こう配形適応フィルタのパイプライン実現

西川　清史†　　貴家　仁志†

Pipeline Implementation of Gradient-Type Adaptive Filters

Kiyoshi NISHIKAWA† and Hitoshi KIYA†

あらまし　本論文では，こう配形適応フィルタのアルゴリズムレベルでのパイプライン実現法について述べる．LSI 化の際に，パイプライン実現を行うことにより，フィルタの最長経路（クリティカルパス）を短縮でき，スループット特性を向上させることが可能となる．また，クリティカルパスの短縮は，使用される演算器の制約を緩和し，消費電力の抑圧にも効果がある．こう配形適応フィルタのパイプライン実現の評価基準として，スループット・収束特性・レイテンシー・演算量などがある．こう配形適応フィルタの一つである，DLMS（delayed LMS）適応フィルタに基づくことで，高いスループットをもつことはもちろん，LMS 適応フィルタと同一の収束特性とレイテンシーをもち，ハードウェア量についても考慮されたパイプライン実現が行えることを示す．

キーワード　適応フィルタ，LMS アルゴリズム，パイプライン実現，低消費電力

1. まえがき

適応フィルタ（adaptive digital filter: ADF）は，通信路や音響系における雑音や残響の除去などの広い分野へ応用されている．また，近年の高帯域な通信やインターネットの普及などに伴い，大量のデータをより高速に処理する要求が生じてきている．一方で，携帯電話やモバイルコンピュータなどの小形携帯機器の発達に伴い，より少ない電力消費で動作する適応フィルタへの要求も高まっている．

これらの要求に対して，その目的の使用に限定して，適応フィルタを専用 LSI で実現することが有効であると考えられている．その際にパイプライン実現を用いることで，より効率的に LSI 化を行うことが検討されている [1]〜[3]．パイプライン実現は，回路を小領域に適切に分割し同時実行化するための技術である．適応フィルタをパイプライン実現することで，フィルタの最長経路（クリティカルパス）の短縮が可能となる．その結果，用いる演算器の性能を一定のままで，単位時間当たりの処理能力，すなわちスループットの向上が可能となる．またクリティカルパスの短縮は，スループット一定の条件の下では，より動作周波数の遅い演算器を利用することを可能とし，この場合回路の消費電力の抑制が可能となる [1]．

適応フィルタのパイプライン実現の研究は，RLS（recursive least squares）アルゴリズムに基づくものより始められた [4]〜[9]．これは，RLS アルゴリズムがシストリックアレーでの実現に適した構成をもつためである．シストリックアレーは，規則的な構成をもち，VLSI での実現に非常に適している [10]．しかしながら，RLS に基づくパイプライン実現は，除算や平方根演算などの複雑な演算が必要となり，ハードウェア規模が大きくなる．また，RLS アルゴリズム自体の有限語長での安定性の問題もあり，実際に適用可能なアプリケーションは，限定されたものとなってしまう．

上記の問題を回避し，適応フィルタのパイプライン実現を可能にするものとして，こう配形適応フィルタ（gradient-type adaptive digital filters; GADF）に基づくパイプライン実現が期待されている [1], [11]〜[13]．GADF としては LMS（least mean square）アルゴリズム [14] に基づく LMS 適応フィルタが最も知られている．しかし，LMS 適応フィルタは遅延器に対する自由度が少なく，それを直接パイプライン実現することは困難であると考えられてきた．このため，GADF のパイプライン実現は DLMS（delayed LMS）[15], [16] と呼ばれる，LMS アルゴリズムの変種アルゴリズムにもとづくものが研究の主流である [17]〜[19]．GADF

† 東京都立大学大学院工学研究科，東京都
Department of Electrical Engineering, Tokyo Metropolitan University, Tokyo, 192–0397 Japan

のパイプライン実現の評価基準として，スループット・収束特性・レイテンシー・演算量などがある．DLMS 適応フィルタを直接パイプライン実現することで，少ない演算量で高いスループット特性を実現することが可能となるが，一方で収束特性が悪化しレイテンシーが増加することが知られている．

この問題に対して，DLMS 適応フィルタに対し補正を行い収束特性の改善を実現する構成が提案されている [20]〜[23]．しかし，この構成は DLMS 適応フィルタの補正のために多くの演算量を必要とする．本論文では，高いスループットを保持したまま LMS 適応フィルタと同一の収束特性及びレイテンシーをもち，少ない演算量で実現する構成を示す．

本論文では，まず **2.** で準備として適応フィルタのパイプライン化に必要な手法について述べる．次に，**3.** において DLMS 適応フィルタに基づく GADF のパイプライン実現について説明し，その問題点を示す．DLMS 適応フィルタのもつ問題の一つである，収束特性を改善するための方法を **4.** で述べる．最後に **5.** で，レイテンシーの減少及びハードウェア量の減少を可能とする構成について述べる．

