(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6079433号
(P6079433)
(45)発行日　平成29年2月15日(2017.2.15)　(24)登録日　平成29年1月27日(2017.1.27)

(51)Int.Cl.　　F I
G06F 17/10 (2006.01)　　G06F 17/10　D
G06F 15/80 (2006.01)　　G06F 15/80

請求項の数 7　(全 17 頁)

(21)出願番号　特願2013-109210 (P2013-109210)
(22)出願日　平成25年5月23日 (2013.5.23)
(65)公開番号　特開2014-229133 (P2014-229133A)
(43)公開日　平成26年12月8日 (2014.12.8)
審査請求日　平成28年2月26日 (2016.2.26)

(73)特許権者　000005223
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　100090273
弁理士　國分　孝悦
(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
(72)発明者　久保田　学
福岡県福岡市早良区百道浜２丁目２番１号　富士通九州ネットワークテクノロジーズ株式会社内

最終頁に続く

(54)【発明の名称】移動平均処理プログラム、及びプロセッサ

(57)【特許請求の範囲】
【請求項１】
ｍ個（ｍは２以上の整数）の演算処理を並列に実行し、かつ０番目から（ｍ－１）番目の入力要素を基に、０番目からｐ番目（ｐは０～ｍ－１の整数）の前記入力要素の和を算出してｐ番目の結果要素としてそれぞれ返す部分総和命令を実行するプロセッサに、
入力データ系列のｉ番目から（ｉ＋ｍ－１）番目（ｉは０及び自然数のうちの任意の数）の要素を０番目から（ｍ－１）番目の前記入力要素とする前記部分総和命令を実行して第１のベクトルデータを取得する第１の演算処理と、
前記入力データ系列の（ｉ＋ｘ）番目から（ｉ＋ｘ＋ｍ－１）番目（ｘは自然数）の要素を０番目から（ｍ－１）番目の前記入力要素とする前記部分総和命令を実行して第２のベクトルデータを取得する第２の演算処理と、
前記入力データ系列のｉ番目から（ｉ＋ｘ－１）番目の要素の和に、前記第１のベクトルデータのｐ番目の要素を減算し、前記第２のベクトルデータのｐ番目の要素を加算する処理を、０番目から（ｍ－１）番目の各要素について並列に行うことにより、互いに異なるｍ個の区間について要素の和を並列に算出する第３の演算処理と、
算出した各区間の要素の和から前記入力データ系列の移動平均を算出する移動平均処理とを実行させることを特徴とする移動平均処理プログラム。
【請求項２】
値ｉをｍずつ増加させて前記第１の演算処理、前記第２の演算処理、前記第３の演算処理、及び前記移動平均処理を１組として繰り返し実行し、

前の組の前記第３の演算処理における（ｉ＋ｍ）番目から（ｉ＋ｘ＋ｍ－１）番目の要素についての和を、次の組の前記第３の演算処理における前記入力データ系列のｉ番目から（ｉ＋ｘ－１）番目の要素の和として演算を行うことを特徴とする請求項１記載の移動平均処理プログラム。

【請求項３】

前記入力データ系列は０番目から（ｎ－１）番目（ｎは２以上の整数）の要素を有し、

前記入力データ系列を各区間についての要素数をｘ個とするように拡張し前記移動平均を算出することを特徴とする請求項２記載の移動平均処理プログラム。

【請求項４】

前記入力データ系列の０番目から（ｎ－１）番目の要素に応じたマスクビットを生成し、

前記マスクビットに応じた要素のデータのロード処理及びストア処理を行うことを特徴とする請求項３記載の移動平均処理プログラム。

【請求項５】

算出された移動平均を格納する出力バッファを有し、

前記入力データ系列の拡張に応じて、前記出力バッファを拡張したことを特徴とする請求項３又は４記載の移動平均処理プログラム。

【請求項６】

前記移動平均を算出する各区間の有効な要素数の逆数を格納した係数テーブルを有し、

前記平均化処理では、前記第３の演算処理の結果に、前記係数テーブルを参照して得られる逆数を乗算し移動平均を算出することを特徴とする請求項１～５の何れか１項に記載の移動平均処理プログラム。

【請求項７】

入力データ系列の移動平均を複数の区間について並列に算出する演算処理部を有し、

前記演算処理部は、

第１のＳＩＭＤレジスタの各要素に対して、前記入力データ系列のｉ番目から（ｉ＋ｐ）番目（ｉは０及び自然数のうちの任意の数、ｐは０～ｍ－１の整数、ｍは２以上の整数）の要素の和を算出して前記第１のＳＩＭＤレジスタにｐ番目の要素として格納する第１の演算処理と、

第２のＳＩＭＤレジスタの各要素に対して、前記入力データ系列の（ｉ＋ｘ）番目から（ｉ＋ｘ＋ｐ）番目（ｘは自然数）の要素の和を算出して前記第２のＳＩＭＤレジスタにｐ番目の要素として格納する第２の演算処理と、

前記入力データ系列のｉ番目から（ｉ＋ｘ－１）番目の要素の和に、前記第２のＳＩＭＤレジスタのｐ番目の要素を加算し、前記第１のＳＩＭＤレジスタのｐ番目の要素を減算して第３のＳＩＭＤレジスタにｐ番目の要素として格納する処理を０番目から（ｍ－１）番目の各要素について並列に行うことにより、互いに異なるｍ個の区間について要素の和を並列に算出する第３の演算処理と、

前記第３のＳＩＭＤレジスタに格納された要素の和から平均値を算出する平均化処理とを実行することを特徴とするプロセッサ。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、移動平均処理プログラム、及びプロセッサに関する。

【背景技術】

【０００２】

近年、無線通信技術では様々な方式が標準化されており、様々な無線通信方式に対応可能なようにソフトウェア無線というプロセッサで信号処理を行うものがある。従来の無線通信処理は、ハードウェア・ロジックで大量のデータを並列処理することで、高い通信性能を実現していた。また、モバイル機器に搭載されるため、低消費電力化の点から動作周波数を高くすることができない。そのため、プロセッサを用いてソフトウェア無線処理を

行う場合も、大量のデータを並列に処理することで性能を維持する必要があり、ＳＩＭＤ（Single Instruction Multiple Data）型のプロセッサが使用されることが多い。

【０００３】

無線通信処理における信号処理のなかで、移動平均を求める処理がある。移動平均処理は、移動平均の対象とする区間の要素数をｎとすると、入力データ系列のインデックスｉに対して、ｉの値をインクリメントしながら、入力データ系列のｉ～（ｉ＋ｎ－１）番目の要素（データ）の平均値を順次求める処理である。移動平均処理は、ｉ～（ｉ＋ｎ－１）番目の要素の総和を求めた後、（ｉ＋ｎ）番目の要素を加算するとともにｉ番目の要素を減算して（ｉ＋１）～（ｉ＋ｎ）番目の要素の総和を求め、区間の要素数ｎで除算し平均値を求めることで効率的に処理を実行することができる（例えば、特許文献１参照）。

【先行技術文献】

【特許文献】

【０００４】

【特許文献１】特開平１０－１４３４９５号公報

【特許文献２】特開２０１２－７５０２３号公報

【特許文献３】特開平１－６１１１４号公報

【特許文献４】特開平１０－２８５５０２号公報

【特許文献５】特開２０１１－２３３０８５号公報

【発明の概要】

【発明が解決しようとする課題】

【０００５】

しかし、ｉ～（ｉ＋ｎ－１）番目についての演算結果に対し加減算等を行い、次の（ｉ＋１）～（ｉ＋ｎ）番目についての移動平均を求めていく移動平均処理は、データの依存関係から並列化が困難であり、ＳＩＭＤ型のプロセッサで効率的に処理することが難しい。本発明は、移動平均処理に係る演算処理を並列化して移動平均処理の処理効率を向上させることを目的とする。

