論　文

分散演算によるマルチプライヤレスLMS適応フィルタの高性能VLSIアーキテクチャ

恒川　佳隆[†]　　高橋　強[††]　　豊田　真嗣[†]　　三浦　守[†]

High-Performance VLSI Architecture of Multiplierless LMS Adaptive Filters Using Distributed Arithmetic

Yoshitaka TSUNEKAWA[†], Kyo TAKAHASHI[††], Shinji TOYODA[†], and Mamoru MIURA[†]

あらまし　現在，定係数のフィルタ実現においてマルチプライヤレスな構成法が盛んに研究されているが，係数が時変となる適応フィルタの実現においてはそのような構成法はほとんど検討されてこなかった．その数少ないマルチプライヤレスな構成法として分散演算を用いた構成法が提案されてきたが，この従来法では入力信号に特殊な符号化を用いているため，収束特性が非常に大きく劣化するという問題点があった．これに対して我々は新たに2の補数形式による一般化を行うことによって大幅に収束特性を改善する．更に，従来法では検討されてこなかったVLSIアーキテクチャを提案し，この構成法に対してVLSI評価を行う．その結果，本提案法においては乗算器を用いた構成法に対して高速性と極めて小さな滞在時間を維持した上で，消費電力，ハードウェア量を大幅に削減できることを明らかにする．

キーワード　LMS適応フィルタ，分散演算，マルチプライヤレス，2の補数表現，VLSIアーキテクチャ

1. まえがき

現在，適応フィルタはエコーキャンセラ，ノイズキャンセラ，自動等化器などに用いられるようにその応用範囲は広く，様々な分野においてその実現の必要性が高まっている．適応フィルタの実現に関しては，高速性，低消費電力，良好な収束特性，小さな滞在時間（Latency）等のように，定係数のフィルタ実現よりも多くの要因が要求される．このため，定係数の場合よりもその実現が難しく，これらの相反する要求をすべて満足させる適応フィルタを実現することは非常に困難である．また，テレビ会議などで必要な音響エコーキャンセラにおいては，室内音場のインパルス応答を高速に推定する能力とインパルス応答変動に対する追従性が要求される[1]．そのため，非常に高次の適応フィルタが必要とされている．しかし，これまでに高次の適応フィルタのハードウェア実現に対するアプローチはあまり行われてこなかった．

適応フィルタをハードウェア実現する方法として，パイプライン処理を用いた構成法がこれまでに提案されてきた[2]～[5]．パイプライン処理を実現するために用いられるアルゴリズムとして，再帰最小2乗(RLS)アルゴリズム，Delayed LMS (DLMS) アルゴリズムが挙げられる．RLSに基づく構成法では，多くの乗算や，除算，平方根のような複雑な演算を必要とするため，非常に膨大なハードウェア量が必要となる．またDLMSに基づく構成法では，各タップごとにレジスタを挿入してパイプライン化し，クリティカルパスを小さな値にすることによって，スループットの向上を図っている．そのため，タップ数分の乗算器が必要となる．また次数の増加に伴い，滞在時間が増大する．滞在時間の増大は，非定常環境でのアルゴリズムの追従性の劣化などを引き起こす[6]．どちらの構成法においても，多くの乗算器が必要となる．乗算器の消費電力，ハードウェア量は加算器などと比較しても非常に大きいため，高次での実現を考慮した場合，膨大な消費電力とハードウェア量が必要となる．

† 岩手大学工学部情報工学科，盛岡市
Faculty of Engineering, Iwate University, Morioka-shi, 020–8551 Japan
†† 岩手県立産業技術短期大学校，岩手県
Iwate Industrial Technology Junior College, Yahaba-cho, Iwate-ken, 028–3615 Japan

これに対して低消費電力，低ハードウェア量を実現するために乗算器を使用しない，つまりマルチプライヤレスな構成法として，分散演算（Distributed Arithmetic）を用いた適応フィルタがこれまでに提案されてきた[7],[8]．しかし我々の検討結果，この従来法には実現において非常に大きな問題点があることがわかった[9]．従来法では，入力信号の符号化に通常の2進数表現における“0”に“−1”を，“1”に“1”を割り当てるという特殊な符号化を用いており，これをもとに式展開を行い，フィルタ出力を求める式及び更新式を導出していた．これによって，入力信号の符号化と内部演算の符号化の違いから推定精度が大幅に劣化していた．更に，実際の適応動作においては入力信号の各ビットは“1”と“−1”ではなく，“1”と“0”として扱われるため，すべての入力信号にオフセットがかかった状態になり，入力信号の自己相関行列の固有値が本来よりも大きな値に設定される．したがって，ステップサイズが小さい値に設定されるため，収束速度が大幅に劣化していた．また，VLSIアーキテクチャについて考慮されておらず，その高速性についても検討が行われてこなかった．

本論文では，これらの問題点を解決するために従来の分散演算を用いた適応フィルタに対して，入力信号の符号化に対しても2の補数形式を適用した高性能なマルチプライヤレスVLSIアーキテクチャを提案する．まず，2の補数形式を用いて式展開を行い，すべての式を2の補数形式によって一般化する．これによって推定精度及び収束速度を大幅に改善できることを，計算機シミュレーションによって示す．更に，従来法で検討されなかったVLSIアーキテクチャ及びその高速性を考慮した構成法について提案する[10]．この構成法に対してVLSI設計システムPARTHENONによるVLSI評価を行い，乗算器を用いた構成法との比較を行う．その結果，本提案法が高次においても高速性と極めて小さい滞在時間を維持した上で，低消費電力，低ハードウェア量を実現できることを明らかにする．

2. 分散演算型適応フィルタ

従来，分散演算は定係数の内積演算を効率的に行うための計算手法として用いられてきたが，係数が時変となる適応信号処理においても有効な演算手法となる．

本章では，まず2の補数形式に基づいた定係数の分散演算の原理について述べ，次にLMSアルゴリズムの更新式から2の補数形式による分散演算型適応フィルタ（以下，DA適応フィルタと呼ぶ）の更新式を導出する．更に，従来法において入力信号に特殊な符号化を用いたことによって生じる問題点について述べる．