なお，パイプライン実現の研究にはいろいろなレベルが考えられることに注意しておく．例えば，演算器のゲートレベルでのパイプライン化などがあげられる [24]．本論文では，アルゴリズムレベルでのパイプライン実現のみを扱い，それ以外のレベルのものは取り扱わない．

2. パイプライン実現

まず準備として，パイプライン実現の手法について簡単に述べる．なお以下では次の変数を使用する．

$\tau_{\rm a}$ ・・・ 1 加算に要する処理時間

$\tau_{\rm m}$ ・・・ 1 乗算に要する処理時間

これらの変数には，

$$\tau_{\rm a} \leqq \tau_{\rm m} \tag{1}$$

の関係が成り立つものと仮定する．また本論文では記述を簡潔にするために，信号の伝搬に要する時間や遅延器での処理時間などは無視できるものとする．

2.1 クリティカルパス

本論文では，パイプライン実現されたフィルタの評価に重要なクリティカルパス（critical path: CP）の説明を図 1 の 3 タップの FIR フィルタを例にして与えよう．

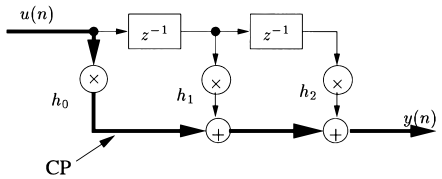

図 1 3 タップ FIR フィルタのシグナルフローグラフ（SFG）

Fig. 1 Signal flow graph (SFG) of a 3-tap FIR filter.

図において太線で示した経路は，回路中もっとも処理時間の長い経路である．この，回路中でもっとも長い経路のことを，CP と呼ぶ．CP での処理時間を $\tau_{\rm cp}$ と表現すると，図 1 の回路では

$$\tau_{\rm cp} = \tau_{\rm m} + 2\tau_{\rm a} \tag{2}$$

となる．式 (2) に示したような，CP での処理時間のことを CP の長さと呼ぶ．CP の長さは，回路の諸特性を決定する重要な要因であることを次に示す．

パイプライン化された回路の評価は，いくつかの特性を用いて行われる．それらの特性は，スループット・必要となるハードウェア量・消費電力などである．このうちスループットとは，単位時間内の処理量と一般には定義される．本論文で扱うディジタルフィルタの場合には，単位時間当たりの出力可能な信号のサンプル数とスループットを定義することができる．すなわち，スループットが高いほど，高いサンプリングレートでサンプリングされた離散時間信号に対して実時間処理することが可能となることを意味する．

一般に，回路の動作周波数 f_{op} は，使用する演算器の性能が一定であるという条件のもとで，

$$f_{op} \sim \frac{1}{\tau_{\rm cp}} \tag{3}$$

のように $\tau_{\rm cp}$ に逆比例する．したがって，CP の短縮により，演算器の性能をあげることなしにスループットの向上が可能となる．

またスループットの向上を期待しない場合，CP を短縮することで，低消費電力での実現が可能となる．換言すると，CP の短縮は各演算器での処理時間を緩和することを可能とする．この結果，回路の動作周波数を低くすることができ，回路の消費電力が低減される [1]．

2.2 クリティカルパスの短縮

遅延器（delay）を挿入することで，CP の長さを短縮ができることを示す．今，図 1 の回路に適切に遅延

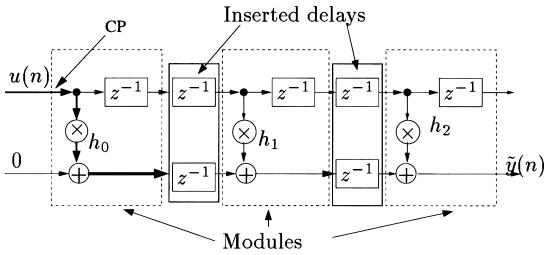

図2　3タップ FIR フィルタのパイプライン化
Fig. 2　Example of pipelining of a 3-tap FIR filter.

器を挿入することで，図 2 の回路を得る．なお，説明のため図 1 の回路に少し変更を加えてある．

図 2 の回路の CP の長さは，

$$\tau_{\rm cp} = \tau_{\rm m} + \tau_{\rm a} \tag{4}$$

となり，図 1 に比べ短縮されていることがわかる．このように，回路を細分化し CP を短縮する手法をパイプライン化と呼び，このとき用いられる遅延器のことをパイプラインラッチと呼ぶ．

この場合のパイプライン化された回路は，図 2 に示したように，モジュール（module）と呼ばれる基本演算単位間に，遅延器を挿入した構成となる．各モジュールは同時に動作を行い，あるモジュールの出力は隣のモジュールへの入力となる．モジュールにもとづくパイプライン構成は，規則的な構成となるため，VLSI 化に適している．