【課題を解決するための手段】

【０００６】

移動平均処理プログラムの一態様は、ｍ個（ｍは２以上の整数）の演算処理を並列に実行し、かつ０番目から（ｍ－１）番目の入力要素を基に、０番目からｐ番目（ｐは０～ｍ－１の整数）の入力要素の和を算出してｐ番目の結果要素としてそれぞれ返す部分総和命令を実行するプロセッサに、以下の処理を実行させる。入力データ系列のｉ番目から（ｉ＋ｍ－１）番目（ｉは０及び自然数のうちの任意の数）の要素を０番目から（ｍ－１）番目の入力要素とする部分総和命令を実行して第１のベクトルデータを取得する第１の演算処理と、入力データ系列の（ｉ＋ｘ）番目から（ｉ＋ｘ＋ｍ－１）番目（ｘは自然数）の要素を０番目から（ｍ－１）番目の入力要素とする前記部分総和命令を実行して第２のベクトルデータを取得する第２の演算処理と、入力データ系列のｉ番目から（ｉ＋ｘ－１）番目の要素の和に、第１のベクトルデータのｐ番目の要素を減算し、第２のベクトルデータのｐ番目の要素を加算する処理を、０番目から（ｍ－１）番目の各要素について並列に行うことにより、互いに異なるｍ個の区間について要素の和を並列に算出する第３の演算処理と、算出した各区間の要素の和から入力データ系列の移動平均を算出する移動平均処理とを実行させる。

【発明の効果】

【０００７】

開示の移動平均処理プログラムは、入力データ系列の部分的な加算処理を別に計算することで、移動平均処理に係る処理を並列して実行し処理効率を向上させることができる。

【図面の簡単な説明】

【０００８】

【図１】本発明の実施形態におけるプロセッサの構成例を示す図である。

【図２】本実施形態における移動平均処理の並列化を説明するための図である。

【図３】本実施形態における部分総和命令の演算を説明するための図である。
【図４】本実施形態における部分総和命令実行時のプロセッサを示す模式図である。
【図５】本実施形態における移動平均処理（中部分処理）の例を示す図である。
【図６】本実施形態における移動平均処理（中部分処理）を示すフローチャートである。
【図７】本実施形態における移動平均処理の例を示す図である。
【図８】本実施形態におけるマスク値設定命令を説明するための図である。
【図９】本実施形態における移動平均処理でのロード処理に係るプログラムの例を示す図である。
【図１０】本実施形態における移動平均処理でのロード処理に係るプログラムの例を示す図である。
【図１１】本実施形態における移動平均処理でのストア処理に係るプログラムの例を示す図である。
【図１２】本実施形態における係数テーブルを説明するための図である。
【図１３】本実施形態における移動平均処理での平均化処理に係るプログラムの例を示す図である。
【図１４】本実施形態における移動平均処理に係るプログラムの例を示す図である。
【図１５】移動平均処理の例を示す図である。
【図１６】図１５に示す移動平均処理に係るプログラムを示す図である。
【図１７】本実施形態における部分総和命令実行時のプロセッサの他の例を示す模式図である。
【発明を実施するための形態】
【０００９】
以下、本発明の実施形態を図面に基づいて説明する。
【００１０】
図１５は、移動平均処理の例を示す図である。図１５に示す移動平均処理は、入力バッファin_bufに格納されている１つの入力データ系列に対して移動平均を求め、求めた移動平均を出力バッファout_bufに格納する。移動平均の対象とする区間は、入力データ系列における前部分、中部分、後部分で異なる。区間の要素数は、前部分では初期区間をａとして、その後、順次ｘまで１ずつ増加し、中部分ではｘであり、後部分では最終区間をｂまで１ずつ減少する。
【００１１】
入力データ系列における前部分では、入力バッファin_buf[0]～in_buf[a-1]のａ個の要素（入力データ）の平均値が出力バッファout_buf[0]に格納され、入力バッファin_buf[0]～in_buf[a]の（ａ＋１）個の要素の平均値が出力バッファout_buf[1]に格納される。要素数が増加する、入力データ系列における前部分では、新規の要素を加算するのみで最古の要素を減算せず、除数を１ずつ増加させて移動平均を求める。このようにして移動平均を順次求めていき、入力バッファin_buf[0]～in_buf[x-2]の（ｘ－１）個の要素の平均値が出力バッファout_buf[x-a-1]に格納される。
【００１２】
また、入力データ系列における中部分では、入力バッファin_buf[0]～in_buf[x-1]のｘ個の要素の平均値が出力バッファout_buf[x-a]に格納され、入力バッファin_buf[1]～in_buf[x]のｘ個の要素の平均値が出力バッファout_buf[x-a+1]に格納される。入力データ系列における中部分では、新規の要素を加算し、最古の要素を減算して、ｘで除算して移動平均を求める。このようにして移動平均を順次求めていき、入力バッファin_buf[n-x]～in_buf[n-1]のｘ個の要素の平均値が出力バッファout_buf[n-a]に格納される。
【００１３】
また、入力データ系列における後部分では、入力バッファin_buf[n-x+1]～in_buf[n-1]の（ｘ－１）個の要素の平均値が出力バッファout_buf[n-a+1]に格納される。要素数が減少する、入力データ系列における後部分では、最古の要素を減算するのみで新規の要素を加算せず、除数を１ずつ減少させて移動平均を求める。このようにして移動平均を順次求

めていき、入力バッファin_buf[n-b-1]～in_buf[n-1]の（ｂ＋１）個の要素の平均値が出力バッファout_buf[n+x-a-b-1]に格納され、入力バッファin_buf[n-b]～in_buf[n-1]のｂ個の要素の平均値が出力バッファout_buf[n+x-a-b]に格納される。

【００１４】

前述の図１５に示した移動平均処理に係るプログラム例を図１６に示す。事前処理１６０１では、入力バッファin_buf[0]から入力バッファin_buf[a-2]までの各要素を加算して総和値sumが算出される。前部分処理１６０２では、新規の要素である入力バッファin_buf[i]の要素を総和値sumに加算した後にカウント値cntで除算して求めた平均値avrが、出力バッファout_buf[j]に格納される。中部分処理１６０３では、総和値sumに新規の要素である入力バッファin_buf[i]の要素を加算し、最古の要素である入力バッファin_buf[i-x]の要素を減算した後に値xで除算して求めた平均値avrが、出力バッファout_buf[j]に格納される。後部分処理１６０４では、最古の要素である入力バッファin_buf[i-x]の要素を総和値sumから減算した後にカウント値cntで除算して求めた平均値avrが、出力バッファout_buf[j]に格納される。

【００１５】

図１６に示したような移動平均処理では、前の演算結果に対して、新規の要素を１つ加算したり最古の要素を１つ減算したりするために、演算処理を並列化することが困難であり、ＳＩＭＤ型のプロセッサで効率良く処理することが難しい。以下に説明する実施形態では、入力データ系列の部分的な加算処理を別個に計算することで、移動平均処理に係る処理を並列して実行可能にし、処理効率の向上を図る。

【００１６】

図１は、本発明の実施形態におけるプロセッサの構成例を示す図である。本実施形態におけるプロセッサ１０は、例えば並列に演算処理を実行可能なＳＩＭＤ型のプロセッサであり、ｍ個のデータを並列処理するＳＩＭＤ型のプロセッサを一例として示している。プロセッサ１０は、プログラムカウンタ（ＰＣ）１２、命令デコーダ（ＤＥＣＯＤＥＲ）１４、ＳＩＭＤレジスタファイル１６、ＳＩＭＤパイプライン・レジスタ１８Ａ、１８Ｂ、２２、及び演算処理部２１を有する。演算処理部２１は、複数の演算器（ＡＬＵ）２０－０、２０－１、…、２０－（ｍ－１）を有する。

【００１７】

プロセッサ１０は、命令メモリ（ＩＲＡＭ）３０からプログラムカウンタ１２のカウント値に応じて読み出された命令を命令デコーダ１４でデコードする。そして、プロセッサ１０は、命令デコーダ１４でのデコード結果に応じて、演算処理に用いるデータをレジスタファイル１６のＳＩＭＤレジスタから読み出してＳＩＭＤパイプライン・レジスタ１８Ａ、１８Ｂに格納する。演算処理に必要なデータ等は、データメモリ（ＤＲＡＭ）４０から適宜読み出されてレジスタファイル１６のＳＩＭＤレジスタに格納されている。

