計測自動制御学会東北支部 第201回研究集会 (2002.5.22)
資料番号 201-6

分散演算を用いたブロックＬＭＳ適応フィルタの超高速ＶＬＳＩアーキテクチャ

Very High-Speed VLSI Architectures of Block LMS Adaptive Digital Filter Using Distributed Arithmetic

○高橋 強*，樋口直司**，恒川佳隆**，田山典男**

○Kyo Takahashi*, Naoji HIGUCHI**, Yoshitaka TSUNEKAWA**, Norio TAYAMA**

*岩手県立産業技術短期大学校，**岩手大学

*Iwate Industrial Technology Junior College, **Iwate University

キーワード： 分散演算 (distributed arithmetic), ブロックLMSアルゴリズム (block LMS algorithm), 超高速 (very high-speed), 出力滞在時間 (output latency)

連絡先： 〒028-3615 岩手県紫波郡矢巾町大字南矢幅10-3-1 岩手県立産業技術短期大学校電子技術科
高橋 強，Tel.: (019)-697-9082, Fax.: (019)-697-9089, E-mail:kyo@iwate-it.ac.jp

1. はじめに

適応フィルタ (ADF) は，エコーキャンセラ，ノイズキャンセラ，アダプティブイコライザなど幅広くもちいられており，実現の必要性がますます高まっている．そして，実現に際しては，高速性，低消費電力，良好な収束特性，短い出力滞在時間，小規模なハードウエアなど，実に様々な性能が要求される．近年では，適応フィルタは広帯域信号や移動体通信において重要な役割を担うことが期待されおり，この際には高速なサンプリングレートが特に要求される．

これまで，我々は分散演算をLMS適応フィルタに適用した，分散演算形LMS適応フィルタ (DA-ADF) を提案してきた[6)-10)]．分散演算は，内積演算を効率よく求める手法として知られており，ハードウエア規模や消費電力の大きな乗算器を用いない構成が可能である[2)-7)]．また，演算時間はベクトルの次数ではなく語長に依存することも特徴である．これらより，DA-ADFは高次における実現においても，高速性，タップ数に依存しない短い出力滞在時間，良好な収束特性，小規模なハードウエア，低消費電力を同時に満たす構成が可能である．しかし，サンプリングレートは高々数MHz程度であるため，高速なサンプリングレートを要求されるアプリケーションには適していない．

LMSアルゴリズムの高速アルゴリズムであるブロックLMS(BLMS)アルゴリズムが提案されている[11, 12)]．通常のLMSアルゴリズムは入力信号をサンプリング時刻ごとに処理するのに対し，BLMSアルゴリズムではLサンプリング時刻ごとにL個の入力信号を並列に処理する．これにより，1サン

プルあたりの処理時間を減少させることが可能である．*Clark*らは，演算量の削減を目的にアルゴリズムを時間領域から周波数領域に変換して用いた[11]．しかし，高速フーリエ変換を用いて入力信号やパラメータを周波数領域に変換するため，タップ数が増加するにしたがい出力滞在時間が急激に増加するという問題点がある．また，これまで効果的な構成法については検討されていない．

本報告では，時間領域のブロックLMS適応アルゴリズムを用いて，これまで提案されていない分散演算を用いたブロックLMS適応アルゴリズム(BDAアルゴリズム)とそのマルチメモリブロック構造に対するアルゴリズム(MBDAアルゴリズム)を導出する．さらに，高速化を図るために，新たな更新方法であるプライオリティ・アップデートを提案する．MBDAアルゴリズムの収束特性を計算機シミュレーションで評価した結果，マルチメモリブロック構造の分散演算形LMS適応アルゴリズム(MDAアルゴリズム)[6]とほぼ同等の収束特性を有することが明らかになった．さらに，MBDA適応フィルタの効果的なVLSIアーキテクチャを検討した．その結果，提案するアーキテクチャは高速なサンプリングレートを有し，しかも出力滞在時間が短いことが明らかになった．

2. BDAアルゴリズムの導出

2.1 LMSアルゴリズム

WidrowらのLMS(Least mean square)アルゴリズムは次のように表される[1]．入力信号を$x(k)$とするとp次入力信号ベクトル$\boldsymbol{\varphi}(k)$は

$$\boldsymbol{\varphi}(k) = [x(k), x(k-1), \cdots, x(k-p+1)]^T \quad (1)$$

と表される．次に，pタップFIRフィルタの出力$y(k)$は次式で求められる．

$$y(k) = \boldsymbol{\varphi}^T(k)\,\boldsymbol{w}(k) \quad (2)$$

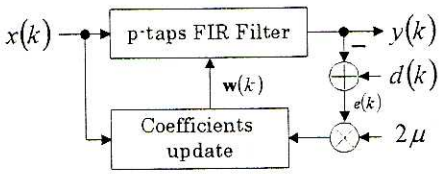

Fig. 1 Block diagram of LMS-ADF.

ここで，$\boldsymbol{w}(k)$はFIRフィルタのタップ係数で

$$\boldsymbol{w}(k) = [w_0(k), w_1(k), \cdots, w_{p-1}(k)]^T \quad (3)$$

である．誤差信号を$e(k)$とすると，LMSアルゴリズムは次の様に表される．

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + 2\mu e(k)\,\boldsymbol{\varphi}(k) \quad (4)$$

なお，$d(k)$は所望信号を表し，

$$e(k) = d(k) - y(k) \quad (5)$$

である．LMSアルゴリズムはサンプリング時刻kごとに出力計算と更新動作を実行する．LMS適応フィルタの基本構成をFig. 1に示す．

2.2 BLMSアルゴリズム

ブロックLMSアルゴリズムは，Lサンプル時刻毎にL個の入力信号を並列に処理する．これにより，1入力サンプルあたりのサンプリング周期を$1/L$に短縮することが可能になる．ブロック長L，ブロック番号j，タップ数pに対して，入力信号ベクトル$\boldsymbol{\varphi}_{j,i}$を次の様に表す．

$$\boldsymbol{\varphi}_{j,i} = [x_{j,i}, x_{j,(i-1)}, \cdots, x_{j,(i-p+1)}]^T,$$

ここで，サンプリング時刻kは，

$$k = jL + i, \quad i = 0, -1, \cdots, -L+1,$$

であり，iはブロック内における時刻を表す．BLMSアルゴリズムの入力信号マトリクスは次の様に表される．

$$\boldsymbol{\Gamma}_j = [\boldsymbol{\varphi}_{j,0}, \boldsymbol{\varphi}_{j,(-1)}, \cdots, \boldsymbol{\varphi}_{j,(-L+1)}]^T$$

$$= \begin{bmatrix} x_{j,0} & x_{j,(-1)} & \cdots & x_{j,(-L+1)} \\ x_{j,(-1)} & x_{j,(-2)} & \cdots & x_{j,(-L)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{j,(-p+1)} & x_{j,(-p)} & \cdots & x_{j,(-L-p+2)} \end{bmatrix}^T$$