次に，レイテンシー（latency）について述べる．時刻 n における $u(n)$ が入力され，その信号に対応する信号 $y(n)$ が出力されるまでの時間差のことをレイテンシー（latency）と呼ぶ．今，図 2 と 図 1 の回路を比較することで，出力信号の関係が

$$\tilde{y}(n) = y(n-2) \tag{5}$$

となることがわかる．すなわち，図 2 の回路は，図 1 の回路に比べ短い CP をもつが，レイテンシーの値が 2 大きい．このように，パイプライン化のための遅延器の挿入は，レイテンシーを増大させる．一般に，実時間処理の要求では，レイテンシーはある値以下である必要がある．

パイプラインラッチは遅延器に限定されるものではない．しかし本論文で扱う，信号処理回路のアルゴリズムレベルでのパイプライン化では，パイプラインラッチを遅延器に限定して考えることが可能である．

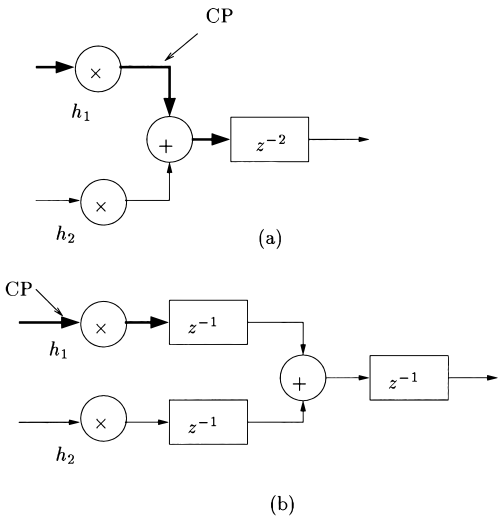

図3　リタイミング
Fig. 3　Example of applying retiming technique.

このため，本論文では特に断らない限り遅延器とパイプラインラッチを同一のものとして扱うものとする．

2.2.1　リタイミング手法

上の例では，パイプライン化のために元の回路には存在しない遅延器を挿入したが，回路中に既に存在する遅延器を配置しなおすことで CP を短縮することも可能である．この手法はリタイミング（retiming）と呼ばれる [1]．リタイミングの一例を図 3に示す．

リタイミングを用いて，同図 (a) を (b) の構成に整理することにより，CP が短縮される．リタイミングは，遅延器を移動する際に回路の入出力関係が変わらないように実行しなければならない．また，リタイミングはレイテンシーを増加させないが，遅延器数を増加させることが多い．

2.3　フィードバック経路がある回路

ADF のようにフィードバック経路がある回路をパイプライン化する場合には，次のような注意が必要である．

今，例として次式で与えられる 1 次のフィードバック回路を考えよう．

$$y(n) = ay(n-1) + x(n) \tag{6}$$

式 (6) のシグナルフローグラフ（signal flow graph: SFG）を 図 4 (a) に示す．この回路の CP の長さは，

$$\tau_{\rm cp} = \tau_{\rm m} + \tau_{\rm a} \tag{7}$$

である．

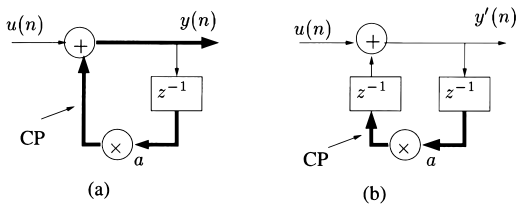

図 4　フィードバックのある回路のパイプライン化
Fig. 4　Pipelining of a circuit containing a feed-back path.

図 4 (a) の回路に遅延器を挿入し，CP の短縮を考えると，図 4 (b) に示した SFG が得られる．この回路の CP は $\tau'_{\rm cp} = \tau_{\rm m}$ となり CP の長さは短縮されている．しかしながら，図 4 (b) の入出力関係を求めると，

$$y'(n) = ay'(n-2) + x(n) \tag{8}$$

となる．式 (6) と式 (8) の比較から図 4 (a) と (b) の回路では，入出力関係が異なったものとなってしまうことがわかる．

この例から，フィードバック経路のある回路では，遅延器の挿入により入出力関係が変化することがわかる．このことが LMS ADF をパイプライン化する際に問題となる．なお，フィードバック経路がある回路にも，リタイミング手法を応用することは可能である．

3. DLMS 適応フィルタのパイプライン実現

本章では，GADF のパイプライン実現を考えよう．現在までに提案されているパイプライン実現の多くは，DLMS（delayed LMS）ADF にもとづいている．この理由は，LMS ADF [14] では，フィルタの係数が再帰的に計算されるため SFG にフィードバックが存在し，**2. 2** で述べた手法を直接適用することはできないためである．