【００１８】

また、プロセッサ１０は、ＳＩＭＤパイプライン・レジスタ１８Ａ、１８Ｂに格納したデータを用い、命令デコーダ１４でのデコード結果に応じた演算処理を演算処理部２１の演算器２０－０、２０－１、…、２０－（ｍ－１）で実行し、演算結果をＳＩＭＤパイプライン・レジスタ２２に格納する。ＳＩＭＤパイプライン・レジスタ２２に格納された演算結果は、ＳＩＭＤレジスタファイル１６に書き込まれる。

【００１９】

ここで、入力データ系列における中部分の移動平均処理でのｘ個の要素の平均値を求める処理に注目すると、ｘ個の要素の和を求める演算は、図２（Ａ）及び（Ｂ）に示すように新規の要素を加算し、最古の要素を減算する演算を行う。本実施形態では、図２（Ｂ）に示すように、ｘ個の要素の和を求める演算をｍ並列で行うために、各演算において減算する古い要素の総和２０１及び加算する新しい要素の総和２０２を別に計算し、それを用いてｘ個の要素の和を求めることで並列化する。

【００２０】

図２に示す例は、ｘ個の要素の和を求める演算を４並列で行う例であり、減算する古い

要素の総和２０１として、ｔ番目の要素、ｔ番目と（ｔ＋１）番目の要素の総和、ｔ番目から（ｔ＋２）番目の要素の総和、及びｔ番目から（ｔ＋３）番目の要素の総和を求める。また、加算する新しい要素の総和２０２として、ｕ番目の要素、ｕ番目と（ｕ＋１）番目の要素の総和、ｕ番目から（ｕ＋２）番目の要素の総和、及びｕ番目から（ｕ＋３）番目の要素の総和を求める。

【００２１】

そして、ｔ番目から（ｕ－１）番目の要素の総和に対して、古い要素の総和２０１を減算する処理及び新しい要素の総和２０２を加算する処理をそれぞれ並列に行うことでｘ個の要素の和を求める演算を並列化する。この減算する古い要素の総和や加算する新しい要素の総和を求めるために、本実施形態では、ｍ個のデータを並列処理可能なプロセッサにおいて、０～（ｍ－１）番目の要素の部分総和を返す部分総和命令を設ける。

【００２２】

部分総和命令は、ＳＩＭＤレジスタｖｓに格納されているｍ個の要素（データ）から部分総和を算出し、ＳＩＭＤレジスタｖｒに格納する命令である。すなわち、部分総和命令を実行すると、ＳＩＭＤレジスタｖｓの０番目の入力要素がベクトルデータの０番目の結果要素としてＳＩＭＤレジスタｖｒに格納され、ＳＩＭＤレジスタｖｓの０番目と１番目の入力要素の加算結果がベクトルデータの１番目の結果要素としてＳＩＭＤレジスタｖｒに格納される。また、ＳＩＭＤレジスタｖｓの０番目からｍ番目の入力要素の加算結果がベクトルデータのｍ番目の結果要素としてＳＩＭＤレジスタｖｒに格納される。すなわち、部分総和命令では、ＳＩＭＤレジスタｖｓの０番目からｋ番目（ｋは整数）の入力要素の加算結果がベクトルデータのｋ番目の結果要素としてＳＩＭＤレジスタｖｒに格納される。

【００２３】

図３（Ａ）は、ｍ＝４としたときの部分総和命令の演算処理を行う演算回路の構成例を示す図である。ＳＩＭＤレジスタｖｓの０番目の要素vs[0]がＳＩＭＤレジスタｖｒの０番目の要素vr[0]に入力される。加算器３０１は、ＳＩＭＤレジスタｖｓの０番目の要素vs[0]と１番目の要素vs[1]が入力され、その出力がＳＩＭＤレジスタｖｒの１番目の要素vr[1]に入力される。加算器３０２は、加算器３０１の出力とＳＩＭＤレジスタｖｓの２番目の要素vs[2]が入力され、その出力がＳＩＭＤレジスタｖｒの２番目の要素vr[2]に入力される。また、加算器３０３は、加算器３０２の出力とＳＩＭＤレジスタｖｓの３番目の要素vs[3]が入力され、その出力がＳＩＭＤレジスタｖｒの３番目の要素vr[3]に入力される。このようにして図３（Ａ）に示す演算回路は、ｍ＝４としたときの部分総和命令に応じた図３（Ｂ）に示す演算処理を実行する。

【００２４】

なお、部分総和命令の演算処理を行う演算回路の構成は、図３（Ａ）に示した回路に限定されるものではなく、図３（Ｂ）に示す演算処理を実行可能な回路構成であれば良い。例えば、図３（Ｃ）に示すような回路でも良い。図３（Ｃ）は、ｍ＝４としたときの部分総和命令の演算処理を行う演算回路の他の構成例を示す図である。ＳＩＭＤレジスタｖｓの０番目の要素vs[0]がＳＩＭＤレジスタｖｒの０番目の要素vr[0]に入力される。加算器３１１は、ＳＩＭＤレジスタｖｓの０番目の要素vs[0]と１番目の要素vs[1]が入力される。加算器３１１の出力がＳＩＭＤレジスタｖｒの１番目の要素vr[1]に入力される。加算器３１２は、加算器３１１の出力とＳＩＭＤレジスタｖｓの２番目の要素vs[2]が入力される。加算器３１２の出力がＳＩＭＤレジスタｖｒの２番目の要素vr[2]に入力される。また、加算器３１３は、ＳＩＭＤレジスタｖｓの２番目の要素vs[2]と３番目の要素vs[3]が入力される。加算器３１４は、加算器３１１の出力と加算器３１３の出力が入力される。加算器３１４の出力がＳＩＭＤレジスタｖｒの３番目の要素vr[3]に入力される。図３（Ｃ）に示した回路構成では、部分総和命令の演算処理に係る加算の段数が低減できる。

【００２５】

図４は、本実施形態における部分総和命令実行時のプロセッサを示す模式図である。図４においても、ｍ＝４としたときの部分総和命令実行時のプロセッサを一例として示して

いる。図４において、図１に示した構成要素と同一の機能を有する構成要素には同一の符号を付し、重複する説明は省略する。演算処理部２１が有する複数の演算器により複数の加算器（ＡＤＤ）２４－１、２４－２、２４－３が実現される。

【００２６】

ＳＩＭＤパイプライン・レジスタ１８の０番目の要素が、ＳＩＭＤパイプライン・レジスタ２２の０番目の要素として入力される。ＳＩＭＤパイプライン・レジスタ１８の１番目の要素と０番目の要素が加算器２４－１に入力され、加算器２４－１の出力がＳＩＭＤパイプライン・レジスタ２２の１番目の要素として入力される。ＳＩＭＤパイプライン・レジスタ１８の２番目の要素と加算器２４－１の出力が加算器２４－２に入力され、加算器２４－２の出力がＳＩＭＤパイプライン・レジスタ２２の２番目の要素として入力される。ＳＩＭＤパイプライン・レジスタ１８の３番目の要素と加算器２４－２の出力が加算器２４－３に入力され、加算器２４－３の出力がＳＩＭＤパイプライン・レジスタ２２の３番目の要素として入力される。なお、図４においては、演算処理部２１において３段の加算処理が行われるが１サイクルで実行可能である。

【００２７】

図１７は、本実施形態における部分総和命令実行時のプロセッサの他の例を示す模式図である。図１７においても、ｍ＝４としたときの部分総和命令実行時のプロセッサを一例として示しており、図４（Ｃ）に示したように部分総和命令の演算処理を行う例に対応するものである。図１７において、図１、図４に示した構成要素と同一の機能を有する構成要素には同一の符号を付し、重複する説明は省略する。演算処理部２１Ｂが有する複数の演算器により複数の加算器（ＡＤＤ）２４Ｂ－１、２４Ｂ－２、２４Ｂ－３、２４Ｂ－４が実現される。