2.1 提案法の分散演算

分散演算は，内積演算をテーブルルックアップによって実現する計算手法である[11]．

ここで，項数Nの定係数ベクトル$\boldsymbol{a}=(a_1,\ldots,a_N)$と変数ベクトル$\boldsymbol{v}=(v_1,\ldots,v_N)$との内積

$$y=\boldsymbol{a}\boldsymbol{v}=\sum_{i=1}^{N}a_iv_i \tag{1}$$

を考える．ただし，$-1\leq v_i<1$で，v_iはBビットの固定小数点形の2の補数表示である．つまり，

$$v_i=-v_i^0+\sum_{k=1}^{B-1}v_i^k2^{-k} \tag{2}$$

と表される．ここで，v_i^kはv_iのkビット目の値で0または1である．式(2)を式(1)に代入すれば，内積演算$\boldsymbol{av}$は次式で示される．

$$\boldsymbol{y}=-\Phi(v_1^0,\ldots,v_N^0)+\sum_{k=1}^{B-1}\Phi(v_1^k,\ldots,v_N^k)2^{-k} \tag{3}$$

ただし，部分積を表す関数Φは

$$\Phi(v_1^k,\ldots,v_N^k)=\sum_{i=1}^{N}a_iv_i^k \tag{4}$$

である．

式(3)より，分散演算を用いて内積を求めるためには，あらかじめ関数Φのテーブルを用意しておく必要がある．関数Φのテーブルには$(v_1^k,\ldots,v_N^k)$をアドレスとして，その各ビットパターンに対応した演算の結果を格納しておく．計算時は，アドレスのビットパターンによって関数Φのテーブルから演算結果を読み出し，右シフト加算を行う．この操作を語長B回行うことによって内積を求めることができる．図1に分散演算のアーキテクチャを示す．関数Φのテーブルは$(v_1^k,\ldots,v_N^k)$をアドレスとするROMで実現でき，右シフト加算は加算器とレジスタによって実現できる．

以上より，原理的には処理時間が項数Nに依存せず，語長Bのみに依存する構成が実現可能となる．また滞在時間も項数に依存せず，一定の値を保つことができる．ここでは滞在時間をあるサンプルがシステム

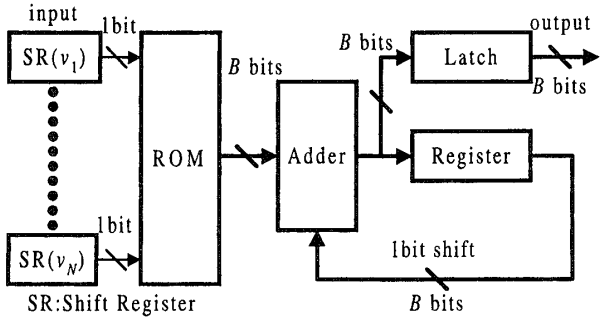


図1 定係数における分散演算の基本構成
Fig. 1 Basic structure of distributed arithmetic for constant coefficients.

に入力されてから，そのサンプルに対応した演算結果が出力されるまでの時間と定義する．更に，乗算器を使用せずに内積演算の結果を求めることができるため，大幅に消費電力及びハードウェア量を削減することが可能となる．このようにハードウェアの効率性を考慮した場合，分散演算は非常に有効な手法となる．

2.2 DA 適応フィルタの更新式の導出

LMS に基づく更新式から，2 の補数形式を用いて式展開を行うことによって DA 適応フィルタの更新式を導出する．

タップ数 N の入力信号ベクトルを

$$\boldsymbol{S}(k) = [s(k), s(k-1), \ldots, s(k-N+1)]^T \quad (5)$$

タップ数 N の係数ベクトルを

$$\boldsymbol{W}(k) = [w_0(k), w_1(k), \ldots, w_{N-1}(k)]^T \quad (6)$$

とする．入力信号ベクトル $\boldsymbol{S}(k)$ を

$$\boldsymbol{S}(k) = \boldsymbol{A}(k)\,\boldsymbol{F} \quad (7)$$

と定義すると，フィルタ出力を求める式は次式で表される．

$$y(k) = \boldsymbol{S}^T(k)\,\boldsymbol{W}(k) = \boldsymbol{F}^T\,\boldsymbol{A}^T(k)\,\boldsymbol{W}(k) \quad (8)$$

式 (7)，(8) において，アドレスマトリクス $\boldsymbol{A}(k)$ は

$$\boldsymbol{A}(k) = \begin{bmatrix} b_0(k) & b_0(k-1) & \cdots & b_0(k-N+1) \\ b_1(k) & b_1(k-1) & \cdots & b_1(k-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ b_{B-1}(k) & b_{B-1}(k-1) & \cdots & b_{B-1}(k-N+1) \end{bmatrix}^T \quad (9)$$

スケーリングベクトル $\boldsymbol{F}$ は

$$\boldsymbol{F} = [-2^0, 2^{-1}, \ldots, 2^{-(B-1)}]^T \quad (10)$$

で表される．ここで，$b_i(k)$ は時刻 k における入力信号 $s(k)$ の i ビット目の値である．式 (8) において

$$\boldsymbol{P}(k) = \boldsymbol{A}^T(k)\,\boldsymbol{W}(k) \quad (11)$$

と定義することによって，フィルタ出力を求める式は

$$y(k) = \boldsymbol{F}^T\,\boldsymbol{P}(k) \quad (12)$$

となる．ここで，関数 $\boldsymbol{P}(k)$ は

$$\boldsymbol{P}(k) = [p_0(k), p_1(k), \ldots, p_{B-1}(k)]^T \quad (13)$$

と定義する．

次に，LMS の更新式から 2 の補数形式による DA 適応フィルタの更新式を導出する過程を示す．LMS による更新式は次式で表される．

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\mu e(k)\,\boldsymbol{S}(k) \quad (14)$$

また，誤差信号 $e(k)$ は次式で求められる．

$$e(k) = d(k) - y(k) \quad (15)$$

ここで，$d(k)$ は未知システムの所望信号を表す．式 (14) の両辺に左から $\boldsymbol{A}^T(k)$ を掛けることにより，次式が得られる．

$$\boldsymbol{A}^T(k)\,\boldsymbol{W}(k+1) = \boldsymbol{A}^T(k)\{\boldsymbol{W}(k) + 2\mu e(k)\,\boldsymbol{A}(k)\,\boldsymbol{F}\} \quad (16)$$

更に，式 (16) において

$$\boldsymbol{P}(k+1) = \boldsymbol{A}^T(k)\,\boldsymbol{W}(k+1) \quad (17)$$

と定義することによって，次の更新式が得られる．

$$\boldsymbol{P}(k+1) = \boldsymbol{P}(k) + 2\mu e(k)\,\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F} \quad (18)$$