DLMS は，リタイミング手法に使用可能な遅延器が存在するという特徴がある．この特徴を利用し，DLMS ADF のパイプライン化が実現される．

なお，GADF のパイプライン化には PIPLMS と呼ばれる手法も提案されている [1], [25]．しかし，PIPLMS は DLMS に比べ，収束特性が悪くハードウェア規模も大きいなどの問題があるため，現在のところ応用例はあまり見られない．このため，本論文では PIPLMS の説明は割愛する．

本章以下で使用する変数は以下のとおりである．時刻 n における，適応フィルタの k 番目の係数を $w_k(n)$ で表し，そのベクトルを $\boldsymbol{w}(n)$ とし，

$$\boldsymbol{w}(n) = [w_0(n)\ w_1(n)\ \ldots\ w_{N-1}(n)]^T \tag{9}$$

と定義する．ここで，N は $\boldsymbol{w}(n)$ のタップ数を，上付きの ^{T} は転置を表す．また，$u(n)$ 及び $d(n)$ は時刻 n における，$\boldsymbol{w}(n)$ の入力及び所望信号を表し，入力信号ベクトル $\boldsymbol{u}(n)$ を

$$\boldsymbol{u}(n) = [u(n)\ u(n-1)\ \ldots\ u(n-N+1)]^T \tag{10}$$

と定義する．

3. 1　DLMS アルゴリズム

まず DLMS アルゴリズムについて説明する．DLMS アルゴリズムの係数更新式は

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n-D)\boldsymbol{u}(n-D) \tag{11}$$

と与えられる [15], [16]．ここで D は整数であり，誤差信号 $e(n-D)$ は

$$e(n-D) = d(n-D) - \boldsymbol{u}^T(n-D)\boldsymbol{w}(n-D) \tag{12}$$

と定義される．また，μ はステップサイズパラメータである．なお $\boldsymbol{w}(n)$ の出力 $y(n)$ を

$$y(n) = \boldsymbol{u}^T(n)\boldsymbol{w}(n) \tag{13}$$

とベクトル表記する．一方，LMS アルゴリズムの係数更新式は，良く知られているように

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n)\boldsymbol{u}(n) \tag{14}$$

$$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n) \tag{15}$$

と与えられる [14].

式 (14) と式 (11) よりわかるように，DLMS ADF は LMS ADF の誤差信号フィードバック経路に D 個の遅延器を挿入することに相当し，図 5 に示したブロック図で表される．DLMS ADF のパイプライン化は，この挿入された D 個の遅延器をリタイミングすることで実現される．

3. 2　DLMS ADF のアーキテクチャ

次に，DLMS ADF のアーキテクチャ（構成）を示す．まず，図 6 にパイプライン化する前の DLMS ADF の SFG を示す．この図は，$\boldsymbol{w}(n)$ が 3 タップの場合の例である．

図で z^{-D} で示した遅延器をリタイミングすることで，パイプライン実現が可能となる．このとき D の値が大きいほど，CP をより短縮することが可能となる．D の選択可能な範囲は，

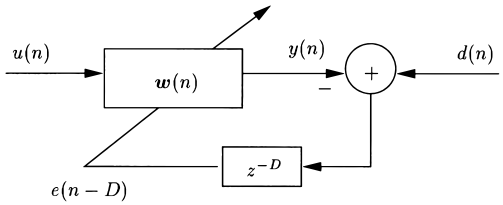

図 5　DLMS による適応フィルタ
Fig. 5　Adaptive filter driven by DLMS algorithm.

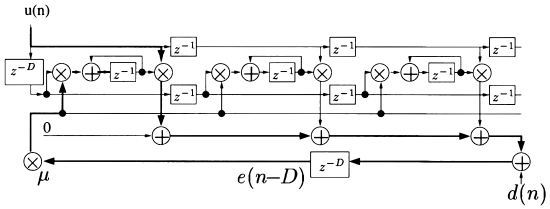

図 6　3 タップ DLMS 適応フィルタ．太線は $D = 0$ のときのクリティカルパスを示す
Fig. 6　SFG of a 3-tap DLMS adaptive filter. Bold line indicates the critical path of the LMS adaptive filter.

$$0 \leqq D \leqq N \tag{16}$$

となり，$D = N$ と選択した場合に最も CP を短縮できる．$D = N$ と選択した場合には，適応フィルタ $\boldsymbol{w}(n)$ の 1 タップを一つのモジュールとしてパイプライン実現を行うことになる．また，$D < N$ の場合には，複数タップを単位としてモジュールが構成される [26].

$D = N$ と選択した場合の $i\,(i = 0, 1, \ldots, D-1)$ 番目のモジュールの構成例を図 7 に，フィルタ全体の構成例を図 8 にそれぞれ示す．図 7 で $y_i(n)$ は，出力信号 $y(n)$ の部分和を表しており，

$$y_i(n-i) = y_{i-1}(n-i) + w_i(n-1-i)u(n-2i) \tag{17}$$

と与えられる．

ここで，CP の長さの比較を行う．なお，本論文では CP の計算に [21] で提案された仮定を使用する．すなわち，μ の値を 2 のベキ乗に選ぶことで，掛算をシフト演算で代用できることを利用し，μ と $e(n)$ との掛算を無視できるものとする [21].