【００２８】

ＳＩＭＤパイプライン・レジスタ１８の０番目の要素が、ＳＩＭＤパイプライン・レジスタ２２の０番目の要素として入力される。ＳＩＭＤパイプライン・レジスタ１８の１番目の要素と０番目の要素が加算器２４Ｂ－１に入力され、加算器２４Ｂ－１の出力がＳＩＭＤパイプライン・レジスタ２２の１番目の要素として入力される。ＳＩＭＤパイプライン・レジスタ１８の２番目の要素と加算器２４Ｂ－１の出力が加算器２４Ｂ－２に入力され、加算器２４Ｂ－２の出力がＳＩＭＤパイプライン・レジスタ２２の２番目の要素として入力される。また、ＳＩＭＤパイプライン・レジスタ１８の３番目の要素と２番目の要素が加算器２４Ｂ－３に入力される。加算器２４Ｂ－１の出力と加算器２４Ｂ－３の出力が加算器２４Ｂ－４に入力され、加算器２４Ｂ－４の出力がＳＩＭＤパイプライン・レジスタ２２の３番目の要素として入力される。なお、図１７においても、演算処理部２１における加算処理は１サイクルで実行可能である。

【００２９】

図５は、本実施形態における移動平均処理（中部分処理）の例を示す図であり、ｍ＝４（４並列）の場合を例として示している。プロセッサ１０は、部分総和命令によりｔ～（ｔ＋３）番目の要素の部分総和及びｕ～（ｕ＋３）番目の要素の部分総和をそれぞれ求める。そして、プロセッサ１０は、図５（Ａ）に示す１つ前での最後の要素の演算結果であるｔ番目から（ｕ－１）番目の要素の総和に対して、部分総和命令により求めたｕ～（ｕ＋３）番目の要素の部分総和を加算する（（図５（Ｂ））。さらに、プロセッサ１０は、図５（Ｂ）に示す加算結果に対して、部分総和命令により求めたｔ～（ｔ＋３）番目の要素の部分総和を減算する（（図５（Ｃ））。これにより、（ｔ＋１）番目からｕ番目の要素の総和を求める演算、（ｔ＋２）番目から（ｕ＋１）番目の要素の総和を求める演算、（ｔ＋３）番目から（ｕ＋２）番目の要素の総和を求める演算、及び（ｔ＋４）番目から（ｕ＋３）番目の要素の総和を求める演算を並列して実行することができる。

【００３０】

同様にして、プロセッサ１０は、部分総和命令により（ｔ＋４）～（ｔ＋７）番目の要素の部分総和及び（ｕ＋４）～（ｕ＋７）番目の要素の部分総和をそれぞれ求める。そして、プロセッサ１０は、図５（Ｄ）に示す１つ前（図５（Ｃ））での最後の要素の演算結

果である（ｔ＋４）番目から（ｕ＋３）番目の要素の総和に対して、部分総和命令により求めた（ｕ＋４）～（ｕ＋７）番目の要素の部分総和を加算する（（図５（Ｅ））。さらに、プロセッサ１０は、図５（Ｅ）に示す加算結果に対して、部分総和命令により求めた（ｔ＋４）～（ｔ＋７）番目の要素の部分総和を減算する（（図５（Ｆ））。これにより、（ｔ＋５）番目から（ｕ＋４）番目の要素の総和を求める演算、（ｔ＋６）番目から（ｕ＋５）番目の要素の総和を求める演算、（ｔ＋７）番目から（ｕ＋６）番目の要素の総和を求める演算、及び（ｔ＋８）番目から（ｕ＋７）番目の要素の総和を求める演算を並列して実行することができる。

【００３１】

例えば、図５に示したように要素の総和を求める演算を４並列で実行すると、中部分処理での処理効率は４倍になる。なお、図５においては、１つ前での演算結果に対して新しい要素の部分総和を加算した後に古い要素の部分総和を減算する例を示したが、１つ前での演算結果に対して古い要素の部分総和を減算した後に新しい要素の部分総和を加算するようにしても良い。

【００３２】

図６は、本実施形態における移動平均処理（中部分処理）を示すフローチャートである。プロセッサ１０は、前部分処理が終了して中部分処理に進むと、ステップＳ６０１にて、前部分処理における最後の要素の演算結果をＳＩＭＤレジスタｖｒ０の各要素vr0[0]～vr0[m-1]に入力する。次に、ステップＳ６０２にて、プロセッサ１０は、中部分処理で処理すべき要素のうち、未処理の要素が存在するか否かを判定する。その結果、未処理の要素が存在しなければ、中部分処理に係る計算を終了して後部分処理に進む。一方、未処理の要素が存在する場合には、ステップＳ６０３へ進む。

【００３３】

ステップＳ６０３にて、プロセッサ１０は、部分総和命令を実行して、加算するｕ～（ｕ＋ｍ－１）番目の新しい要素の部分総和を求め、ＳＩＭＤレジスタｖｒ１の各要素vr1[0]～vr1[m-1]に入力する。また、ステップＳ６０４にて、プロセッサ１０は、部分総和命令を実行して、減算するｔ～（ｔ＋ｍ－１）番目の古い要素の部分総和を求め、ＳＩＭＤレジスタｖｒ２の各要素vr2[0]～vr2[m-1]に入力する。なお、ステップＳ６０３及びステップＳ６０４の処理は順不同である。

【００３４】

次に、ステップＳ６０５にて、プロセッサ１０は、対応する要素毎にＳＩＭＤレジスタｖｒ０にＳＩＭＤレジスタｖｒ１を加算しＳＩＭＤレジスタｖｒ２を減算する処理を行い、演算結果をＳＩＭＤレジスタｖｒ３の各要素vr3[0]～vr3[m-1]に入力する。続いて、ステップＳ６０６にて、プロセッサ１０は、ＳＩＭＤレジスタｖｒ３の各要素vr3[0]～vr3[m-1]に対して区間に含まれる要素数に応じた平均化処理を行って平均値を算出し、所定の記憶領域に書き込む。そして、ステップＳ６０７にて、プロセッサ１０は、ＳＩＭＤレジスタｖｒ３の最後の要素vr3[m-1]をＳＩＭＤレジスタｖｒ０の各要素vr0[0]～vr0[m-1]に入力して、ステップＳ６０２に戻る。

【００３５】

以上説明したように、ｍ個のデータを並列処理可能なプロセッサにおいて、入力データ系列の部分的な０～（ｍ－１）番目の要素の部分総和を返す部分総和命令を設けることで、移動平均処理での区間の要素の和を求める演算を並列して実行することができる。これにより、移動平均処理に係る処理を並列に行うことを可能にし処理効率を向上させることができ、例えばＳＩＭＤ型のプロセッサで移動平均処理を効率良く処理することが可能になる。

【００３６】

ここで、前述した図１５に示したような移動平均処理では、入力データ系列における前部分、中部分、後部分で区間の要素数が変化している。そのため、図１６に示したように条件分岐が発生するとともに、それぞれのループ処理が小さくなりループ制御に係るオーバーヘッドが大きくなり、処理効率が低下してしまう。そこで、本実施形態では、各区間

の要素数をｘとして演算処理が実行できるように、図７に示すように入力バッファin_buf及び出力バッファout_bufを仮想的に拡張して、移動平均処理に係る処理効率をさらに向上させる。

【００３７】

図７は、本実施形態における移動平均処理の例を示す図である。入力バッファin_bufの配列をインデックスが（－ｘ）まで負側に仮想的に拡張し、入力バッファin_bufの配列をインデックスが（ｎ＋ｘ－ｂ－１）まで正側に仮想的に拡張する。また、出力バッファout_bufの配列をインデックスが（－ａ＋１）まで負側に仮想的に拡張する。

【００３８】

このように入力バッファin_bufの配列を仮想的に拡張することで、入力データ系列における前部分、中部分、後部分での移動平均処理における区間の要素数をｘに揃えることができる。また、出力バッファout_bufの配列を仮想的に拡張することで、事前処理についても各区間で平均値を求める通常の処理と同様の処理で実行することができる。したがって、移動平均処理において条件分岐がなくなり、また１つのループ処理で移動平均処理が実行可能になるので、移動平均処理に係る処理効率を向上させることができる。

【００３９】

ここで、拡張された入力バッファin_bufの配列における負のインデックス、すなわち入力バッファin_bufの（－ｘ）番目から（－１）番目の各要素については、マスク付きのロード命令により読み出しを抑制して０とする。同様に、拡張された入力バッファin_bufにおけるｎ以上のインデックス、すなわち入力バッファin_bufのｎ番目から（ｎ＋ｘ－ｂ－１）番目の各要素については、マスク付きのロード命令により読み出しを抑制して０とする。また、拡張された出力バッファout_bufの配列における負のインデックス、すなわち出力バッファout_bufの（－ａ＋１）番目から（－１）番目の各要素については、マスク付きのストア命令により書き込みを抑制する。