ここで，LMS 適応フィルタと DA 適応フィルタの更新動作の違いについて述べる．LMS 適応フィルタの更新動作では式 (14) に示すように係数 $\boldsymbol{W}(k)$ を直接更新する．これに対して DA 適応フィルタの更新動作は，フィルタ出力の計算時にアドレスマトリクス $\boldsymbol{A}^T(k)$ の行ベクトル $(b_i(k), \ldots, b_i(k-N+1))$ のビットパターンによって関数テーブル（以下，適応関数空間と呼ぶ）から読み出した関数 $\boldsymbol{P}(k)$ に対して，式 (18) における右辺の第 2 項の更新値を加える．その加

算結果を同じアドレスマトリクス $\boldsymbol{A}^T(k)$ の行ベクトルによって指定される適応関数空間のアドレスに格納し，関数 $\boldsymbol{P}(k)$ を更新する．

式 (18) に示した更新式では，毎回更新値を求めるたびに $\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$ という行列の乗算を行う必要がある．そのため，その計算に時間がかかり，リアルタイム処理が困難になるという問題点がある．そこで我々は，スケーリングベクトル $\boldsymbol{F}$ を含めた $\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F}$ を入力信号の統計的性質により，$0.25N\,\boldsymbol{F}$ に置き換えて考えることが可能となることを明らかにした．この導出については付録に示す．この結果を式 (18) に適用すると，更新式は

$$\boldsymbol{P}(k+1) = \boldsymbol{P}(k) + 0.5\mu N e(k)\,\boldsymbol{F} \tag{19}$$

となる．この更新式を用いた場合にも，多くの計算機シミュレーションにより収束することが確認されている．また，提案法と同様に $\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$ を対角化した従来の分散演算を用いた適応フィルタは，実際にディジタル電話回線の適応キャンセラ等に用いられている [7]．

ここで N 及び μ を 2 のべき乗で考えた場合，式 (19) によって更新値を誤差 $e(k)$ に対するシフト操作のみで求めることが可能となる．これによって，高速なリアルタイム処理が実現できる．この DA 適応フィルタのブロック図を図 2 に示す．フィルタ出力を求める部分は分散演算の基本構成をほぼそのまま用いることができる．また，DA 適応フィルタでは適応関数空間を RAM によって実現している．

2.3 従来法の問題点

本節では，従来法と提案法の DA 適応フィルタの式展開の違いを示し，従来法の問題点について述べる．

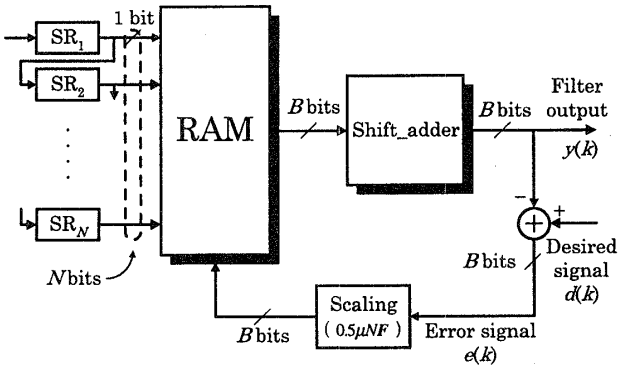


図 2　DA 適応フィルタのブロック図
Fig. 2　Block diagram of DA adaptive filter.

従来法では入力信号 $s(k)$ の符号化に以下に示すような特殊な符号化を用いていた．各入力信号 $s(k)$ をオフセットバイナリー形式で符号化し，その各ビットにおいて論理値 “0” が実値 “−1” を，論理値 “1” が実値 “1” をとると考える．そのとき，入力信号ベクトルは以下の式で表される．

$$\boldsymbol{S}(k) = \boldsymbol{A}'(k)\,\boldsymbol{F}' \tag{20}$$

上式において，アドレスマトリクス $\boldsymbol{A}'(k)$ は

$$\boldsymbol{A}'(k) = \begin{bmatrix} b'_1(k) & b'_1(k-1) & \cdots & b'_1(k-N+1) \\ b'_2(k) & b'_2(k-1) & \cdots & b'_2(k-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ b'_B(k) & b'_B(k-1) & \cdots & b'_B(k-N+1) \end{bmatrix}^T \tag{21}$$

スケーリングベクトル $\boldsymbol{F}'$ は

$$\boldsymbol{F}' = [2^{-1}, 2^{-2}, \ldots, 2^{-B}]^T \tag{22}$$

である．

次に，LMS の更新式から DA 適応フィルタの更新式を導出する過程を示す．従来法においても，提案法と同様に LMS の更新式である式 (14) の両辺に左から $\boldsymbol{A}'^T(k)$ を掛けることにより，次式が得られる．

$$\boldsymbol{P}'(k+1) = \boldsymbol{P}'(k) + 2\mu e'(k)\,\boldsymbol{A}'^T(k)\,\boldsymbol{A}'(k)\,\boldsymbol{F}' \tag{23}$$

入力信号が平均 0 の白色信号であり，入力信号を構成する各ビットが互いに無相関であるという条件のもとで $\boldsymbol{A}'^T(k)\,\boldsymbol{A}'(k)$ の期待値を求めると，

$$E\left[\boldsymbol{A}'^T(k)\,\boldsymbol{A}'(k)\right] = \begin{bmatrix} N & 0 & \cdots & 0 \\ 0 & N & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N \end{bmatrix} \tag{24}$$

となる．ここで $E[\cdot]$ は期待値操作を示す．この $\boldsymbol{A}'^T(k)\,\boldsymbol{A}'(k)$ の統計的性質を式 (23) に適用することにより，更新式は

$$\boldsymbol{P}'(k+1) = \boldsymbol{P}'(k) + 2\mu N e'(k)\,\boldsymbol{F}' \tag{25}$$

となる．

従来法において入力信号にこのような特殊な符号化を用いた理由は，式 (24) で示した $E[\boldsymbol{A'}^T(k)\,\boldsymbol{A'}(k)]$ の対角化を容易に行い，式展開を簡単化するためと考えられる．これに対して，2 の補数形式を用いた場合には従来法のように $E[\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)]$ を対角化することは不可能となる．しかし，我々は **2.2** に示したようにスケーリングベクトル $\boldsymbol{F}$を含めて考えることによって，2 の補数形式においても $E[\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)]$ の対角化が可能となることを明らかにした．これによって，初めて 2 の補数形式による式展開が可能となった．