まず，LMS ADF の CP は $D = 0$ としたときの，図 6 の太線で示した経路であり，

$$\tau_{\mathrm{cp,lms}} = 2\tau_{\mathrm{m}} + (N+2)\tau_{\mathrm{a}} \tag{18}$$

となる．上式からわかるように，LMS ADF の CP の

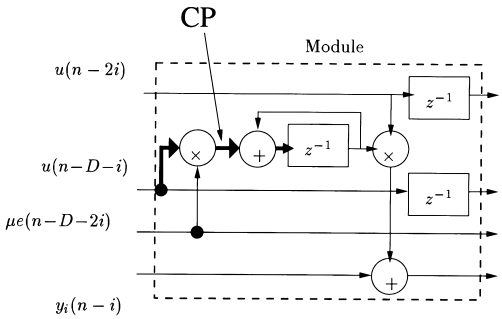

図 7　DLMS のパイプライン実現における i 番目のモジュールの構成
Fig. 7　Structure of the i-th module of pipelined adaptive filter based on the DLMS.

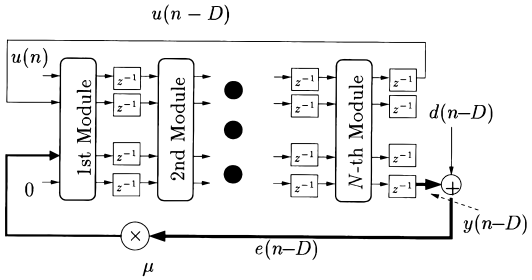

図 8　3 タップ DLMS 適応フィルタ．太線はクリティカルパスを示す
Fig. 8　SFG of a 3-tap DLMS adaptive filter. Bold line indicates the critical path.

長さは，フィルタの長さ N に比例する．次にパイプライン化された DLMS ADF では，

$$\tau_{\mathrm{cp,dlms}} = \tau_{\mathrm{m}} + 2\tau_{\mathrm{a}} \tag{19}$$

となり，N には依存しない一定値となる．フィルタ長 N が長くなるほど，これら二つの CP の長さの差が大きくなり，パイプライン化が有効であるといえる．

なお，モジュールやフィルタ全体の構成には自由度があり，図に示した物は一例であることに注意されたい．

3.3　DLMS の問題点

パイプライン化された DLMS ADF の CP 及び構成は，挿入する遅延器の量 D により決定されることを述べた．すなわち，D を多くすることで，CP の短縮が行えスループットの向上が可能となる．しかし，D の増加は μ の選択範囲を狭くし，ADF の収束速度を悪化させる [15]．同時に，レイテンシーの大きさが D 増加する．以下では，この問題の改善が主題となる．

4. 収束特性の改善

ここでは，DLMS ADF の収束特性の問題を改善するパイプライン構成法について述べる．

4.1 DLMS の LMS への等価変換

先に見たように，式 (11) の $\boldsymbol{w}(n)$ と式 (12) の $\boldsymbol{w}(n-D)$ との時間差 D が，収束性の悪化などの問題を生じさせる．この時間差 D を短縮することで，収束特性の改善が可能である．

しかし，遅延器はパイプライン実現のために必要であるため，遅延器の数を減らさずに時間差だけを短縮することが必要となる．このことは，式 (12) で定義される $e(n-D)$ の代わりに，遅延量を補正した誤差

$$\epsilon(n-D) \triangleq d(n-D) - \boldsymbol{u}^T(n-D)\boldsymbol{w}(n) \qquad (20)$$

を利用できれば実現可能である． $\epsilon(n-D)$ の実現方法を以下に述べる．なお，このような DLMS アルゴリズムの補正の考え方は，パイプライン実現とは独立に [27] で示されている．

4.1.1 ルックアヘッド変換

今，LMS の更新式 (14) の，時刻 n における関係を求めると

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \mu e(n-1)\boldsymbol{u}(n-1) \qquad (21)$$

となる．式 (14) に式 (21) を代入すると，

$$\begin{aligned}\boldsymbol{w}(n+1) &= \boldsymbol{w}(n-1) + \mu e(n-1)\boldsymbol{u}(n-1) \\ &\quad + \mu e(n)\boldsymbol{u}(n)\end{aligned} \qquad (22)$$

を得る．この操作を D 回繰り返すことで，

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n-D) + \sum_{i=0}^{D} \mu e(n-i)\boldsymbol{u}(n-i) \qquad (23)$$