【００４０】

マスク付きのロード命令は、マスクレジスタｍｒの上位側からｘビット目の値が０の場合には０を、値が１の場合にはロードデータをＳＩＭＤレジスタのｘ番目の要素に格納する。また、マスク付きのストア命令は、マスクレジスタｍｒの上位側からｘビット目の値が１の場合のみ、ＳＩＭＤレジスタのｘ番目の要素のストア処理を行う（値が０の場合には何もしない）。なお、本実施形態では、ｍビットのマスクレジスタｍｒにおいて、最上位のビットを０ビット目とし、最下位のビットを（ｍ－１）ビット目とする。

【００４１】

マスクレジスタｍｒに値を設定するマスク値設定命令、及びマスク値設定命令__Ｒについて説明する。図８（Ａ）は、マスク値設定命令を説明するための図である。マスク値設定命令は、値がｋ（０＜ｋ＜ｍ）の場合に、マスクレジスタｍｒの０ビット目から（ｋ－１）ビット目までの値を１に設定し、ｋビット目から（ｍ－１）ビット目までの値を０に設定する。すなわち、マスク値設定命令は、値がｋの場合には、マスクレジスタｍｒの上位側からｋビットの値を１に設定し、残りの（ｍ－ｋ）ビットの値を０に設定する。また、マスク値設定命令は、値が０以下の場合には、マスクレジスタｍｒのすべてのビットの値を０に設定し、値がｍ以上の場合には、マスクレジスタｍｒのすべてのビットの値を１に設定する。

【００４２】

図８（Ｂ）は、マスク値設定命令__Ｒを説明するための図である。マスク値設定命令__Ｒは、値がｋ（０＜ｋ＜ｍ）の場合に、マスクレジスタｍｒの０ビット目から（ｍ－ｋ－１）ビット目までの値を０に設定し、（ｍ－ｋ）ビット目から（ｍ－１）ビット目までの値を１に設定する。すなわち、マスク値設定命令__Ｒは、値がｋの場合には、マスクレジスタｍｒの下位側からｋビットの値を１に設定し、残りの（ｍ－ｋ）ビットの値を０に設定する。また、マスク値設定命令__Ｒは、値が０以下の場合には、マスクレジスタｍｒのすべてのビットの値を０に設定し、値がｍ以上の場合には、マスクレジスタｍｒのすべてのビットの値を１に設定する。

【００４３】
例えば、図９に示すプログラムにより、図７に示した領域７０１に含まれる入力バッファin_bufの（－ｘ）番目から（－１）番目に対応する各要素が０とされる。図９において、コード９０１の処理は、マスク値設定命令__Ｒにより値（ｉｄｘ０＋ｍ）に応じたマスク値をマスクレジスタｍ０に設定する。また、コード９０２の処理は、マスクレジスタｍ０の値に応じて、入力バッファin_bufの（ｉｄｘ０）番目から（ｉｄｘ０＋ｍ－１）番目に対応する要素をロードしてＳＩＭＤレジスタｖｔに格納する。また、コード９０３の処理は、値ｉｄｘ０にｍを加算し新たな値ｉｄｘ０とする。この図９に示したプログラムを実行することで、入力バッファin_bufの（－ｘ）番目から（－１）番目に対応する各要素は０とされ、入力バッファin_bufの０番目以降に対応する各要素はロードされてＳＩＭＤレジスタｖｔに格納される。

【００４４】
また、図１０に示すプログラムにより、図７に示した領域７０２に含まれる入力バッファin_bufのｎ番目から（ｎ＋ｘ－ｂ－１）番目に対応する各要素が０とされる。図１０において、コード１００１の処理は、マスク値設定命令により値（ｉｄｘ１）に応じたマスク値をマスクレジスタｍ１に設定する。また、コード１００２の処理は、マスクレジスタｍ１の値に応じて、入力バッファin_bufのｉ番目から（ｉ＋ｍ－１）番目に対応する要素をロードしてＳＩＭＤレジスタｖｕに格納する。また、コード１００３の処理は、値ｉｄｘ１からｍを減算し新たな値ｉｄｘ１とする。この図１０に示したプログラムを実行することで、入力バッファin_bufのｎ番目以降に対応する各要素は０とされ、入力バッファin_bufの０番目から（ｎ－１）番目に対応する各要素はロードされてＳＩＭＤレジスタｖｕに格納される。

【００４５】
また、図１１に示すプログラムにより、図７に示した領域７０３に含まれる出力バッファの（－ａ＋１）番目から（－１）番目に対応する要素のストア処理が抑制される。図１１において、コード１１０１の処理は、マスク値設定命令__Ｒにより値（ｉｄｘ２＋ｍ）に応じたマスク値をマスクレジスタｍ２に設定する。また、コード１１０２の処理は、マスクレジスタｍ２の値に応じて、出力バッファout_bufの（ｉｄｘ２）番目から（ｉｄｘ２＋ｍ－１）番目に対応する要素に移動平均結果をストアするストア処理を行う。また、コード１１０３の処理は、値ｉｄｘ２にｍを加算し新たな値ｉｄｘ２とする。この図１１に示したプログラムを実行することで、出力バッファout_bufの（－ａ＋１）番目から（－１）番目に対応する各要素に係るストア処理は抑制され、出力バッファout_bufの０番目以降に対応する各要素について演算結果のストア処理が行われる。

【００４６】
移動平均処理においては、平均値を算出するために平均化処理を行うが、この平均化処理を除算により行うと処理に要するサイクル数が大きくなってしまう。また、入力データ系列における前部分、中部分、後部分で、有効な要素数が異なる。そこで、本実施形態では、平均値を求めるための除数の逆数が格納された係数テーブルを設ける。そして、係数テーブルから得られる値ｉに応じた除数の逆数を、区間中の要素の総和に乗算することで平均値を算出するように、処理効率を向上させる。

【００４７】
図１２は、本実施形態における係数テーブルに格納される値ｉに応じた除数及びその逆数を示した図である。値ｉが０～（ｘ－１）であるときは、入力データ系列における前部分（事前処理を含む）に相当し、有効な要素数、すなわち除数は（ｉ＋１）であり、その逆数は１／（ｉ＋１）である。また、値ｉがｘ～（ｎ－１）であるときは、入力データ系列における中部分に相当し、有効な要素数、すなわち除数はｘであり、その逆数は１／ｘである。また、値ｉがｎより大きいときは、入力データ系列における後部分に相当し、有効な要素数、すなわち除数は（ｘ＋ｎ－ｉ－１）であり、その逆数は１／（ｘ＋ｎ－ｉ－１）である。

【００４８】

したがって、図１２に示した値ｉに応じた逆数を、ｉ番目の要素として格納した係数テーブルdiv_tblを作成し、図１３に示すプログラムにより係数テーブルdiv_tblから値ｉに応じた逆数を取得して区間の要素の総和に乗算することで平均値を算出することができる。図１３において、コード１３０１の処理は、係数テーブルdiv_tblのｉ番目から（ｉ＋ｍ－１）番目の要素をロードしてＳＩＭＤレジスタｖｄに格納する。また、コード１３０２の処理は、対応する要素毎に、ＳＩＭＤレジスタｖｓｕｍに格納されている区間の要素の総和とＳＩＭＤレジスタｖｄに格納されている値とを乗算して得られる平均値を、ＳＩＭＤレジスタｖａｖｅに格納する。

【００４９】

以上説明した各処理を適用した本実施形態における移動平均処理に係るプログラムの例を図１４に示す。この図１４において、図９～図１１、図１３に示した構成要素と同一の構成要素には同一の符号を付し、重複する説明は省略する。なお、図１４においては、区間の要素の総和を求める際に減算される部分総和が格納されるＳＩＭＤレジスタをｖｔ２とし、その部分総和を求めるための入力バッファin_bufからのロードデータが格納されるＳＩＭＤレジスタをｖｔ１としている。同様に、区間の要素の総和を求める際に加算される部分総和が格納されるＳＩＭＤレジスタをｖｕ２とし、その部分総和を求めるための入力バッファin_bufからのロードデータが格納されるＳＩＭＤレジスタをｖｕ１としている。