出力信号$\boldsymbol{y}_j$と誤差信号$\boldsymbol{e}_j$は，次式で求められる．

$$\boldsymbol{y}_j = \boldsymbol{\Gamma}_j \boldsymbol{w}_j,$$

$$\boldsymbol{e}_j = \boldsymbol{d}_j - \boldsymbol{y}_j.$$

ここで，出力信号$\boldsymbol{y}_j$,所望信号$\boldsymbol{d}_j$,誤差信号$\boldsymbol{e}_j$そして，タップ係数$\boldsymbol{w}_j$は

$$\boldsymbol{y}_j = [y_{j,0}, y_{j,(-1)}, \cdots, y_{j,(-L+1)}]^T, \quad (6)$$

$$\boldsymbol{d}_j = [d_{j,0}, d_{j,(-1)}, \cdots, d_{j,(-L+1)}]^T, \quad (7)$$

$$\boldsymbol{e}_j = [e_{j,0}, e_{j,(-1)}, \cdots, e_{j,(-L+1)}]^T, \quad (8)$$

$$\boldsymbol{w}_j = [w_j(0), w_j(1), \cdots, w_j(-p+1)]^T \quad (9)$$

である．これらより，ブロックLMSアルゴリズムは次のように表される．

$$\begin{aligned} \boldsymbol{w}_{(j+1)} &= \boldsymbol{w}_j + \frac{2\mu_B}{L} \boldsymbol{\Gamma}_j^T \boldsymbol{e}_j \\ &= \boldsymbol{w}_j + \frac{2\mu_B}{L} q_j, \end{aligned}$$

ここで，μ_Bは収束速度と推定精度を決定するステップサイズパラメータ，そして

$$\begin{aligned} \boldsymbol{q}_j &= [q_{j,0}, q_{j,(-1)}, \cdots, q_{j,(-L+1)}]^T \\ &= \boldsymbol{\Gamma}_j^T \boldsymbol{e}_j \\ &= \sum_{i=0}^{-L+1} \boldsymbol{\varphi}_{j,i} e_{j,i}. \end{aligned}$$

である．ブロック長$L=3$のBLMS-ADFの基本構成をFig. 2に示す．入力信号が確定した時点で動作が開始され，FIRフィルタ出力が同時に得られる．次いで，誤差信号の計算とスケーリングを並列に実行して更新値を求め，これらの更新値を加えた後にタップ係数を更新する．なお，タップ係数は各FIRフィルタに共通である．LMS-ADFとBLMS-ADF($L=3$)の動作タイミングをFig. 3に示す．通常のLMS-ADFはサンプリング時刻ごとに出力計算と係数更新動作を行うのに対して，BLMS-ADFは3サンプリング時刻ごとに適応動作を並列に実行する．したがって，BLMS-ADFは1サンプルあたりの処理時間を1/3に短縮することが可能になる．

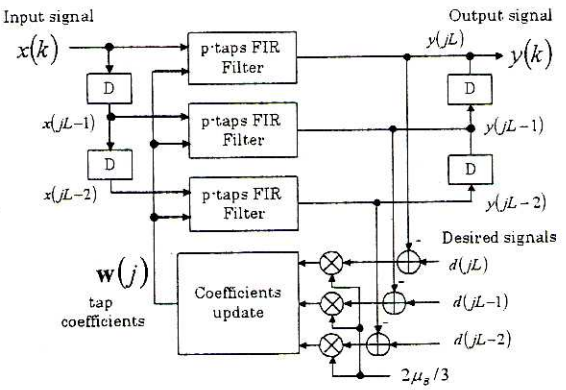

Fig. 2 Block diagram of BLMS-ADF with L=3.

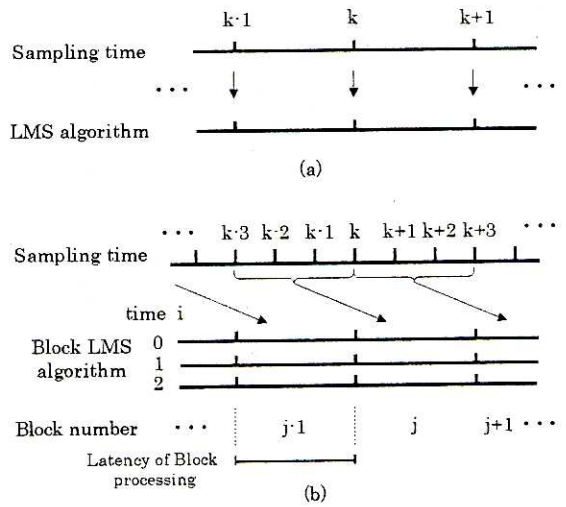

Fig. 3 Comparison of processing timing between LMS and block LMS algorithm. (a) LMS algorithm, (b) Block LMS algorithm with L=3.

2.3 分散演算形ブロックLMSアルゴリズム

分散演算は定係数ベクトルの内積演算を効率よく求める手法としてよく知られていが，係数が変化

する適応フィルタにおいても有効である[2, 3, 4, 5]. 分散演算における内積演算は部分積のシフト加算により実行されるが，p次ベクトルの内積演算における部分積数は，そのパターン数に応じた2^pである．この集合を全適応関数空間と呼び，WAFS(Whole Adaptive Function Space)と表すことにする．WAFSは，定係数の分散演算ではあらかじめ決定されるが，適応フィルタでは逐次的に最適値を推定する．LMSアルゴリズムに分散演算を適用した分散演算形LMS適応フィルタ(DA-ADF)の基本構成をFig. 4に示す．出力信号$y(k)$は，WAFS(RAMを用いて実現される)から指定された部分積を語長回数だけ読み出し，これらをシフト加算することにより得られる．そして，WAFSの部分積はスケーリングされた誤差信号を用いて更新される．この際，スケーリング値を2のべき乗で近似することにより，乗算器を用いない構成が可能になる．なお，WAFSの要素を指定するアドレス信号には，入力信号ベクトルのビットパターンから構成されるp次のアドレスベクトルを用いる．更新式と出力計算式を以下に示す．

$$\boldsymbol{P}(k+1) = \boldsymbol{P}(k) + 0.5\mu pe(k)\,\boldsymbol{F}, \qquad (10)$$

$$y(k) = \boldsymbol{F}^T\,\boldsymbol{P}(k). \qquad (11)$$

ここで，

$$\boldsymbol{P}(k) = [p_0(k), p_1(k), \cdots, p_{B-1}(k)]^T, \qquad (12)$$

$$\boldsymbol{F} = [-2^0, 2^{-1}, \ldots, 2^{-(B-1)}]^T \qquad (13)$$

である．

2.3.1 BDAアルゴリズム

BDAアルゴリズムは通常のBLMSアルゴリズムに分散演算を適用して，次のように求められる．p次の入力信号ベクトル$\boldsymbol{\varphi}_{j,i}$を次のように表す．

$$\boldsymbol{\varphi}_{j,i} = \boldsymbol{A}_{j,i}\,\boldsymbol{F}, \quad i = 0, 1, \cdots, -L+1 \qquad (14)$$