を得る．式 (14) から式 (23) を得る操作のことを D 段のルックアヘッド変換（look-ahead transform）と呼ぶ．

式 (23) の関係から

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-D) + \sum_{i=1}^{D} \mu e(n-i)\boldsymbol{u}(n-i) \qquad (24)$$

であることがわかる．この結果，

$$\begin{aligned}&\boldsymbol{u}^T(n-D)\boldsymbol{w}(n) \\ &\quad = \boldsymbol{u}^T(n-D)\boldsymbol{w}(n-D) \\ &\qquad + \boldsymbol{u}^T(n-D)\sum_{i=1}^{D} \mu e(n-i)\boldsymbol{u}(n-i) \\ &\quad = y(n-D) + \boldsymbol{u}^T(n-D)\sum_{i=1}^{D} \mu e(n-i)\boldsymbol{u}(n-i)\end{aligned} \qquad (25)$$

と表せる．

4.1.2 補正された誤差

式 (25) を利用することで，補正された誤差 $\epsilon(n-D)$ は

$$\begin{aligned}\epsilon(n-D) &= d(n-D) - \boldsymbol{u}^T(n-D)\boldsymbol{w}(n) \\ &= d(n-D) \\ &\quad - \boldsymbol{u}^T(n-D)\boldsymbol{w}(n-D) + \Lambda(n) \\ &= e(n-D) + \Lambda(n)\end{aligned} \qquad (26)$$

と表せる．ここで，$\Lambda(n)$ は

$$\Lambda(n) = -\boldsymbol{u}^T(n-D)\sum_{i=1}^{D} \mu e(n-i)\boldsymbol{u}(n-i) \qquad (27)$$

である．すなわち，DLMS の誤差信号 $e(n-D)$ に項 $\Lambda(n)$ を付加することで，$\boldsymbol{w}(n)$ の時間差 D を補正することが可能となる．このときのブロック図を図 9 に示す．

最後に，上記の変換を使用したときの適応フィルタ $\boldsymbol{w}(n)$ の更新式を示す．

$$\begin{aligned}\boldsymbol{w}(n+1) &= \boldsymbol{w}(n) + \mu\epsilon(n-D)\boldsymbol{u}(n-D) \\ &= \boldsymbol{w}(n) + \mu e(n-D)\boldsymbol{u}(n-D) \\ &\quad + \mu\Lambda(n)\boldsymbol{u}(n-D)\end{aligned} \qquad (28)$$

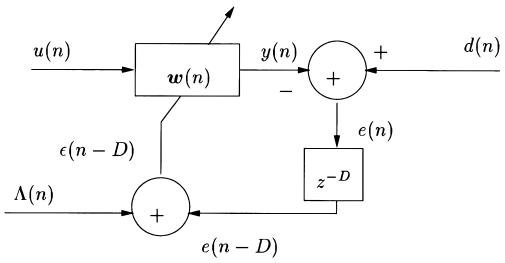

図 9 付加項 $\Lambda(n)$ による DLMS の収束特性の改善
Fig. 9 Modified structure of DLMS with additional term $\Lambda(n)$ for improving the convergence characteristics.

このアルゴリズムのことを，[11] では LDLMS（lookahead DLMS）と呼んでいる．

この変換を用いることで，挿入する遅延器の量 D を変化させることなく，$\boldsymbol{w}(n)$ の時間差だけを減少させることが可能となる．この結果，D 段のルックアヘッドを行うことで，LMS と同等の収束特性を実現できる．しかしながら，式 (28) のアルゴリズムは DLMS に基づいているため，レイテンシーの増加量は D のままである．また **5. 3** に示すように，$\Lambda(n)$ を実現するためには，必要となるハードウェアの量が大幅に増加するという新たな問題が生じる．

4. 2 アーキテクチャ

式 (28) のアルゴリズムのアーキテクチャは，付加項 $\Lambda(n)$ の実現の仕方により二つに分類することができる．すなわち，(i) DLMS 部分とは独立に $\Lambda(n)$ を実現するものと，(ii) DLMS のアーキテクチャの各モジュールに $\Lambda(n)$ を含めるものである．

松原らにより提案された方法 [11], [20] は，DLMS 部分と $\Lambda(n)$ 部分とを分けて構成するアーキテクチャである．すなわち，フィルタ部分及び係数更新部分は，従来の DLMS の構成をそのまま利用し，$\Lambda(n)$ を計算する部分を付加する形となる．一方，モジュールの内部に $\Lambda(n)$ を含める構成は，Douglas らによって提案された [12], [21].