次に，我々の検討によって明らかとなった従来法の大きな問題点について述べる．従来法では，先で述べたように入力信号に特殊な符号化を用いていた．これによって，入力信号の符号化と内部演算の符号化の違いから推定精度が大幅に劣化していた．更に，実際の適応動作においては入力信号の各ビットは “1” と “−1” ではなく “1” と “0” として扱われる．そのため，すべての入力信号にオフセットがかかった状態になり，符号化された入力信号の自己相関行列の固有値が本来よりも大きな値に設定される．これによって，ステップサイズが小さい値に設定されるため，収束特性が大幅に劣化していた [9]．

この問題点を解決するために，我々は **2.1**，**2.2** に示したように 2 の補数形式を用いてすべての式を展開し，符号化方式を統一した．これによって，入力信号の符号化と内部演算の符号化を同じ符号化で扱えるため，推定精度を大幅に改善することができる．更に，従来法のように実際の適応動作において入力信号にオフセットがかからないため，入力信号の自己相関行列の固有値を正確に求めることができ，ステップサイズを適切な値に設定することができる．したがって，収束速度を大幅に改善することが可能となる．

3. 適応関数空間の分割化

DA 適応フィルタでは，タップ数 N の増加に伴い収束速度が劣化するが，この原因について説明する．まず，DA 適応フィルタにおいて更新される適応関数空間は，**2.2** で示したようにアドレスマトリクス $\boldsymbol{A}^T(k)$ の N 次行ベクトルをアドレスとして指定される要素である．したがって，全適応関数空間（容量 2^N）が一つのアドレスベクトル当りに更新される確率 Pr_{update} は

$$Pr_{update} = \frac{1}{2^N} \tag{26}$$

となる．タップ数 N の増加は，適応関数空間の容量を増加させるために更新確率が減少し，収束状態に達するまでには多くの繰返しが必要になる．また，容量の大きな適応関数空間を RAM で実現することは，消費電力やハードウェア量の点で非常に不利となる．

この問題点を解決するために，適応関数空間を分割する手法が提案されてきた [12]．これは，容量 2^N の適応関数空間を M 個に分割することにより，個々の空間を小容量化して更新確率を向上させる手法である．この際，分割された適応関数空間のアドレスビット数 R は (N/M) となるので，分割された適応関数空間の容量は 2^R になる．したがって，分割された適応関数空間が一つのアドレスベクトル当りに更新される確率 Pr'_{update} は

$$Pr'_{update} = \frac{1}{2^R} \tag{27}$$

となり，分割しない場合の 2^{N-R} 倍に向上する．このように，分割化による更新確率の向上に伴って収束速度が改善され，消費電力やハードウェア量も減少させることが可能になる．ところが，この分割化の手法を従来法に適用しても，消費電力やハードウェア量は減少するが，収束速度の改善はあまり見られず大きく劣化したままである．

そこで我々は DA 適応フィルタと同様に，この適応関数空間を分割化した場合の構成法に対しても 2 の補数形式を用いた．この場合にも 2 の補数形式を適用することができるのは，DA 適応フィルタの場合と同様にアドレスマトリクスの乗算の対角化を可能としたからである．これによって，従来法に対して収束特性を大幅に改善し，更に適応関数空間の分割数が多いほど収束速度を速くすることを可能とする．この適応関数空間を分割した構成をマルチメモリブロック構造（Multi-Memory Block Structure）と呼び，この構造を適用した DA 適応フィルタを MDA 適応フィルタと呼ぶことにする．

MDA 適応フィルタにおける 2 の補数形式を用いた式展開を以下に示す．N タップの DA 適応フィルタの適応関数空間を M 個に分割した場合の MDA 適応フィルタについて考える．各フィルタ係数ベクトルを

$$\boldsymbol{W}_m(k) = [w_{m0}(k), w_{m1}(k), \ldots, w_{m(R-1)}(k)]^T \tag{28}$$

各 MDA 適応関数ベクトルを

$$\boldsymbol{P}_m(k) = [p_{m0}(k), p_{m1}(k), \ldots, p_{m(B-1)}(k)]^T \quad (m = 0, 1, \ldots, M-1) \tag{29}$$

と定義する．ここで，R は各適応関数空間のアドレスのビット数を表す．各 MDA 適応関数ベクトルを式 (30) のように表すことによって，フィルタ出力は式 (31) で求めることができる．

$$\boldsymbol{P}_m(k) = \boldsymbol{A}_m^T(k)\,\boldsymbol{W}_m(k) \tag{30}$$

$$y(k) = \sum_{m=0}^{M-1} \boldsymbol{F}^T\,\boldsymbol{P}_m(k) \tag{31}$$

式 (30) において，アドレスマトリクス $\boldsymbol{A}_m(k)$ は

$$\boldsymbol{A}_m(k) = \begin{bmatrix} b_{m0}(k) & b_{m0}(k-1) & \cdots & b_{m0}(k-R+1) \\ b_{m1}(k) & b_{m1}(k-1) & \cdots & b_{m1}(k-R+1) \\ \vdots & \vdots & \ddots & \vdots \\ b_{m(B-1)}(k) & b_{m(B-1)}(k-1) & \cdots & b_{m(B-1)}(k-R+1) \end{bmatrix}^T \tag{32}$$

と定義する．

次に，MDA 適応フィルタの更新式を以下に示す．

$$\boldsymbol{P}_m(k+1) = \boldsymbol{P}_m(k) + 0.5\mu Re(k)\,\boldsymbol{F} \quad (m = 0, 1, \ldots, M-1) \tag{33}$$

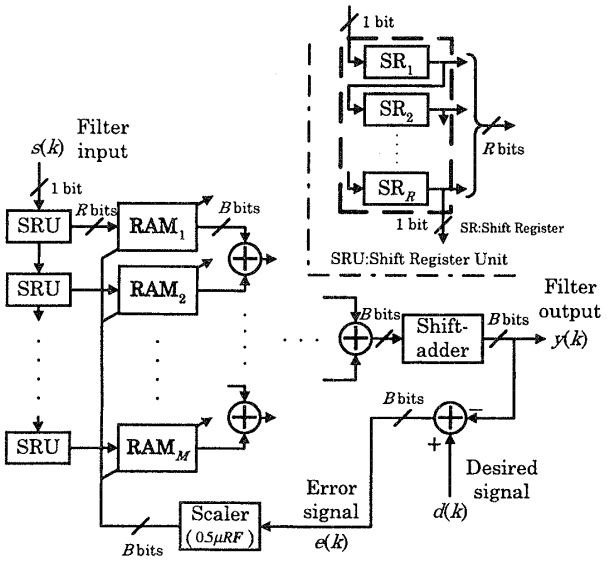


図 3　MDA 適応フィルタのブロック図
Fig. 3　Block diagram of MDA adaptive filter.