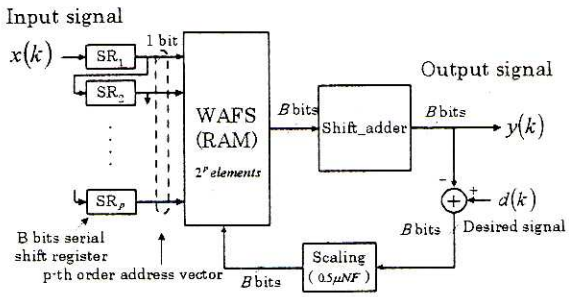

Fig. 4 Block diagram of DA-ADF.

$\boldsymbol{A}_{j,i}$は入力信号のビットパターンを要素に持つアドレスマトリクスであり，$\boldsymbol{F}$はスケーリングベクトルである．すなわち，

$$\boldsymbol{A}_{j,i} = \begin{bmatrix} b_{j,i}(0) & \cdots & b_{j,(i-p+1)}(0) \\ b_{j,i}(1) & \cdots & b_{j,(i-p+1)}(1) \\ \vdots & \ddots & \vdots \\ b_{j,i}(B-1) & \cdots & b_{j,(i-p+1)}(B-1) \end{bmatrix}^T,$$

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

である．ここで$b_{j,i}(l)$は，ブロックjの時刻iにおける入力信号$x_{j,i}$の第lビット$(l = 0, 1, \cdots, B-1)$を表す．また，アドレスマトリクスの列ベクトル

$$\boldsymbol{Av}_{j,i}(l) = [b_{j,i}(l), b_{j,(i-1)}(l), \cdots, b_{j,(i-p+1)}(l)]^T, \quad l = 0, 1, \cdots, B-1$$

をアドレスベクトルと呼び，その値を次のように定義する．

$$Av_{j,i}(m) = \boldsymbol{Av}_{j,i}^T(m)\,\boldsymbol{F}_A$$
$$\boldsymbol{F}_A = [2^{(p-1)}, 2^{(p-2)}, \cdots, 2^0]^T$$

分散演算における内積演算では，アドレスベクトルに対して部分積が定義されるため，アドレスベクトルは出力計算と更新動作においてWAFSの要素を指定するために用いられる．これらの関係を用いて，通常のBLMSアルゴリズムは次のように表される．

$$\boldsymbol{w}_{(j+1)} = \boldsymbol{w}_j + \frac{2\mu_B}{L}\sum_{i=0}^{-L+1} \boldsymbol{A}_{j,i}\,\boldsymbol{F}e_{j,i}$$

ブロック j 内の時刻 $i=0,1,\cdots,-L+1$ に対してこの式を表すと

$$\begin{aligned}\boldsymbol{w}_{j,(-L+2)} &= \boldsymbol{w}_{j,(-L+1)} \\ &+ \frac{2\mu_B}{L}\boldsymbol{A}_{j,(-L+1)}\boldsymbol{F}e_{j,(-L+1)} \quad (15)\\ &\vdots \\ \boldsymbol{w}_{j,0} &= \boldsymbol{w}_{j,(-1)}+\frac{2\mu_B}{L}\boldsymbol{A}_{j,(-1)}\boldsymbol{F}e_{j,(-1)} \\ & \quad (16)\end{aligned}$$

$$\boldsymbol{w}_{(j+1),(-L+1)} = \boldsymbol{w}_{j,0}+\frac{2\mu_B}{L}\boldsymbol{A}_{j,0}\boldsymbol{F}e_{j,0} \quad (17)$$

となる．ここで，$\boldsymbol{w}_{j,i}$ はブロック j の時刻 i におけるタップ係数ベクトルであり次の関係を有する．

$$\boldsymbol{w}_{(j+1)} = \boldsymbol{w}_{(j+1),(-L+1)}$$

この式の両辺に左から $\boldsymbol{A}_{j,i}^T$ を掛けると，式(15)から式(17)は

$$\begin{aligned}\boldsymbol{A}_{j,(-L+1)}^T\boldsymbol{w}_{j,(-L+2)} &= \boldsymbol{A}_{j,(-L+1)}^T\boldsymbol{w}_{j,(-L+1)} \\ &+\frac{2\mu_B}{L}\boldsymbol{A}_{j,(-L+1)}^T\boldsymbol{A}_{j,(-L+1)}\boldsymbol{F}e_{j,(-L+1)} \quad (18)\\ &\vdots \\ \boldsymbol{A}_{j,(-1)}^T\boldsymbol{w}_{j,0} &= \boldsymbol{A}_{j,(-1)}^T\boldsymbol{w}_{j,(-1)} \\ &+\frac{2\mu_B}{L}\boldsymbol{A}_{j,(-1)}^T\boldsymbol{A}_{j,(-1)}\boldsymbol{F}e_{j,(-1)} \quad (19)\\ \boldsymbol{A}_{j,0}^T\boldsymbol{w}_{(j+1),(-L+1)} &= \boldsymbol{A}_{j,0}^T\boldsymbol{w}_{j,0} \\ &+\frac{2\mu_B}{L}\boldsymbol{A}_{j,0}^T\boldsymbol{A}_{j,0}\boldsymbol{F}e_{j,0} \quad (20)\end{aligned}$$

となる．ここで，

$$\begin{aligned}\boldsymbol{P}_{j,i}^i &\equiv \boldsymbol{A}_{j,i}^T\boldsymbol{w}_{j,i} \\ &= [p_{j,i}(Av_{j,i}(0)),\cdots,p_{j,i}(Av_{j,i}(B-1))]^T,\end{aligned}$$

$$\begin{aligned}\boldsymbol{P}_{j,(i+1)}^i &\equiv \boldsymbol{A}_{j,i}^T\boldsymbol{w}_{j,(i+1)} \\ &= [p_{j,(i+1)}(Av_{j,i}(0)),\cdots,p_{j,(i+1)}(Av_{j,i}(B-1))]^T\end{aligned}$$

と定義する．なお，$\boldsymbol{P}_{j,i}^{i'}$ はB次のベクトルで，アドレスマトリクス $\boldsymbol{A}_{j,i'}^T$ に対するWAFSの部分集合である．この部分集合を適応関数空間と呼び，AFS(Adaptive Function Space)と表すことにする．