CP の長さは，どちらの構成も DLMS のもの式 (19) と同じ長さになる．Douglas の構成では，再帰的に $\Lambda(n)$ の計算を行っているため，実現の際に有限語長演算の影響が現れると考えられる．また，松原の構成では $\Lambda(n)$ の計算をそのまま実現しているため，Douglas の構成よりも多くの演算器が必要となる．

5. レイテンシーの改善及び演算量の削減

前章では，DLMS ADF の収束特性の改善は可能であるが，レイテンシーの増加及び演算量の増大の問題が残ることを見た．ここでは，これらの問題を改善する構成について述べる．

5. 1 LMS にもとづくパイプライン構成

3. 1 で述べたように，LMS にはリタイミングに必要なディレーが存在しないため，パイプライン実現は不可能であると考えられてきた．これに対し，LMS に対して等価変換を行うことで，LMS アルゴリズムに基づくアーキテクチャが，実現可能であることが示された [22], [28].

すなわち，LMS アルゴリズムを等価変換することで，LMS ADF に元々存在する遅延器を利用し，パイプライン化が実現可能となる [28]．この手法は，遅延器を新たに挿入する必要がないため，LMS ADF と等価な特性を得ることができる．

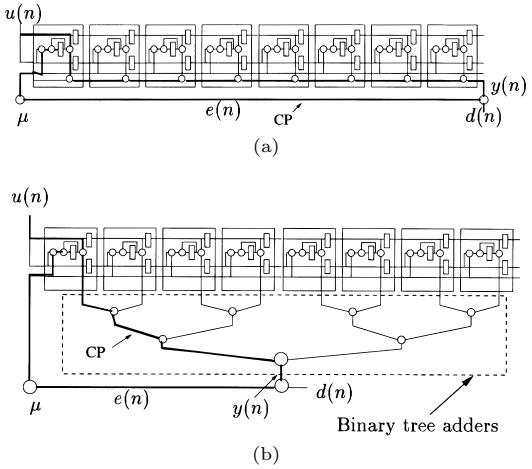

図 10 バイナリツリー構成による CP の短縮（$N = 8$ の場合）．太線は CP を表す
Fig. 10 Shortening of CP using binary-tree structure for the calculation path of $y(n)$.

この等価変換にもとづくアーキテクチャは 2 種類に分類される．すなわち，(i) 収束特性及びレイテンシー共に LMS ADF と完全に等価な特性を実現するものと，(ii) 収束特性は LMS ADF と等価であるが，レイテンシーが 1 増加するものである．構成 (i) では，CP の長さが $2\tau_{\mathrm{m}} + 2\tau_{\mathrm{a}}$ となり，式 (19) に与えられる DLMS の CP よりも増加するが，構成 (ii) では CP の長さが式 (19) に等しくなる．

しかし，LMS アルゴリズムの等価変換を実現するために必要となるハードウェア量が，[20], [21] と同程度となり，演算量の多さの問題は残されたままとなる．

5. 2 ハードウェア量の少ない構成

多少のレイテンシーの増加を許すことにより，少ないハードウェア量の増加で LMS ADF と同等の収束特性を実現しパイプライン化された DLMS ADF と同一の CP を実現するアーキテクチャを示す [23].

ここでは，図 7 のモジュールを遅延器を挟まずに接続した図 10 (a) の回路をもとにして考える．この回路での CP は，図の太線で示した出力 $y(n)$ を計算する経路であり，その長さは式 (18) の $\tau_{\mathrm{cp,lms}}$ で与えられる．なお図は $N = 8$ の場合を示している．

DLMS とは異なる方法を用いることで，図 10 (a)

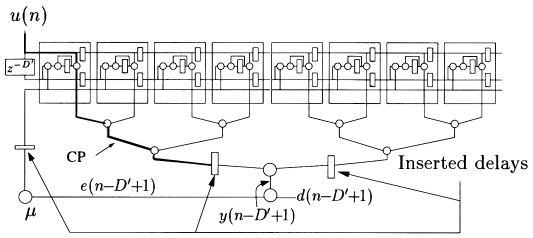

図 11　遅延器の挿入による CP の短縮（$N = 8$ の場合）
Fig. 11　Further shortening of CP by inserting delays.

の回路の CP の長さを，式 (19) の $\tau_{\mathrm{cp,dlms}}$ まで短縮できることを述べる．まず第 1 段階として，$y(n)$ の計算を行う経路を，図 10 (b) に示したように，バイナリツリー構成にすることを考える．この結果，CP の長さは，

$$\tau_{\mathrm{cp}} = (\log_2 N + 2)\tau_{\mathrm{a}} + 3\tau_{\mathrm{m}} \tag{29}$$

となり，図 10 (a) の $\tau_{\mathrm{cp,lms}}$ よりも短縮できる．しかし式 (29) は，いぜん式 (19) で与えられる DLMS の CP よりも長い．

次に第 2 段階として，図 10 (b) の回路のバイナリツリーの部分に，遅延器を挿入することを考える．このとき，遅延器を挿入する間隔は CP の長さが

$$\tau_{\mathrm{cp}} \leqq \tau_{\mathrm{cp,dlms}} = 2\tau_{\mathrm{a}} + \tau_{\mathrm{m}} \tag{30}$$