MDA 適応フィルタでは，図 3 のブロック図に示すように適応関数空間を分割して小容量化することにより更新確率を向上させ，収束速度を大幅に改善することが可能である．また，適応関数空間を実現する RAM の容量も大幅に削減されるため，低消費電力及び低ハードウェア量を実現することができる．

4. シミュレーションによる比較

本章では，収束速度における提案法の有効性を，計算機シミュレーションによって明らかにする．まず，そのシミュレーションモデルを図 4 に示す．ここでは，入力として平均 0，分散 0.01 の白色信号を用いる．また，未知システム出力 $d(k)$ には -75[dB] の白色信号を観測雑音 $v(k)$ として加えており，結果は独立した 20 回の試行の集合平均である．それぞれのステップサイズ μ は 2 のべき乗において最も良い収束特性を示すときの値とする．

図 5 はタップ数が 16 の場合に対する提案法と従来法の適応アルゴリズムの収束特性である．なお，このシミュレーションは内部演算を語長 20 ビットで行った

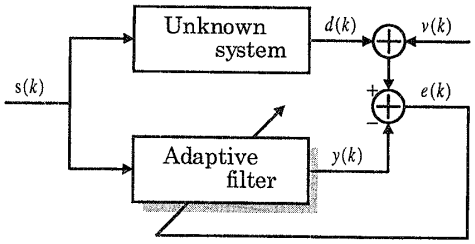


図 4　シミュレーションモデル
Fig. 4　Simulation Model.

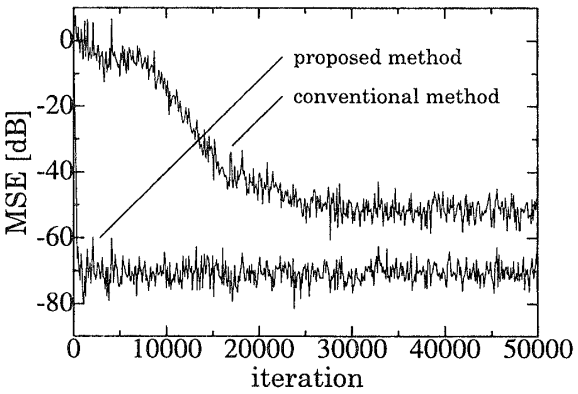


図 5　提案法と従来法の収束特性の比較
Fig. 5　Comparison of convergence characteristics between proposed method and conventional method.

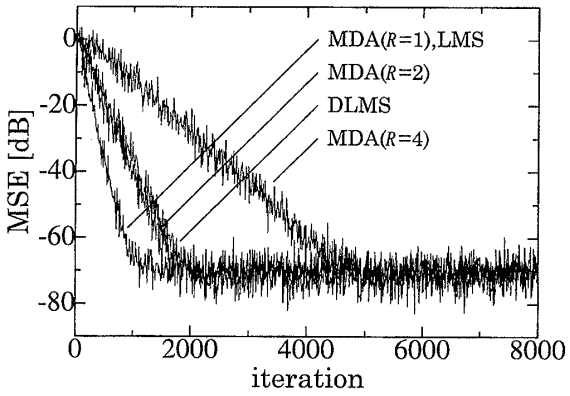


図6 分割数を変化させた場合のMDAの収束特性
Fig. 6 Convergence characteristics of MDA to change of the number of divisions.

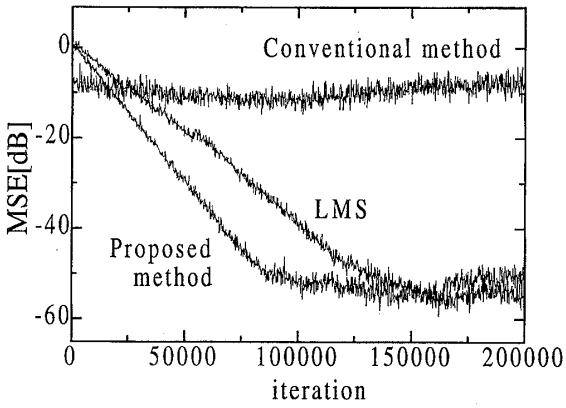


図7 有色信号に対する収束特性
Fig. 7 Convergence characteristics of LMS, proposed method and conventional method when the input signal was the colored process.

ものである．このように，それほどタップ数が大きくない場合でも，従来法の推定精度及び収束速度が大幅に劣化している．これに対して提案法では収束速度及び推定精度を大幅に改善できていることがわかる．また，このシミュレーションについては多くの計算例から解析を行っており，タップ数を大きくするほど従来法が大きく劣化することを確認している[9]．

次に，提案法のMDA適応アルゴリズムの収束特性を明らかにするために，LMSアルゴリズム，及び**6.**においてVLSI評価の比較対象として用いるDLMSアルゴリズムの収束特性との比較を行う[4]．図6にタップ数が64の場合の提案法，LMS，そしてDLMSの収束特性のシミュレーション結果を示す．この結果から，提案法の収束特性において適応関数空間のアドレスが2ビット($R=2$)の場合には，DLMSアルゴリズムの収束特性とほぼ一致している．更にアドレスが1ビット($R=1$)の場合には，LMSと同等の収束特性を得ることができる．このように提案法のMDA適応フィルタの適応アルゴリズムには，適応関数空間のアドレスのビット数が小さいほど良好な収束速度を示すという特長がある．

更に，有色信号に対する収束特性を図7に示す．ここで，有色信号は係数0.99の1次AR過程を用いて生成した．また，DA適応フィルタについては適応関数空間のアドレスが1ビット($R=1$)の場合である．この結果より，提案法はLMSよりも高速な収束特性を示していることがわかる．この原因は，LMSとDA適応フィルタの係数更新メカニズムが異なることによる．つまり，LMSでは入力信号ベクトルの各要素が対応する係数の更新に直接寄与しているため，有色性の影響をそのまま受ける．これに対してDA適応フィルタでは，アドレスマトリクス $\boldsymbol{A}^T(k)$ の行ベクトル $(b_i(k),\ldots,b_i(k-N+1))$ が対応する適応関数空間の更新に寄与する．したがって，この場合のビット間の相関は連続する入力信号の相関に直接は関係しないため，DA適応フィルタの入力信号は有色性が軽減され，有色信号に対しても収束速度の劣化は少ないものと考えられる．しかし，従来法のDA適応フィルタでは入力信号のオフセットの影響により，収束速度は大きく劣化したままである．