【００５０】

図１４において、コード１４０１の処理は、部分総和命令によりＳＩＭＤレジスタｖｔ１に格納されている値を基に部分総和を算出してＳＩＭＤレジスタｖｔ２に格納する。また、コード１４０２の処理は、部分総和命令によりＳＩＭＤレジスタｖｕ１に格納されている値を基に部分総和を算出してＳＩＭＤレジスタｖｕ２に格納する。コード１４０３の処理は、対応する要素毎に、ＳＩＭＤレジスタｖｓｕｍの値にＳＩＭＤレジスタｖｕ２の値を加算するとともにＳＩＭＤレジスタｖｔ２の値を減算して区間の要素の総和を算出し、ＳＩＭＤレジスタｖｓｕｍに格納する（上書きする）。コード１４０４の処理は、ＳＩＭＤレジスタｖｓｕｍの最後の要素vsum[m-1]をＳＩＭＤレジスタｖｓｕｍの各要素に格納する（上書きする）。

【００５１】

なお、前記実施形態は、何れも本発明を実施するにあたっての具体化のほんの一例を示したものに過ぎず、これらによって本発明の技術的範囲が限定的に解釈されてはならないものである。すなわち、本発明はその技術思想、またはその主要な特徴から逸脱することなく、様々な形で実施することができる。

本発明の諸態様を付記として以下に示す。

【００５２】

（付記１）

ｍ個（ｍは２以上の整数）の演算処理を並列に実行し、かつ０番目から（ｍ－１）番目の入力要素を基に、０番目からｐ番目（ｐは０～ｍ－１の整数）の前記入力要素の和を算出してｐ番目の結果要素としてそれぞれ返す部分総和命令を実行するプロセッサに、

入力データ系列のｉ番目から（ｉ＋ｍ－１）番目（ｉは０及び自然数のうちの任意の数）の要素を０番目から（ｍ－１）番目の前記入力要素とする前記部分総和命令を実行して第１のベクトルデータを取得する第１の演算処理と、

前記入力データ系列の（ｉ＋ｘ）番目から（ｉ＋ｘ＋ｍ－１）番目（ｘは自然数）の要素を０番目から（ｍ－１）番目の前記入力要素とする前記部分総和命令を実行して第２のベクトルデータを取得する第２の演算処理と、

前記入力データ系列のｉ番目から（ｉ＋ｘ－１）番目の要素の和に、前記第１のベクトルデータのｐ番目の要素を減算し、前記第２のベクトルデータのｐ番目の要素を加算する処理を、０番目から（ｍ－１）番目の各要素について並列に行うことにより、互いに異なるｍ個の区間について要素の和を並列に算出する第３の演算処理と、

算出した各区間の要素の和から前記入力データ系列の移動平均を算出する移動平均処理

とを実行させることを特徴とする移動平均処理プログラム。
（付記２）
値ｉをｍずつ増加させて前記第１の演算処理、前記第２の演算処理、前記第３の演算処理、及び前記移動平均処理を１組として繰り返し実行し、
前の組の前記第３の演算処理における（ｉ＋ｍ）番目から（ｉ＋ｘ＋ｍ－１）番目の要素についての和を、次の組の前記第３の演算処理における前記入力データ系列のｉ番目から（ｉ＋ｘ－１）番目の要素の和として演算を行うことを特徴とする付記１記載の移動平均処理プログラム。
（付記３）
前記入力データ系列は０番目から（ｎ－１）番目（ｎは２以上の整数）の要素を有し、
前記入力データ系列を各区間についての要素数をｘ個とするように拡張し前記移動平均を算出することを特徴とする付記２記載の移動平均処理プログラム。
（付記４）
前記入力データ系列の０番目から（ｎ－１）番目の要素に応じたマスクビットを生成し、
前記マスクビットに応じた要素のデータのロード処理及びストア処理を行うことを特徴とする付記３記載の移動平均処理プログラム。
（付記５）
前記入力データ系列を（－ｘ）番目まで拡張し、前記入力データ系列の（－ｘ）番目から（－１）番目の要素については前記マスクビットにより要素のロード処理を抑制することを特徴とする付記４記載の移動平均処理プログラム。
（付記６）
前記移動平均を求める最終の区間の要素数がｂ（ｂはｘ以下の任意の整数）である場合に、入力データ系列を（ｎ＋ｘ－ｂ－１）番目まで拡張し、入力データ系列のｎ番目から（ｎ＋ｘ－ｂ－１）番目までの要素のロード処理を抑制することを特徴とする付記４記載の移動平均処理プログラム。
（付記７）
算出された移動平均を格納する出力バッファを有し、
前記入力データ系列の拡張に応じて、前記出力バッファを拡張したことを特徴とする付記３記載の移動平均処理プログラム。
（付記８）
前記移動平均を求める初期の区間の要素数がａ（ａはｘ以下の任意の整数）である場合に、前記出力バッファの系列を（－ａ＋１）番目まで拡張し、
前記出力バッファの系列の（－ａ＋１）番目から（－１）番目の要素のストア処理を抑制することを特徴とする付記４記載の移動平均処理プログラム。
（付記９）
前記移動平均を算出する各区間の有効な要素数の逆数を格納した係数テーブルを有し、
前記平均化処理では、前記第３の演算処理の結果に、前記係数テーブルを参照して得られる逆数を乗算し移動平均を算出することを特徴とする付記１記載の移動平均処理プログラム。
（付記１０）
前記プロセッサは、ＳＩＭＤ型のプロセッサであることを特徴とする付記４記載の移動平均処理プログラム。
（付記１１）
入力データ系列の移動平均を複数の区間について並列に算出する演算処理部を有し、
前記演算処理部は、
第１のＳＩＭＤレジスタの各要素に対して、前記入力データ系列のｉ番目から（ｉ＋ｐ）番目（ｉは０及び自然数のうちの任意の数、ｐは０～ｍ－１の整数、ｍは２以上の整数）の要素の和を算出して前記第１のＳＩＭＤレジスタにｐ番目の要素として格納する第１の演算処理と、

第２のＳＩＭＤレジスタの各要素に対して、前記入力データ系列の（ｉ＋ｘ）番目から（ｉ＋ｘ＋ｐ）番目（ｘは自然数）の要素の和を算出して前記第２のＳＩＭＤレジスタにｐ番目の要素として格納する第２の演算処理と、

前記入力データ系列のｉ番目から（ｉ＋ｘ－１）番目の要素の和に、前記第２のＳＩＭＤレジスタのｐ番目の要素を加算し、前記第１のＳＩＭＤレジスタのｐ番目の要素を減算して第３のＳＩＭＤレジスタにｐ番目の要素として格納する処理を０番目から（ｍ－１）番目の各要素について並列に行うことにより、互いに異なるｍ個の区間について要素の和を並列に算出する第３の演算処理と、

前記第３のＳＩＭＤレジスタに格納された要素の和から平均値を算出する平均化処理とを実行することを特徴とするプロセッサ。

（付記１２）

前記入力データ系列は０番目から（ｎ－１）番目（ｎは２以上の整数）の要素を有し、

前記入力データ系列を各区間についての要素数をｘ個とするように拡張し前記平均値を算出することを特徴とする付記１１記載のプロセッサ。

（付記１３）

前記入力データ系列の０番目から（ｎ－１）番目の要素に応じたマスクビットを生成し、

前記マスクビットに応じた要素のデータのロード処理及びストア処理を行うことを特徴とする付記１２記載のプロセッサ。

（付記１４）

前記移動平均を算出する各区間の有効な要素数の逆数を格納した係数テーブルを有し、

前記平均化処理では、前記第３の演算処理の結果に、前記係数テーブルを参照して得られる逆数を乗算し移動平均を算出することを特徴とする付記１１記載のプロセッサ。

（付記１５）

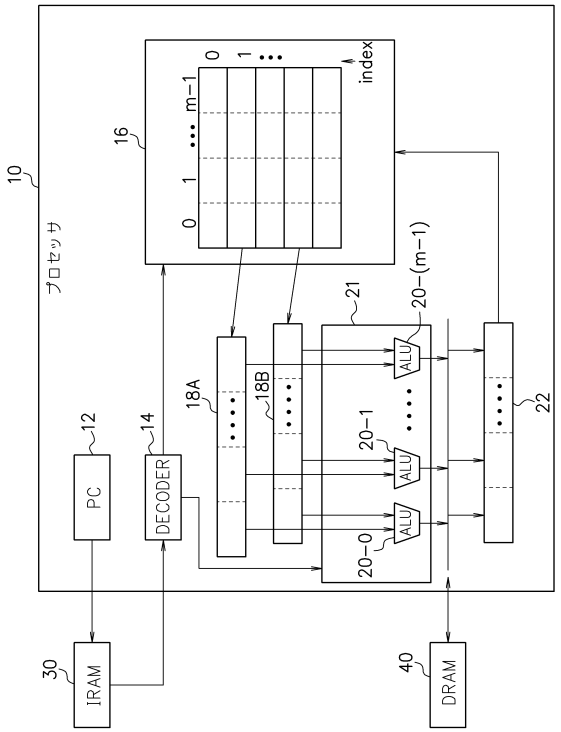
前記プロセッサは、ＳＩＭＤ型のプロセッサであることを特徴とする付記１１記載のプロセッサ。