これらの定義より，式(18)から式(20)は次のように表される．

$$\begin{aligned}\boldsymbol{P}_{j,(-L+2)}^{(-L+1)} &= \boldsymbol{P}_{j,(-L+1)}^{(-L+1)} \\ &+\frac{2\mu_B}{L}\boldsymbol{A}_{j,(-L+1)}^T\boldsymbol{A}_{j,(-L+1)}\boldsymbol{F}e_{j,(-L+1)} \quad (21)\\ &\vdots \\ \boldsymbol{P}_{j,0}^{(-1)} &= \boldsymbol{P}_{j,(-1)}^{(-1)} \\ &+\frac{2\mu_B}{L}\boldsymbol{A}_{j,(-1)}^T\boldsymbol{A}_{j,(-1)}\boldsymbol{F}e_{j,(-1)} \quad (22)\end{aligned}$$

$$\boldsymbol{P}_{(j+1),(-L+1)}^0 = \boldsymbol{P}_{j,0}^0+\frac{2\mu_B}{L}\boldsymbol{A}_{j,0}^T\boldsymbol{A}_{j,0}\boldsymbol{F}e_{j,0} \quad (23)$$

入力信号を平均0の白色信号と仮定すると，$\boldsymbol{A}_{j,i}^T\boldsymbol{A}_{j,i}$ の平均値は

$$E[\boldsymbol{A}_{j,i}^T\boldsymbol{A}_{j,i}] = 0.25p\boldsymbol{F}$$

となり[6]，これを式(21)から式(23)に代入すると，次のように簡略化される．

$$\begin{aligned}\boldsymbol{P}_{j,(-L+2)}^{(-L+1)} &= \boldsymbol{P}_{j,(-L+1)}^{(-L+1)}+\boldsymbol{u}_{j,(-L+1)} \\ & \quad (24)\\ &\vdots \\ \boldsymbol{P}_{j,0}^{(-1)} &= \boldsymbol{P}_{j,(-1)}^{(-1)}+\boldsymbol{u}_{j,(-1)} \quad (25)\\ \boldsymbol{P}_{(j+1),(-L+1)}^0 &= \boldsymbol{P}_{j,0}^0+\boldsymbol{u}_{j,0} \quad (26)\end{aligned}$$

ここで，

$$\begin{aligned}\boldsymbol{u}_{j,i} &= 0.5p\frac{\mu_B}{L}\boldsymbol{F}e_{j,i} \\ &= [u_{j,i}(0),u_{j,i}(1),\cdots,u_{j,i}(B-1)]^T \\ &\quad i=0,-1,\cdots,-L+1\end{aligned}$$

である．これら L 個の更新式は並列に実行され，共有するWAFSの要素をアドレスベクトルによって順次選択して更新する．また，誤差信号のスケーリング値を2のべき乗で近似することにより，乗算器を用いない構成が可能になる．なお，WAFSは次のように表される．

$$\boldsymbol{P}\boldsymbol{w}_{j,i} = [p_{j,i}(0),p_{j,i}(1),\cdots,p_{j,i}(2^p-1)]^T$$

次に，BLMSアルゴリズムの出力方程式は

$$\boldsymbol{y}_j = [y_{j,0},y_{j,(-1)},\cdots,y_{j,(-L+1)}]^T$$

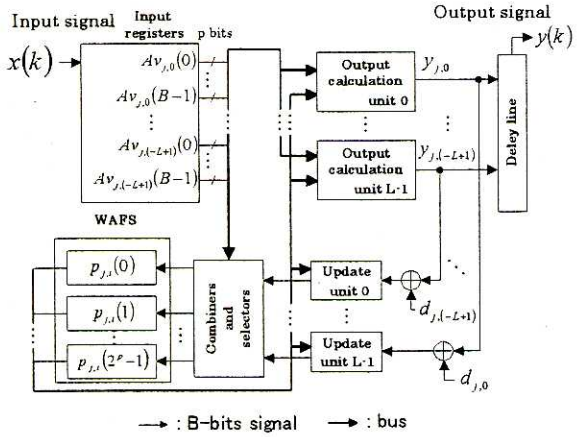

Fig. 5 Block diagram of BDA-ADF.

$$= [\boldsymbol{\varphi}_{j,0}^T \boldsymbol{w}_j, \boldsymbol{\varphi}_{j,(-1)}^T \boldsymbol{w}_j, \cdots, \boldsymbol{\varphi}_{j,(-L+1)}^T \boldsymbol{w}_j]^T$$

であるが，式(14)を適用して

$$\begin{aligned} \boldsymbol{y}_j &= [\boldsymbol{F}^T \boldsymbol{A}_{j,0}^T \boldsymbol{w}_j, \cdots, \boldsymbol{F}^T \boldsymbol{A}_{j,(-L+1)}^T \boldsymbol{w}_j]^T \\ &= [\boldsymbol{F}^T \boldsymbol{P}_{j,(-L+1)}^0, \cdots, \boldsymbol{F}^T \boldsymbol{P}_{j,(-L+1)}^{(-L+1)}]^T \end{aligned} \quad (27)$$

となる．ここで，

$$\boldsymbol{P}_j^i \equiv \boldsymbol{P}_{j,(-L+1)}^i, \quad i = 0, -1, \cdots, -L+1$$

と定義して，これを式(27)に適用する．これより，出力計算式は

$$\begin{aligned} \boldsymbol{y}_j &= [y_{j,0}, y_{j,(-1)}, \cdots, y_{j,(-L+1)}]^T \\ &= [\boldsymbol{F}^T \boldsymbol{P}_j^0, \boldsymbol{F}^T \boldsymbol{P}_j^{(-1)}, \cdots, \boldsymbol{F}^T \boldsymbol{P}_j^{(-L+1)}]^T \end{aligned} \quad (28)$$

となり，各要素は同時に計算可能である．BDA-ADFの基本構成を Fig. 5 に示す．

2.3.2 プライオリティ・アップデート

式(24)~式(26)の並列更新動作を，従来のDA-ADFと同様に更新値ベクトル$\boldsymbol{u}_{j,i}$の第0要素あるいは第$B-1$要素から順次実行すると，同じ要素が$n(=2, \cdots, L)$個の更新式で同時に更新対象になる場合がある．この際には，これらの更新値和を求めてから要素を更新する必要がある．しかし，nに応じて加算処理時間は異なるため，パイプライン処理には適していない．また，同時に選択される要素が複数存在する場合もあるため，WAFSの要素数に等しい2^p個の加算器群が必要になる．これは，ハードウエア規模と消費電力の増加につながる．