ただし，これらの理論な考察については，今後更に検討が必要である．

5. MDA適応フィルタの構成法

これまで従来法ではVLSIアーキテクチャ及びその高速性に関して検討が行われてこなかった．本章では，高速性を考慮したMDA適応フィルタのハードウェア構成を提案する．

MDA適応フィルタの構成のアルゴリズムは，フィルタ出力の計算とRAMの更新の異なる二つのステージからなる．処理速度の向上を図る構成として，図8のような構成を提案する．同図は分割数が2の場合の構成である．以下，この構成をMDA適応フィルタの高速形と呼ぶことにする．

まず，フィルタ出力を求めるステージについて述べる．ここでは，以下に示す動作を語長回繰り返すことによってフィルタ出力を求める．

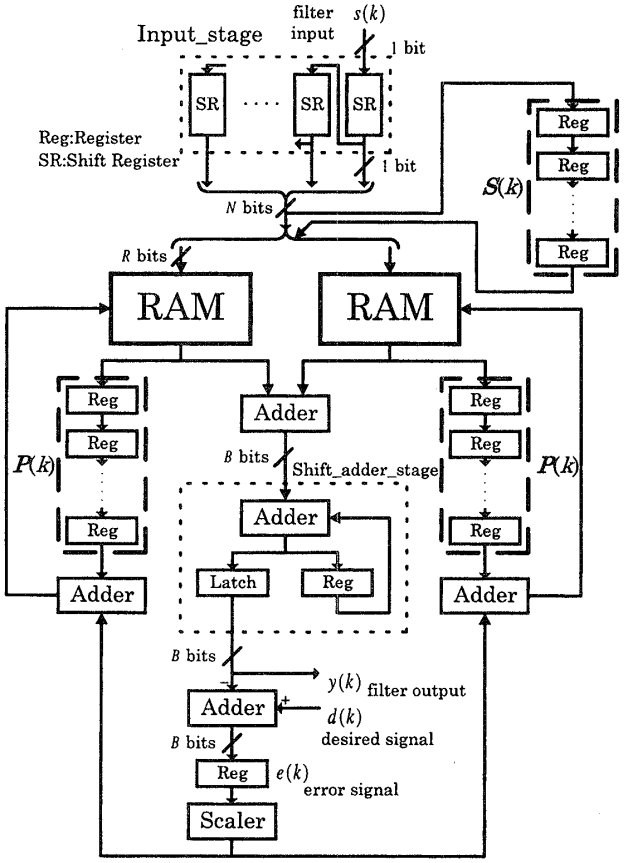


図8　MDA適応フィルタの高速形構成
Fig. 8　High-speed structure of MDA adaptive filter.

[step1]　入力部に入力信号 $s(k)$ を1ビット入力する．

[step2]　入力部から出力されるビットパターンによってRAMのアドレスを指定し，関数 $\boldsymbol{P}(k)$ の値を読み出す．

[step3]　読み出された関数 $\boldsymbol{P}(k)$ の値をシフト加算部で累積加算する．

更にこのステージでは，得られたフィルタ出力 $y(k)$ と所望信号 $d(k)$ との差をとり，誤差 $e(k)$ を求めている．また，次のステージでRAMの更新を行うために必要なデータであるRAMのアドレス指定に用いた入力 $\boldsymbol{S}(k)$ と，そのアドレスから読み出された関数 $\boldsymbol{P}(k)$ をレジスタで保持している．

次に，RAMの更新を行うステージについて説明する．前のステージでレジスタを用いて保持していた入力 $\boldsymbol{S}(k)$ によって更新するべきRAMのアドレスを指定し，保持しておいた関数 $\boldsymbol{P}(k)$ とシフトされた誤差 $e(k)$ との加算結果をそのアドレスに書き込む．この動作を語長回行うことによってRAMの更新を行う．

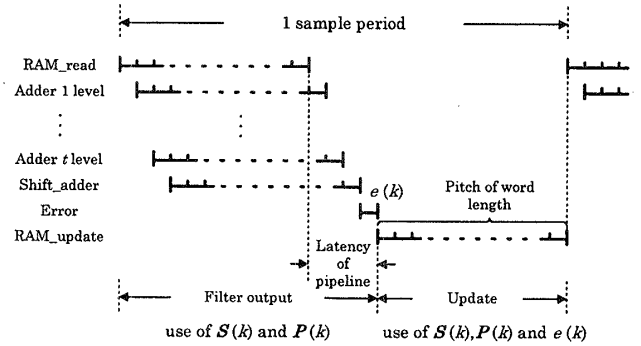


図9　MDA適応フィルタの高速形のタイムチャート
Fig. 9　Timechart for high-speed structure of MDA adaptive filter.

この更新動作において従来法の構成では，入力 $\boldsymbol{S}(k)$ のみを保持していた．したがってRAMの更新動作は，RAMから関数 $\boldsymbol{P}(k)$ を読み出し，更新値と加え合わせ，その加算結果をRAMに書き込むという動作になっていた．そのため，パイプラインのピッチが大きくなり，処理速度が大きく低下するという問題があった．それに対し我々の構成法では関数 $\boldsymbol{P}(k)$ の値も保持する構成にした．これによってRAMの更新動作は，保持しておいた関数 $\boldsymbol{P}(k)$ の値と更新値の加算結果をRAMに書き込むという動作だけになる．したがって，パイプラインのピッチをほぼ加算器の演算時間にまで減少することができるため，従来法よりも高速な実現が可能となる．この構成のタイムチャートを図9に示す．同図より，次数を増加させた場合でもパイプラインの加算段数が数段増加するだけである．パイプラインの加算段数は語長に対して占める割合が小さいため，この構成法では処理速度及び滞在時間をほぼ一定の値に保つことが可能となる．

6. VLSI評価

本章では，マルチプライヤレスな構成法であるMDA適応フィルタの高速形の構成法に対してVLSI設計システムPARTHENONによってVLSI評価を行う[13]．MDA適応フィルタの高速形の構成において，タップ数が32，64，128の場合の構成に対するVLSI評価の結果を表 1 に示す．ここで，実部品として用いたセルライブラリーの設計ルールは0.8 μmCMOSスタンダードセル（VLSIテクノロジ社）であり，電源電圧は5.0[V]である．ここでの設計仕様は，RAMの分割数をタップ数32の場合は8，64の場合は16，128の場合は32とする．また，演算に用いるデータ形式は2

表1 MDA適応フィルタの高速形に対するVLSI評価
Table 1 VLSI evaluation for high-speed structure of MDA adaptive filter.

