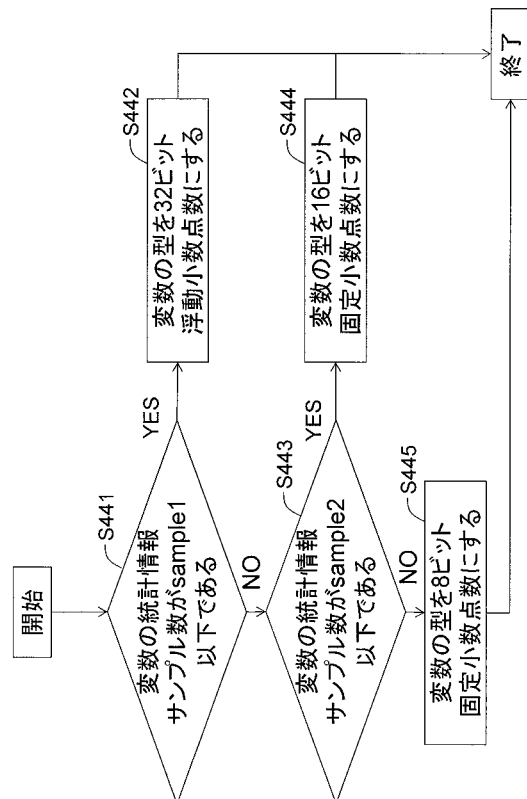
【図 38】



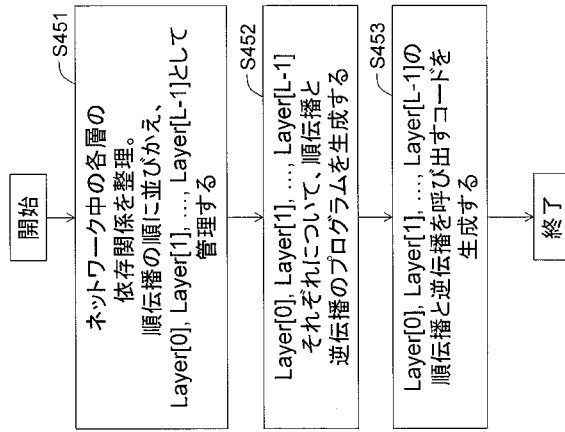
【図 39】



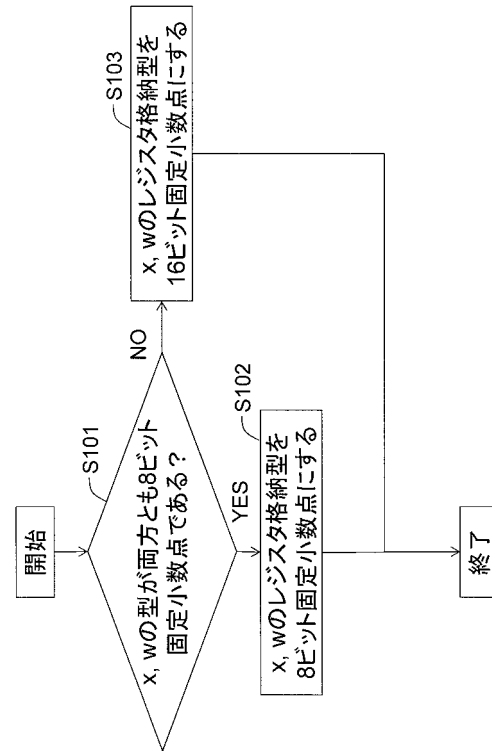
【図 40】



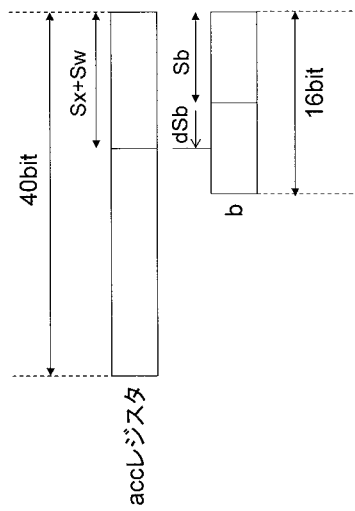
【図 4 1】



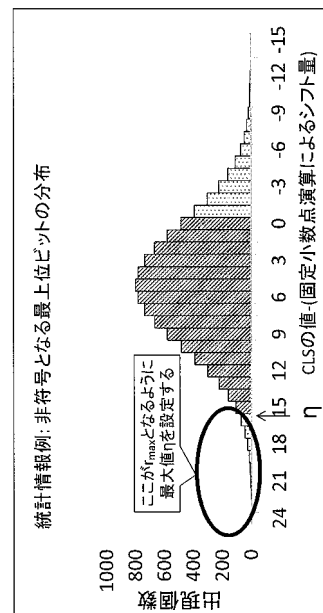
【図 4 2】



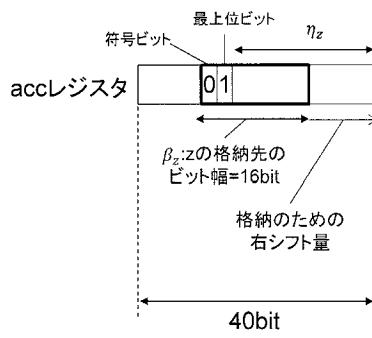
【図 4 3】



【図 4 4】



【図 4 5】



【図 4 6】