一方，更新値ベクトル$\boldsymbol{u}_{i,j}$の要素は誤差信号に対するスケーリングベクトルの重み付けで決定されるため，絶対値が大きい要素ほどアルゴリズムが収束するための効果が大きい．そこで，個々の更新式においてWAFSのある要素を更新する際に，最も絶対値の大きい更新値を用いて更新動作を実行することを考える．さらに，更新動作を規則的に実行するために式(24)から式(26)をWAFSを対象とする更新式に拡張する．

$$\boldsymbol{P}\boldsymbol{w}_{j,(-L+2)} = \boldsymbol{P}\boldsymbol{w}_{j,(-L+1)} + \boldsymbol{U}_{j,(-L+1)} \quad (29)$$

$$\vdots$$

$$\boldsymbol{P}\boldsymbol{w}_{j,0} = \boldsymbol{P}\boldsymbol{w}_{j,(-1)} + \boldsymbol{U}_{j,(-1)} \quad (30)$$

$$\boldsymbol{P}\boldsymbol{w}_{(j+1),(-L+1)} = \boldsymbol{P}\boldsymbol{w}_{j,0} + \boldsymbol{U}_{j,0} \quad (31)$$

ここで，$\boldsymbol{P}\boldsymbol{w}_{j,i}$と$\boldsymbol{U}_{j,i}$は，それぞれブロック$j$の時刻$i$におけるWAFSとその更新値を表し，

$$\begin{aligned} \boldsymbol{U}_{j,i} &= [u_{0,i}, u_{1,i}, \cdots, u_{(2^p-1),i}]^T \\ &= \boldsymbol{T}_{j,i}[0.5p\frac{\mu_B}{L}\boldsymbol{F}e_{j,i}] \end{aligned}$$

であり，$\boldsymbol{T}_{j,i}$は$2^p \times B$の変換行列である．ここで，更新式の有するB個のアクセスベクトルの中で，いくつかのアクセスベクトルが同じ値を有する場合には，それらのなかで最も大きな更新値を有するアクセスベクトルを選択する．これより，$\boldsymbol{T}_{j,i}$の列ベクトルの要素には一つの'1'が存在し，その他は全て'0'となる．なお，入力信号ベクトルが確定した時点で適応動作は開始されるので，全てのアクセスベクトルも確定している．これにより，WAFSに対する更新値$\boldsymbol{U}_{j,i}$は同時使用が可能になる．そして，更新動作をWAFSの要素$p_{j,i}(0) \sim p_{j,i}(2^p-1)$

の順に実行する．この更新方法をプライオリティ・アップデートと呼ぶことにする．更新式は，式(29)から式(31)へ順次代入して

$$\boldsymbol{Pw}_{(j+1),(-L+1)} = \boldsymbol{Pw}_{j,(-L+1)} + \sum_{i=0}^{-L+1} \boldsymbol{U}_{j,i}$$

となり，さらに

$$\begin{aligned}\boldsymbol{Pw}_{(j+1)} &\equiv \boldsymbol{Pw}_{(j+1),(-L+1)} \\ &= [p_{(j+1)}(0), \cdots, p_{(j+1)}(2^p-1)]^T, \\ \boldsymbol{Pw}_j &\equiv \boldsymbol{Pw}_{j,(-L+1)} \\ &= [p_j(0), \cdots, p_j(2^p-1)]^T.\end{aligned}$$

と置くことにより，次式のように表される．

$$\boldsymbol{Pw}_{(j+1)} = \boldsymbol{Pw}_j + \sum_{i=0}^{-L+1} \boldsymbol{U}_{j,i} \qquad (32)$$

2.4 マルチメモリブロック構造

BDAアルゴリズムのWAFSの容量は2^pwordsであるため，タップ数pが増加するにしたがい容量が急激に増加する．これにより，ハードウエア規模と消費電力は増加し，収束速度が大幅に劣化する．この問題を解決するために，マルチメモリブロック構造(Multi-memory block structure)が提案されている[4]．マルチメモリブロック構造では，p次のタップ係数をM個に分割した(p/M)次のベクトルに対してそれぞれWAFSを定義する．これにより，個々の容量は$2^{(p/M)}$words，総容量は$M \cdot 2^{(p/M)}$wordsと小容量になるため，小規模なハードウエアと低消費電力を実現でき，収束速度を大幅に改善することが可能である．

マルチメモリブロック構造を適用したBDAアルゴリズム(MBDAアルゴリズム)は以下のように表される．分割されたタップ係数とWAFSを次のように定義する．

$$\begin{aligned}\boldsymbol{w}_j^m &= [w_j^m(0), w_j^m(1), \cdots, w_j^m(R-1)]^T \\ \boldsymbol{Pw}_{j,i}^m &= [p_{j,i}^m(0), p_{j,i}^m(1), \cdots, p_{j,i}^m(2^R-1)]^T \\ & \qquad m = 0, 1, \cdots, M-1, \\ & \qquad i = 0, -1, \cdots, -L+1\end{aligned}$$

ここで，

$$R = p/M$$

である．ブロック番号jの時刻iにおける適応関数空間と出力信号はそれぞれ次の様に表される．

$$\begin{aligned}\boldsymbol{P}_j^{i,m} &= \boldsymbol{A}_{j,i}^{m\,T}\, \boldsymbol{w}_j^m, \\ y_{j,i} &= \sum_{m=0}^{M-1} \boldsymbol{F}^T \boldsymbol{P}_j^{i,m}, \\ & \qquad m = 0, 1, \cdots, M-1\end{aligned}$$

ここで，

$$\boldsymbol{A}_{j,i}^m = \begin{bmatrix} b_{j,i}^m(0) & \cdots & b_{j,(i-R+1)}^m(0) \\ b_{j,i}^m(1) & \cdots & b_{j,(i-R+1)}^m(1) \\ \vdots & \ddots & \vdots \\ b_{j,i}^m(B-1) & \cdots & b_{j,(i-R+1)}^m(B-1) \end{bmatrix}^T$$

である．MBDAアルゴリズムの更新式は

$$\boldsymbol{P}_{j,(-L+2)}^{(-L+1),m} = \boldsymbol{P}_{j,(-L+1)}^{(-L+1),m} + 0.5R\frac{\mu_B}{L}\boldsymbol{F}e_{j,(-L+1)} \qquad (33)$$

$$\vdots$$

$$\boldsymbol{P}_{j,0}^{(-1),m} = \boldsymbol{P}_{j,(-1)}^{(-1),m} + 0.5R\frac{\mu_B}{L}\boldsymbol{F}e_{j,(-1)} \qquad (34)$$

$$\boldsymbol{P}_{(j+1),(-L+1)}^{0,m} = \boldsymbol{P}_{j,0}^{0,m} + 0.5R\frac{\mu_B}{L}\boldsymbol{F}e_{j,0} \qquad (35)$$