Number of taps	32	64	128
Machine cycle[ns]	27	27	27
Sampling rate[MHz]	1.122	1.089	1.058
Latency[ns]	486	513	540
Power dissipation[W]	2.176	4.184	8.205
Area[mm^2]	12.857	24.919	49.118
Number of gates	47,331	90,834	178,185

表2 DLMSパイプライン適応フィルタに対するVLSI評価
Table 2 VLSI evaluation for the DLMS-pipelined adaptive filter.

Number of taps	32	64	128
Machine cycle[ns]	63	63	63
Sampling rate[MHz]	15.873	15.873	15.873
Latency[ns]	2016	4032	8064
Power dissipation[W]	6.446	12.892	25.785
Area[mm^2]	51.203	102.406	204.812
Number of gates	249,440	496,960	997,760

の補数表現による語長16ビットの固定小数点とする．

ここでは，提案した構成法の比較対象に，乗算器を使用した構成法としてDLMSアルゴリズムに基づくパイプライン適応フィルタを用いた[4]．この構成法に対するVLSI評価の結果を表2に示す．このDLMSパイプライン適応フィルタは，各タップごとにパイプライン処理を実行する構成になっており，その各タップは乗算器，加算器，遅延器で構成されている．ここで使用した乗算器は，Boothのアルゴリズムに Wallace tree方式とCLA加算器を用いたものである．

この結果から，我々が提案したMDA適応フィルタの高速形の構成法がタップ数128という高次においても，1.06 MHzという高い処理速度と540 nsという極めて小さい滞在時間を実現できることがわかる．更に，DLMSパイプライン適応フィルタの構成に対して消費電力を68.2%，面積を76.0%，ゲート数を82.1%と大幅に削減することが可能となる．また，滞在時間に関しても約93.3%と大幅に減少でき，極めて小さい値に抑えることができる．更により高次の場合に対しても，この構成法は処理速度と滞在時間をほぼ一定の値に保つことが可能である．

また，極めて滞在時間が小さい構成法として，Haradaらの LMSパイプライン適応フィルタの構成法についても比較を行う[14]．この構成法に対するVLSI評価の結果を表3に示す．なお，ここでは滞在時間が最小となる構成法である Arc1に対して比較を

表3 LMSパイプライン適応フィルタに対するVLSI評価
Table 3 VLSI evaluation for the LMS-pipelined adaptive filter.

Number of taps	32	64	128
Machine cycle[ns]	131	131	131
Sampling rate[MHz]	7.634	7.634	7.634
Latency[ns]	131	131	131
Power dissipation[W]	6.833	13.666	27.333
Area[mm^2]	107.350	214.700	429.399
Number of gates	520,576	1,041,152	2,082,304

行った．この構成法では，ルックアヘッド変換を適用することによりLMSアルゴリズムのパイプライン処理化を可能とした．これによって，タップ数に依存しない高速な処理速度，LMSと同等の収束特性を維持した上で，131 ns以下という我々の構成法よりも小さな滞在時間で実現が可能となる．しかし，その実現にはDLMSパイプライン適応フィルタの構成法において必要となるハードウェア量の2倍以上を必要とするため，高次における実現が問題となる．これに対して，本構成法では約1/10程度のハードウェア量で実現が可能となる．

7. むすび

本論文では，従来の分散演算を用いた適応フィルタに対して，入力信号の符号化に対しても2の補数形式を適用した高性能なマルチプライヤレスVLSIアーキテクチャを提案した．まず，従来の分散演算を用いた適応フィルタの収束特性が，入力信号の特殊な符号化によって大幅に劣化していることを明らかにした．それに対して，本提案法ではすべてのアルゴリズムを2の補数形式で一般化することによって，収束速度及び推定精度を大幅に改善した．更に，従来法では検討されてこなかったVLSIアーキテクチャ及びその高速性を考慮した構成法についても提案した．最後に，本提案の構成に対してVLSI設計システムPARTHENONを用いてVLSI評価を行った．その結果，本提案法がタップ数128という高次においても，処理速度1.06 MHz（0.8 μmCMOSスタンダードセル）と滞在時間540nsを実現できることを明らかにした．また，ここで比較対象として用いたDLMSパイプライン適応フィルタの構成法に対して，消費電力を68%，面積を76%，ゲート数を82%削減することを可能とした．更に，より高次の構成に対しても処理速度及び滞在時間をほぼ一定の値に保つことができることを示した．

以上のことから，本提案法においては高次に対して

も，高速性と極めて小さい滞在時間を維持した上で，低消費電力，低ハードウェア量が実現可能となることを明らかにした．

謝辞 本研究を行うにあたり，PARTHENONを御提供頂いたNTTコミュニケーション科学研究所の皆様，及び有益な御助言を頂いた日本ケミコン（株）の横川隆氏，山本和行氏に深く感謝致します．

文　献

[1] 牧野昭二，小泉宣夫，“エコーキャンセラの室内音場における適応特性の改善について，” 信学論（A），vol.J71-A, no.12, pp.2212–2214, Dec. 1988.

[2] K.J. Raghunath and K.K. Parhi, “High-speed RLS using scaled tangent rotation (star),” Proc. IEEE ISCAS'93, Chicago, USA, pp.1959–1962, May 1993.

[3] K.J. Raghunath and K.K. Parhi, “A 100 MHz pipelined RLS adaptive filter,” Proc. IEEE ICASSP'95, Detroit, Michigan, pp.3187–3190, May 1995.

[4] M.D. Meyer and D.P. Agrawal, “A high sampling rate delayed LMS filter architecture,” IEEE Trans. Circuits & Syst.II, vol.40, no.11, pp.727–729, Nov. 1993.

[5] C.L. Wang, “Bit-serial VLSI implementation of delayed LMS transversal adaptive filters,” IEEE Trans. Signal Processing, vol.42, no.8, pp.2169–2175, Aug. 1994.

[6] 松原勝重，西川清史，貴家仁志，“Delayed LMSアルゴリズムに基づくパイプライン適応フィルタ，” 信学論（A），vol.J79-A, no.5, pp.1050–1057, May 1996.

[7] C.F.N. Cowan and J. Mavor, “New digital adaptive-filter implementation using distributed-arithmetic techniques,” IEE Proc., vol.128, Pt.F, no.4, pp.225–230, Aug. 1981.

[8] C.F.N. Cowan, S.G. Smith, and J.H. Elliott, “A digital adaptive filter using a memory-accumulator architecture: Theory and realization,” IEEE Trans. Acoust., Speech & Signal Process., vol.31, no.3, pp.541–549, June 1983.