【符号の説明】

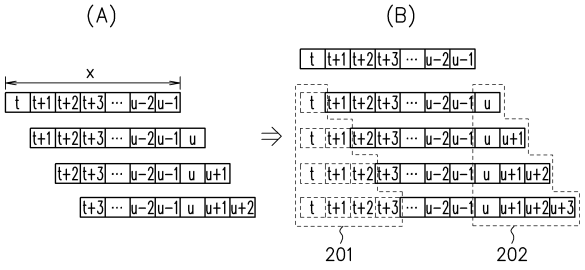
【００５３】

１０　プロセッサ
１２　プログラムカウンタ
１４　命令デコーダ
１６　レジスタファイル
１８、２２　ＳＩＭＤレジスタ
２０　演算器
２１　演算処理部
２４　加算器
３０　命令メモリ
４０　データメモリ

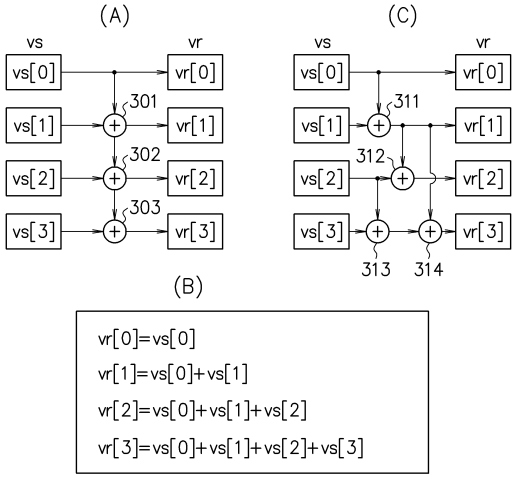
【図１】


10
プロセッサ
16
index
0
1
m−1
21
20−(m−1)
20−1
20−0
ALU
18A
18B
12
PC
14
DECODER
22
30
IRAM
40
DRAM

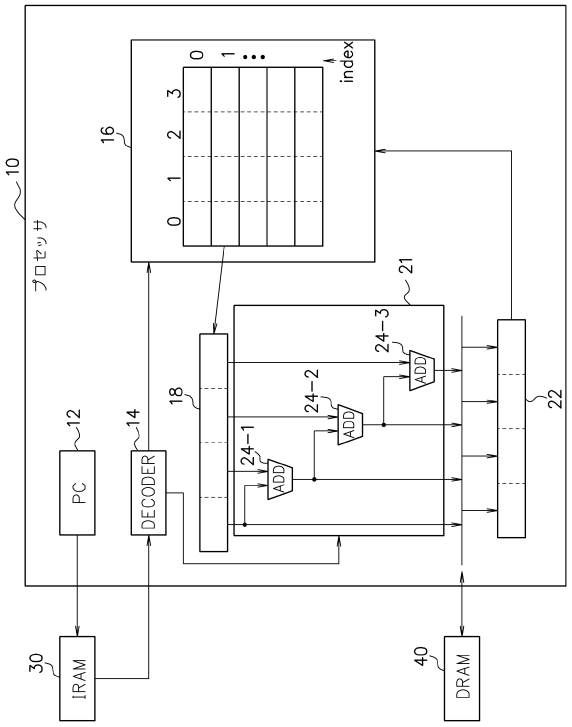
【図２】


(A)
(B)
x
t
t+1
t+2
t+3
u−2
u−1
u
u+1
u+2
201
202

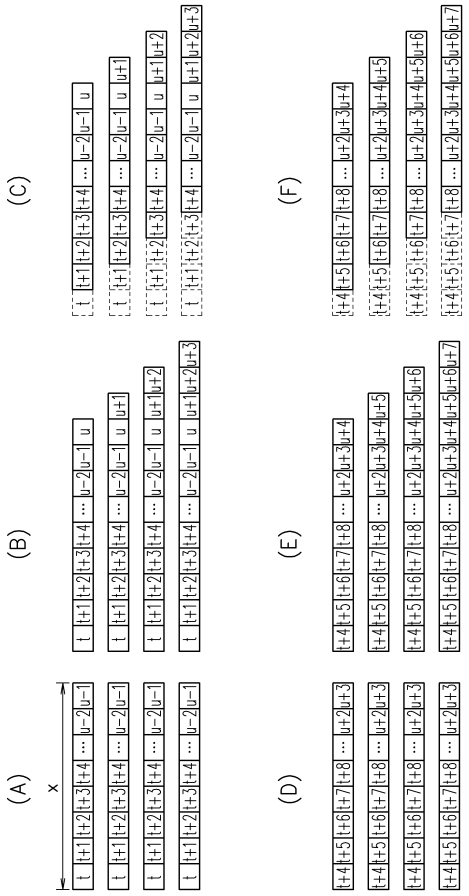
【図３】


(A)
(C)
vs
vr
vs[0]
vs[1]
vs[2]
vs[3]
vr[0]
vr[1]
vr[2]
vr[3]
301
302
303
311
312
313
314
(B)
vr[0]=vs[0]
vr[1]=vs[0]+vs[1]
vr[2]=vs[0]+vs[1]+vs[2]
vr[3]=vs[0]+vs[1]+vs[2]+vs[3]

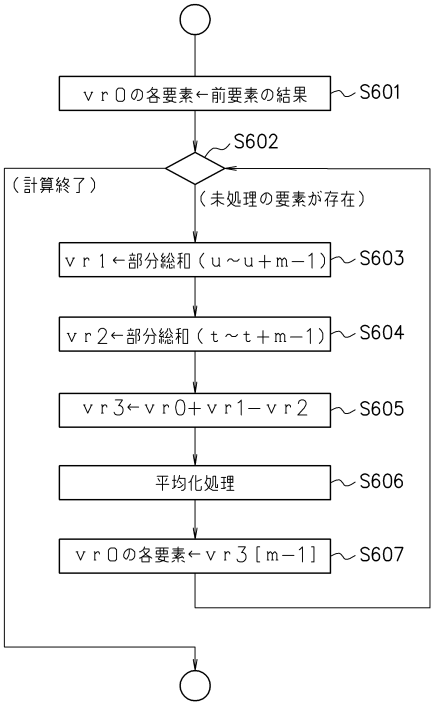
【図４】


10
プロセッサ
16
index
0
1
2
3
21
24−3
24−2
24−1
ADD
18
12
PC
14
DECODER
22
30
IRAM
40
DRAM