である．これらの更新式は，次式で表される第m番のWAFSの中からB個の要素を選択して更新することを表している．

$$\boldsymbol{Pw}_{j,i}^m = [p_{j,i}^m(0), p_{j,i}^m(1), \cdots, p_{j,i}^m(2^R-1)]^T$$

式(33)から式(35)をWAFSに拡張してプライオリティ・アップデートを適用する．

$$\boldsymbol{Pw}_{j,(-L+2)}^m = \boldsymbol{Pw}_{j,(-L+1)}^m + \boldsymbol{U}_{j,(-L+1)}^m \qquad (36)$$

$$\vdots$$

$$\boldsymbol{Pw}_{j,0}^m = \boldsymbol{Pw}_{j,(-1)}^m + \boldsymbol{U}_{j,(-1)}^m \qquad (37)$$

$$\boldsymbol{Pw}_{(j+1),(-L+1)}^m = \boldsymbol{Pw}_{j,0}^m + \boldsymbol{U}_{j,0}^m \qquad (38)$$

Table 1　Step size parameters.M indicates the division number.

M	MDA algorithm	MBDA algorithm
1	2^4	2^2
2	2^3	2^2
4	2^3	2^1
8	2^3	2^0

ここで，$\boldsymbol{U}_{j,i}^m$はm番目のWAFSに対する更新値を表しており，

$$\begin{aligned}\boldsymbol{U}_{j,i}^m &= [u_{0,i}^m, u_{1,i}^m, \cdots, u_{(2^R-1),i}^m]^T \\ &= \boldsymbol{T}_{j,i}^m[0.5R\frac{\mu_B}{L}\boldsymbol{F}e_{j,i}]\end{aligned}$$

である．$\boldsymbol{T}_{j,i}^m$は，$2^R \times B$の変換行列である．更新式は，式(36)から式(38)へ順次代入して

$$\boldsymbol{Pw}_{j,1}^m = \boldsymbol{Pw}_{j,(-L+1)}^m + \sum_{i=0}^{-L+1} \boldsymbol{U}_{j,i}^m$$

となり，さらに

$$\begin{aligned}\boldsymbol{Pw}_{(j+1)}^m &\equiv \boldsymbol{Pw}_{(j+1),(-L+1)}^m \\ &= [p_{(j+1)}^m(0), p_{(j+1)}^m(1), \cdots, p_{(j+1)}^m(2^R-1)]^T, \\ \boldsymbol{Pw}_j^m &\equiv \boldsymbol{Pw}_{j,(-L+1)}^m \\ &= [p_j^m(0), p_j^m(1), \cdots, p_j^m(2^p-1)]^T\end{aligned}$$

と定義することにより，更新式は次式のように表される．

$$\boldsymbol{Pw}_{(j+1)}^m = \boldsymbol{Pw}_j^m + \sum_{i=0}^{-L+1} \boldsymbol{U}_{j,i}^m \qquad (39)$$

となる．MBDA-ADFの基本構成をFig. 6に示す．なお，ここではBDA-ADFの基本構成からの変更箇所のみを示している．

3.　計算機シミュレーション

計算機シミュレーションにより，提案するMBDAアルゴリズムの収束特性を検証する．シミュレーションモデルはFig. 7に示されるシステム同定問題である．未知システムはタップ数8の低域通過FIRフィルタ，入力信号は平均0，分散0.05の白色ガウス

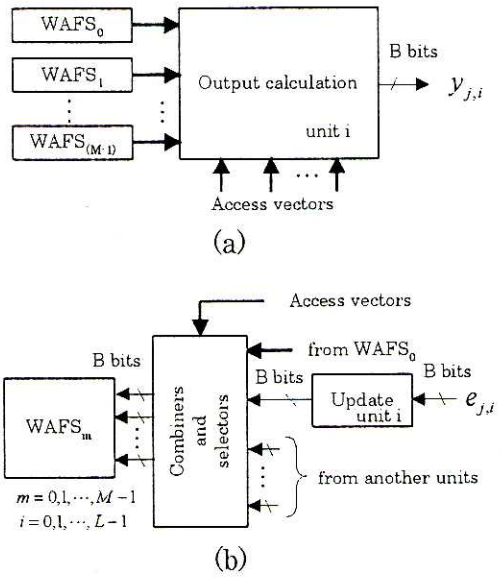

Fig. 6　Block diagram of MBDA-ADF.(a) modification of output calculation. (b) modification of update procedure.

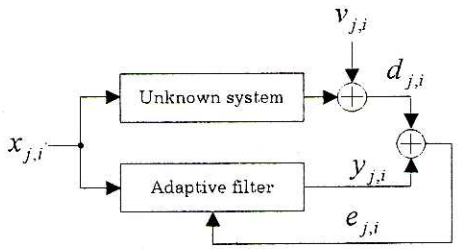

Fig. 7　Simulation model.

信号，そして観測雑音として平均0，分散1.50998×10^{-6}の入力信号とは無相関の白色ガウス信号を加えた．Fig. 8は分割数$M=1,2,4,8$のMDAアルゴリズムの収束特性である．なお，分割数$M=8$のMDAアルゴリズムの収束特性はBLMSアルゴリズムの収束特性と同等である．Fig. 9はプライオリティ・アップデートを用いたMBDAアルゴリズムの収束特性である．なお，ブロック長$L=4$，分割数$M=1,2,4,8$である．Table. 1にシミュレーションに用いたステップサイズを示す．なお，ステップサイズの値は同じMSEを達成し，かつ最も高速な収束速度を示す値を選択した．これらより，提案するMBDAアルゴリズムはMDAアルゴ

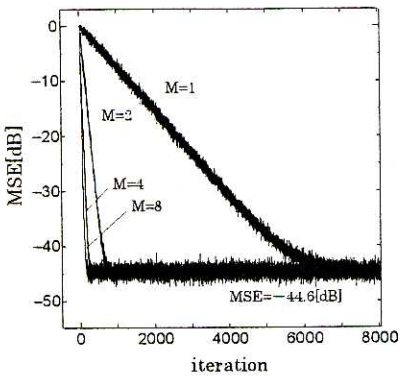

Fig. 8 Convergence characteristics of MDA.

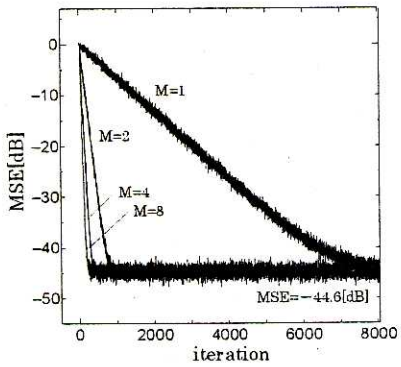

Fig. 9 Convergence characteristics of MBDA using priority update method.