[9] 高橋　強，豊田真嗣，恒川佳隆，三浦　守，“分散演算型LMS適応フィルタの収束特性解析，” 計測自動制御学会東北支部第178回研究集会，Nov. 1998.

[10] 豊田真嗣，高橋　強，恒川佳隆，三浦　守，“分散演算型LMS適応フィルタのVLSI実現，” 第12回ディジタル信号処理シンポジウム講演論文集，B8-3, pp.645–650, Nov. 1997.

[11] A. Peled and B. Liu, “A new hardware realization of digital filters,” IEEE Trans. Acoust., Speech & Signal Process., vol.22, no.12, pp.456–462, Dec. 1974.

[12] C.H. Wei and J.J. Lou, “Multimemory block structure for implementing a digital adaptive filter using distributed arithmetic,” IEE Proc., vol.133, Pt.G, no.1, pp.19–26, Feb. 1986.

[13] NTTデータ通信株式会社，“PARTHENON User's Manual,” 1990.

[14] A. Harada, K Nishikawa, and H. Kiya, “Pipelined architecture of the LMS adaptive digital filter with the minimum output latency,” IEICE Trans. Fundamentals, vol.E81-A, no.8, pp.1578–1585, Aug. 1998.

付　録

2の補数形式においても式(18)の $\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$ を対角化できることを明らかにする．以下にその導出の過程を示す．

$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$ の行列の乗算は以下のように表される．

$$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k) = \begin{bmatrix} b_0(k) & b_0(k-1) & \cdots & b_0(k-N+1) \\ b_1(k) & b_1(k-1) & \cdots & b_1(k-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ b_{B-1}(k) & b_{B-1}(k-1) & \cdots & b_{B-1}(k-N+1) \end{bmatrix} \begin{bmatrix} b_0(k) & b_0(k-1) & \cdots & b_0(k-N+1) \\ b_1(k) & b_1(k-1) & \cdots & b_1(k-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ b_{B-1}(k) & b_{B-1}(k-1) & \cdots & b_{B-1}(k-N+1) \end{bmatrix}^T \tag{A·1}$$

この行列における各要素は，それぞれ0または1の値をとる．ここで，入力信号が平均0の白色信号であり，入力信号を構成する各ビットが互いに無相関であるという条件のもとで期待値をとると，$E[\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)]$ は次のように表される．

$$E[\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)] = \begin{bmatrix} 0.5N & 0.25N & \cdots & 0.25N \\ 0.25N & 0.5N & \cdots & 0.25N \\ \vdots & \vdots & \ddots & \vdots \\ 0.25N & 0.25N & \cdots & 0.5N \end{bmatrix} \tag{A·2}$$

このように，2の補数形式では $E[\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)]$ の対角化を行うことが不可能である．

そこで，この $E[\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)]$ に対してスケーリングベクトル $\boldsymbol{F}$ を含めて考えることによって，以下のように置き換えることが可能となる．

$$E[\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F}] = 0.25N \begin{bmatrix} 2 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 2 \end{bmatrix} \begin{bmatrix} -2^0 \\ 2^{-1} \\ \vdots \\ 2^{-B+1} \end{bmatrix}$$

$$= 0.25N \begin{bmatrix} -2\times 2^0 + \sum_{i=1}^{B-1} 2^{-i} \\ -1\times 2^0 + 2^{-1} + \sum_{i=1}^{B-1} 2^{-i} \\ \vdots \\ -1\times 2^0 + 2^{-B+1} + \sum_{i=1}^{B-1} 2^{-i} \end{bmatrix} \quad \text{(A·3)}$$

ここで，語長 B がある程度大きい場合には $\sum_{i=1}^{B-1} 2^{-i}$ を近似的に 1 として扱うことができるから

$$E[\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F}]$$

$$\approx 0.25N \begin{bmatrix} -2\times 2^0 + 1 \\ -1\times 2^0 + 2^{-1} + 1 \\ \vdots \\ -1\times 2^0 + 2^{-B+1} + 1 \end{bmatrix}$$

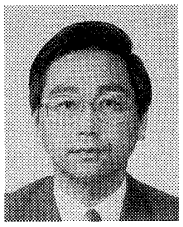
$$= 0.25N \begin{bmatrix} -2^0 \\ 2^{-1} \\ \vdots \\ 2^{-B+1} \end{bmatrix}$$

$$= \begin{bmatrix} 0.25N & 0 & \cdots & 0 \\ 0 & 0.25N & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0.25N \end{bmatrix} \begin{bmatrix} -2^0 \\ 2^{-1} \\ \vdots \\ 2^{-B+1} \end{bmatrix} \quad \text{(A·4)}$$

となる．

このように，2 の補数形式においても $E[\boldsymbol{A}^T(k)\boldsymbol{A}(k)]$ を，対角要素のみが $0.25N$ という値をもつ対角行列に置き換えて考えることが可能となる．

（平成 10 年 11 月 9 日受付，11 年 4 月 12 日再受付）

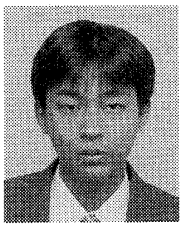


恒川　佳隆　（正員）

昭 55 岩手大・工・電気卒．昭 58 東北大大学院修士課程了．同年岩手大・工・助手．平 6 同大講師．工博．以来，ディジタル信号処理，ディジタル制御の研究に従事．IEEE，計測自動制御学会各会員．

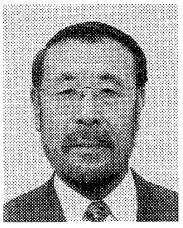
高橋　強　（正員）

昭 62 岩手大・工・電気卒．平 1 同大大学院修士課程了．同年日本無線（株）入社．平 6 岩手県立高度技術専門学院制御システム科技術指導員．平 10 岩手県立産業技術短期大学校技術指導員．ディジタル制御，適応信号処理の研究に従事．計測自動制御学会会員．



豊田　真嗣

平 9 岩手大・工・情報卒．現在同大大学院博士前期課程在学中．適応信号処理に関する研究に従事．



三浦　守　（正員）

昭 37 岩手大・工・電気卒．同年東北大・工・助手．昭 43 秋田工業高専助教授．昭 47 岩手大・工・助教授を経て，昭 60 同大教授．工博．この間，マグネティックス，ディジタル信号処理などに関する研究に従事．平 1 年 10 月より 10 か月間，ウィンザ大（カナダ）及びカリフォルニア大サンタバーバラ校（米国）の客員研究員．電気学会，計測自動制御学会，IEEE，情報処理学会，日本 ME 学会各会員．