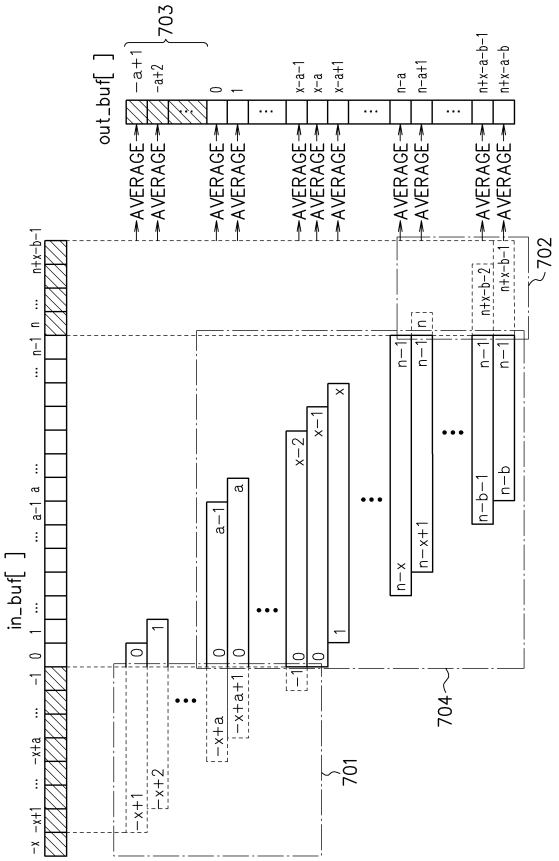
【図５】


(A)
(B)
(C)
(D)
(E)
(F)
x
t
t+1
t+2
t+3
t+4
t+5
t+6
t+7
t+8
u−2
u−1
u
u+1
u+2
u+3
u+4
u+5
u+6
u+7

【図６】



【図７】



【図８】

(A)

	マスク値設定命令
０以下	０００ ・・・ ０００
１	１００ ・・・ ０００
２	１１０ ・・・ ０００
⋮	⋮
ｋ	１１・・・１（ｋビット） ０・・・００（（ｍ－ｋ）ビット）
⋮	⋮
（ｍ－１）	１１１ ・・・ １１０
ｍ以上	１１１ ・・・ １１１

(B)

	マスク値設定命令_Ｒ
０以下	０００ ・・・ ０００
１	０００ ・・・ ００１
２	０００ ・・・ ０１１
⋮	⋮
ｋ	００・・・０（（ｍ－ｋ）ビット） １・・・１１（ｋビット）
⋮	⋮
（ｍ－１）	０１１ ・・・ １１１
ｍ以上	１１１ ・・・ １１１

【図９】

```
idx0=-x;
for(i=0;i<N;i+=m){
    m0=マスク値設定命令_R(idx0+m);  ～901
    vt=ベクタロード(&in_buf[idx0], m0);
      ・                              902
      ・
    idx0+=m;  ～903
}
```

【図１０】

```
idx1=n;
for(i=0;i<N;i+=m){
    m1=マスク値設定命令(idx1);  ～1001
    vu=ベクタロード(&in_buf[i], m1);  ～1002
      ・
      ・
    idx1-=m;  ～1003
}
```

【図１１】

```
idx2=-a+1;
for(i=0;i<N;i+=m){
  m2=マスク値設定命令_R(idx2+m);  ～1101
    ・
    ・
  データストア(&out_buf[idx2], m2, 移動平均結果);
  idx2+=m;  ～1103                    1102
}
```

【図１２】

i	除数	係数
０～ｘ－１	ｉ＋１	１／（ｉ＋１）
ｘ～ｎ－１	ｘ	１／ｘ
ｎ～	ｘ＋ｎ－ｉ－１	１／（ｘ＋ｎ－ｉ－１）

【図１３】

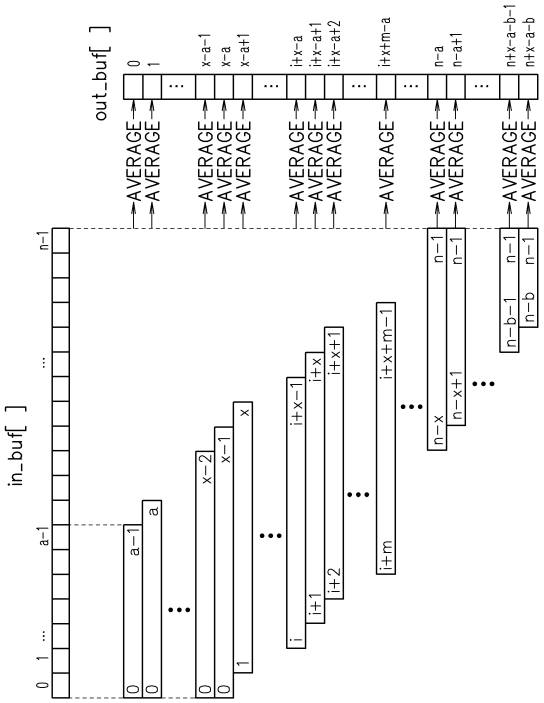
```
for(i=0;i<N;i+=m){
  vd=ベクタロード(&div_tbl[i]);  ～1301
  vave=vsum*vd;  ～1302
    ・
    ・
}
```

【図１４】

```
idx0=-x;
idx1=n;
idx2=-a+1;

vsum={0, 0, …, 0};
for(i=0;i<n+x-a-b;i+=m){
   m0=マスク値設定命令_R(idx0+m);  ～901
   m1=マスク値設定命令(idx1);  ～1001
   vt1=ベクタロード(&in_buf[idx0], m0);  ～902
   vu1=ベクタロード(&in_buf[i], m1);  ～1002
   vd=ベクタロード(&div_tbl[i]);  ～1301
   vt2=部分総和命令(vt1);  ～1401
   vu2=部分総和命令(vu1);  ～1402
   vsum=vsum+vu2-vt2;  ～1403
   vave=vsum*vd  ～1302
   m2=マスク値設定命令_R(idx2+m);  ～1101
   データストア(&out_buf[idx2], m2, vave);
   idx0+=m;  ～903                          ～1102
   idx1-=m;  ～1003
   idx2+=m;  ～1103
   vsum={vsum[m-1], vsum[m-1], …, vsum[m-1]};
}                                           ～1404
```

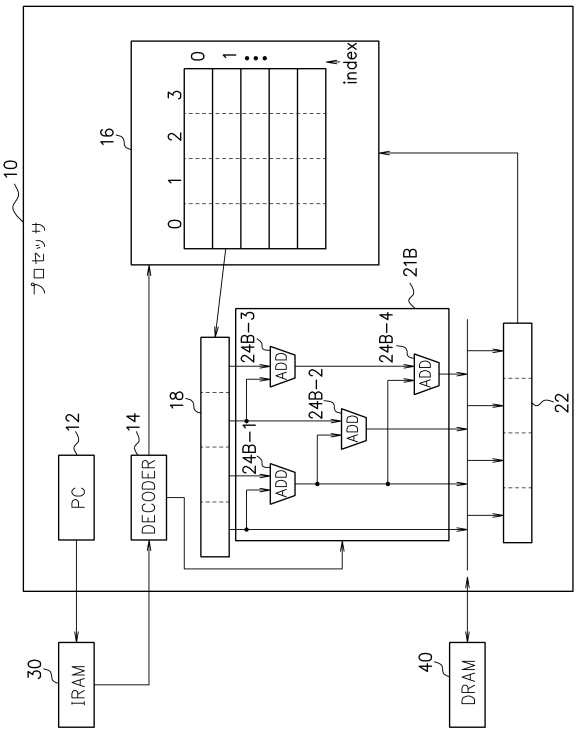
【図１５】



【図１６】

```
sum=0;                                          1601
//事前処理
for(i=0;i<a-1;i++) sum+=in_buf[i];
//前部分処理
for(j=0, cnt=a;cnt<x;i++, j++, cnt++){
   sum+=in_buf[i];
   avr=sum/cnt;
   out_buf[j]=avr;
}                                               1602
//中部分処理
for(;i<n;i++, j++){
   sum+=in_buf[i];
   sum-=in_buf[i-x];                            ～1603
   avr=sum/x;
   out_buf[j]=avr;
}
//後部分処理
for(;i<n+b;i++, j++, cnt--){
   sum-=in_buf[i-x];                            ～1604
   avr=sum/cnt;
   out_buf[j]=avr;
}
```

【図１７】



フロントページの続き

(72)発明者　野元　一宏
福岡県福岡市早良区百道浜２丁目２番１号　富士通九州ネットワークテクノロジーズ株式会社内

審査官　田中　幸雄

(56)参考文献　国際公開第２０１３／０９５６３４（WO，Ａ１）
米国特許出願公開第２００８／０１７２４３７（ＵＳ，Ａ１）
特開平１０－１４３４９５（ＪＰ，Ａ）
特開昭６３－１８７３６６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１７／１０
Ｇ０６Ｆ　１５／８０