リズムとほぼ同等の収束特性を有している．次に，Fig. 10 は分割数$M=4$，ブロック数$L=1,2,3,4$に対するMBDA-ADFの収束特性である．これより，MBDAアルゴリズムは異なるブロック数に対しても同等の良好な収束特性を有することがわかる．なお，多くのシミュレーションにおいても同様の結果を得ている．

4. VLSIアーキテクチャ

新たな更新方法であるプライオリティ・アップデートを用いたMBDA-ADFの高性能アーキテクチャをFig. 11に示す．なお，ブロック長と分割数はどちらも2とした．入力信号レジスタは$(p+L-1)\times B$個の1bitシフトレジスタ(SR)で構成され，Controllerは入力信号のビットパターンによりSelectorの制御

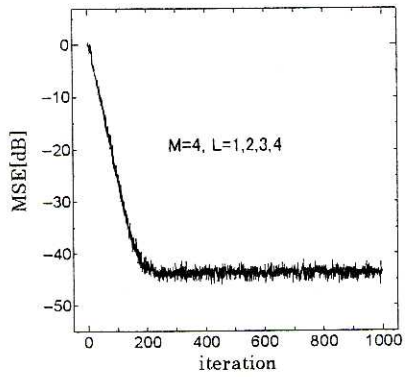

Fig. 10 Convergence characteristics of MBDA using priority update method for various L.

信号を生成する．

MBDAアルゴリズムは，フィルタ出力の計算とWAFSの更新の二つのステージからなる．

出力計算

まず，$WAFS_0$と$WAFS_1$の要素をSelector-0によって選択する．なお，Selector-0はアドレスベクトルによって制御される．そして，選択された2つの要素を加えた後にシフト加算を行う．この操作をB回行うことにより，二つの出力信号が同時に得られる．これまで，WAFSはRAM(Random Access Memory)を用いて実現されてきた．しかし，RAMでは複数要素を同時に読み出すことができないため，出力計算における並列処理は不可能である．そこで，本構成ではWAFSをLatchを用いて実現することにより並列処理を可能にしている．

WAFSの更新

まず，所望信号から出力信号を減算して誤差信号が得られ，次いで誤差信号はScalerにより0～$B-1$ビットの右シフトされた信号が出力される．WAFSの更新は，$p^m_{j,i}(0) \sim p^m_{j,i}(2^R-1)$の順に実行する．まず，WAFSの要素はSelector-0，そして更新値（シフトされた誤差信号）はSelector-1によって選択される．そして，これらを加算した後，Selector-2によって対象となる要素を選択して更新する．

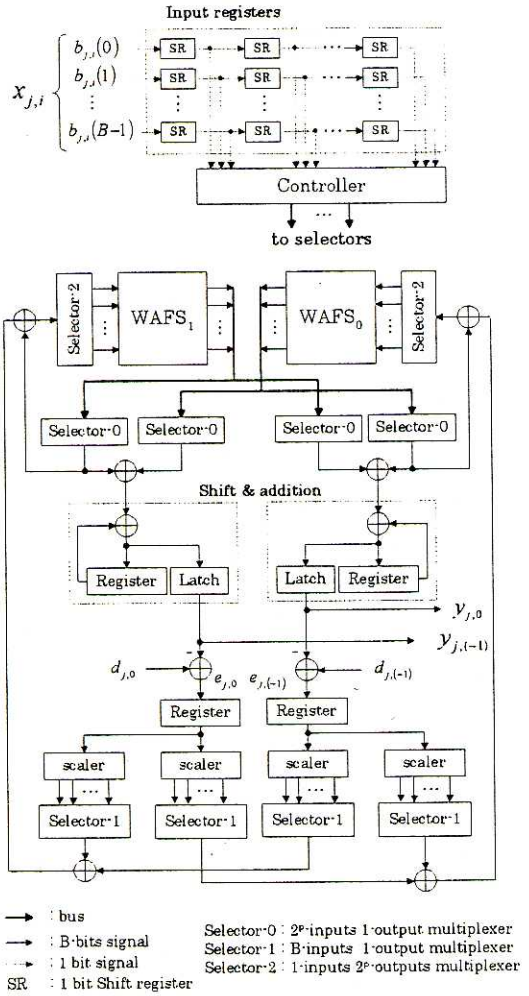

Fig. 11 Block diagram of proposed MBDA-ADF with L=M=2.

タイミングチャートを Fig. 12 に示す．L個の入力信号サンプルに対するサンプリング周期Ts^LとサンプリングレートFs^Lは

$$Ts^L = (\lceil log_2(L+1) \rceil + \lceil log_2 M \rceil + 2^R + B + 1) \cdot \tau_p,$$
$$Fs^L = 1/Ts^L,$$

である．1サンプルあたりのサンプリング周期Tsは，Lで割り

$$Ts = Ts^L / L$$

となるので，サンプリングレートFsは

$$Fs = 1/Ts$$

である．出力滞在時間τ_oは，Ts^Lと出力計算時間

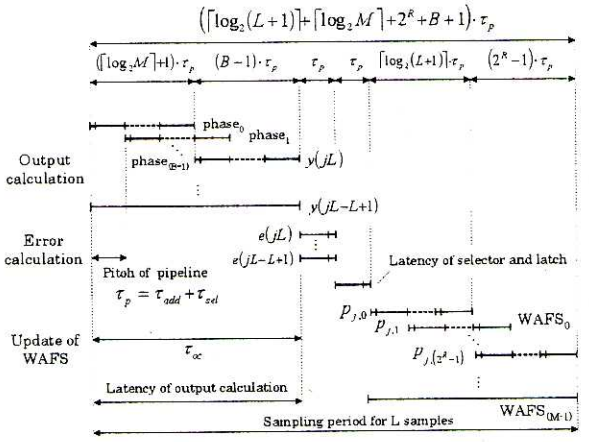

Fig. 12 Timing chart of proposed MBDA-ADF.

τ_{oc}の和であるので，

$$\tau_{oc} = (\lceil log_2 M \rceil + B) \cdot \tau_p$$
$$\tau_o = Ts + \tau_{oc}$$
$$= (\lceil log_2(L+1) \rceil + 2\lceil log_2 M \rceil + 2^R + 2B + 1) \cdot \tau_p$$

となる．ここで，$\lceil x \rceil$はx以上の最小の整数であり，τ_{add}とτ_{sel}はそれぞれ加算器とセレクタの出力滞在時間を表す．ブロック長に対するサンプリングレートと出力滞在時間を Table. 2 に示す．なお，タップ数は$p = 128$，$\tau_{add} = 15ns$，$\tau_{sel} = 7ns$とした．ブロック長Lが増加するにしたがって，サンプリングレートと出力滞在時間もともに増加するが，サンプリングレートの増加に対して出力滞在時間の増加は非常に小さいことがわかる．これは，並列処理のため出力計算時間はブロック長に依存しないこと，そしてWAFSの更新動作においては，L個の更新値を加算するために"Tree-adder"（加算器をツリー状に配置した構造を有する）を使用して更新時間の増加を極力抑制しているためである．分割数が32から64に増加すると，サンプリングレートは増加し，出力滞在時間は減少している．マルチメモリブロック構造の出力計算においては，M個のWAFSから選択される要素を加算する処理時間が新たに追加される．これは，サンプリングレートの減少と出力滞在時間の増加をも

Table 2 Sampling rate Fs and output latency τ_o of MBDA-ADF for M=32,64, p=128 and B=16.

	M=32(R=4)		M=64(R=2)	
L	Fs [MHz]	τ_o [ns]	Fs [MHz]	τ_o [ns]
1	1.16	1326.0	1.61	1105.0
2	2.26	1348.1	3.12	1127.1
4	4.41	1370.2	6.03	1149.2
8	8.62	1392.3	11.68	1171.3
16	16.84	1414.4	22.62	1193.4
32	32.90	1436.5	43.88	1215.5

たらす．しかし，本構成では"Tree-adder"を用いるため，分割による出力計算時間τ_{oc}の増加は極めて少ない．一方，更新時間はWAFSの容量に依存しており

$$(\lceil log_2(L+1) \rceil + 2^R - 1) \cdot \tau_p$$

であるが，分割数の増加により各適応関数空間の容量(2^R)が減少するために更新時間も減少する．これらより，分割数$M = 64$においてサンプリングレートが増加し，出力滞在時間は減少するのである．さらに大きなブロック長を選択すると，MBDA-ADFはより高速なサンプリングレートを実現可能である．

Table. 3にClarkらの周波数領域アルゴリズムによるBLMS-ADF[11]との比較を示す．Clarkらの周波数領域におけるBLMS-ADFは，高速フーリエ変換を用いるためにタップ数とブロック長を同じ2のべき乗に選択する必要がある．そこで，比較においてはブロック長Lとタップ数pを2から128までの2のべき乗を選択した．我々の提案するMBDA-ADFは，$L(=p) = 128$において165.5MHzのサンプリングレートを達成可能である．これは，BLMS-ADFの約277.7%である．出力滞在時間もBLMS-ADFより短く，約39.5%の1259.7nsである．また，$L(=p)$に対する増加も極めて少ない．より大きな$L(=p)$に対して，さらに高速なサンプリング・レートを達成可能である．

Table 3 Comparison of sampling rate Fs and output latency τ_o between MBDA-ADF with $M = 64$ and BLMS-ADF. The word length B=16.

	MBDA		BLMS	
L(=p)	Fs [MHz]	τ_o [ns]	Fs [MHz]	τ_o [ns]
2	3.93	861.9	3.16	918.0
4	7.24	928.2	4.52	1296.0
8	13.4	994.5	7.04	1674.0
16	25.0	1060.8	11.5	2052.0
32	46.7	1127.1	19.5	2430.0
64	87.8	1193.4	33.8	2808.0
128	165.5	1259.7	59.6	3186.0

5. あとがき

本報告では，分散演算形ブロックLMSアルゴリズム(BDAアルゴリズム)と，マルチメモリブロック構造を適用したアルゴリズム(MBDAアルゴリズム)を初めて提案した．さらに，パイプライン処理向きの更新方法であるプライオリティ・アップデートを提案した．提案したアルゴリズムは良好な収束速度と推定精度を有することを計算機シミュレーションにより確認した．さらに，効果的なVLSIアーキテクチャを検討し，サンプリングレートと出力滞在時間を評価した．提案するMBDA-ADFは，タップ数128，分割数64，ブロック長128において165.5MHzのサンプリングレートを達成可能である．これは，従来法の約277.7%である．また，出力滞在時間も1259.7nsと非常に短く，従来法の39.5%である．このように，提案したMBDA-ADFはブロックLMSアルゴリズムの本来有する並列性を有効に利用した高速性と短い出力滞在時間を有し，さらに分散演算の適用によりタップ数に対する出力滞在時間の増加を最小限に抑えることが可能な高性能適応フィルタである．

分散演算の適用はハードウエア規模と消費電力においても有効であるため，今後は詳細なVLSI評価を行い，さらに収束条件などの理論解析も行う予定である．

参考文献

1) B. Widrow, J. R. Glover,Jr., J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong,Jr., and R. C. Goodlin,"Adaptive noise cancelling : Principles and applications," Proc. IEEE, vol.63, pp.1692–1716, Dec.1975.

2) 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.

3) 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.

4) C. H. Wei, 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.

5) 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, Jun. 1983.

6) Y. Tsunekawa, K. Takahashi, S. Toyoda , M. Miura,"High-Performance VLSI Architecture of Multiplierless LMS Adaptive Filters Using Distributed Arithmetic," IEICE Trans. Fundamentals, vol.J82-A, no.10, pp.1518-1528, Oct.1999.

7) K. Takahashi, Y. Tsunekawa, S. Toyoda, M. Miura,"High-Performance Architecture of LMS Adaptive Filters Using Distributed Arithmetic Based on Half-Memory Algorithm," IEICE Trans. Fundamentals, vol.J84-A, no.6, pp.777-787, June.2001.

8) K. Takahashi, Y. Tsunekawa, N. Tayama, K. Seki,"Analysis of the Convergence Condition of LMS Adaptive Filter Using Distributed Arithmetic," Proceedings of ITC-CSCC'01, vol.1, pp.576-579, July.2001.

9) K. Takahashi, Y. Tsunekawa, N. Tayama, K. Seki,"Convergence Condition Analysis of LMS Adaptive Filters Using Distributed Arithmetic," Proceedings of the 187-th SICE Tohoku-Branch Research Convention, vol.187-3, June.2000.

10) K. Takahashi, Y. Tsunekawa, N. Tayama, K. Seki,"Analysis of the Convergence Condition of LMS Adaptive Filter Using Distributed Arithmetic," IEICE TRANS. FUNDAMENTALS, vol.E85-A, NO.6, pp.151-158, JUNE.2002.

11) Gregory A. Clark, Sanjit K. Mitra, Sydney R. Parker,"Block Implementation of Adaptive Digital Filters," IEEE Trans. Circuits and Syst., vol. CAS-28, pp.584–592, June. 1981.

12) A. Feuer, "Performance Analysis of the Block Least Mean Square Algorithm," IEEE Trans. Circuits and Syst., vol. CAS-32, pp.960–963, Sept. 1985.