となるように選択する．この条件は，CP 上の演算器 3 個に 1 個の割合で遅延器を挿入することで満足することができる．図 10 (b) の回路に遅延器を挿入した様子を図 11 に示す．

遅延器を挿入した結果，$\boldsymbol{w}(n)$ の更新式は，DLMS ADF の更新式 (11) 及び式 (12) において $D = D'$ と置いたものと一致する．ここで，D' は

$$D' = \left\lceil \frac{\log_2 N}{3} \right\rceil + 1 \tag{31}$$

で与えられ，更にここで式 $\lceil x \rceil$ は x 以上の最小の整数を表す．この D' の時間差は，DLMS ADF で D' の遅延を挿入したことに相当し，収束特性を悪化させ，更にレイテンシーを増加させる．時間差 D' の影響は，幸い **4.** で述べた手法を適用することで DLMS ADF と同様改善が可能である．

図 11 の構成が，**4.** の構成と異なる点は，補正が必要となる時間差の大きさである．このことを示すために，タップ数 N に対する D と D' の比較を，表 1 に示す．表より，N が増加するにつれ D' と D との差が大きくなることがわかる．このため，付加項 $\Lambda(n)$ の計算をするために必要となる演算器の数を大幅に縮小可能となる．また，増加するレイテンシーの量も D' となる．

表 1　D と D' の比較
Table 1　Comparison between D and D'.

タップ数 N	D	D'
2	2	2
64	64	3
1024	1024	5

5.3 比　　較

最後に，ここまでに述べてきたアーキテクチャの比較を行う．表 2 に，各アーキテクチャの CP の長さ，レイテンシー増加量及び必要となる演算器数の比較を示す．

表より以下のことがわかる．CP に関しては，LMS (i) をのぞきすべて一致する．また，収束特性に関しては，DLMS 以外は LMS と同等となるが，レイテンシーの増加は LMS (i) 及び (ii) が最も少ない．必要となる演算量に関しては，**5.2** に述べた構成が最も少ない．

なお，パイプライン化されたアーキテクチャの比較の観点としては，これ以外にもモジュール構成の可否や，ブロードキャストラインの有無なども重要である．

6. むすび

本論文では，こう配形適応フィルタのパイプライン実現について述べた．DLMS ADF を元にパイプライン実現を行うことで，フィルタの CP の長さを短縮でき，スループットの向上が可能となることを示した．また，収束特性の悪化やレイテンシーの増加といった，DLMS ADF のパイプライン化の際の問題を，少ないハードウェア量の増加で改善する構成を示した．

なお，本論文で述べた手法はすべて 1 次元 GADF を対象としたものであった．これらの手法の 2 次元 GADF への拡張が [29], [30] で述べられている．

文　　献

[1] N.R. Shanbhag and K.K. Parhi, "Pipelined Adaptive Digital Filters," Kluwer Academic Publishers, Massachusetts, 1994.
[2] N.R. Shanbhag and M.Goel, "Low-power adaptive filter architectures and their application to 51.84 Mb/s ATM-LAN," IEEE Trans. Signal Processing, vol.45,

表2　各アーキテクチャの比較（タップ数 N）
Table 2　Comparison of pipelined architectures of the GADF (N indicates the number of taps).

アーキテクチャ	クリティカルパスの長さ	レイテンシーの増加量	必要となる演算器数		
			乗算器	加算器	遅延器
DLMS [16]	$\tau_m + 2\tau_a$	N	$2N+1$	$2N+1$	$8N-2$
LDLMS [20]	$\tau_m + 2\tau_a$	N	$4N-5$	N^2+2	$N^2+10N-4$
文献 [21]	$\tau_m + 2\tau_a$	N	$5N+1$	$5N+1$	$6N$
LMS[28] (i)	$2\tau_m + 2\tau_a$	0	$5N-2$	$5N-2$	$5N-2$
LMS[28] (ii)	$\tau_m + 2\tau_a$	1	$6N-1$	$6N-2$	$7N-1$
5.2 節の構成 [23]	$\tau_m + 2\tau_a$	$D'-1$	$2N+3D'$	$2N+3D'+1$	$3N+(3+D')D'-1+\sum_{i=0}^{D'-1}\lceil \frac{N}{2^{3i-1}} \rceil$

no.5, pp.1276–1290, May 1997.

[3] 酒井英昭，田中耕司，“適応フィルタと VLSI 実現,” システム/制御/情報，vol.38, no.8, pp.411–416, Aug. 1994.

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

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

[6] K.J. Raghumath and K.K. Parhi, “Pipelined RLS adaptive filtering using scaled tangent rotations (STAR),” IEEE Trans. Signal Processing, vol.44, no.10, pp.2591–2604, Oct. 1996.

[7] K.J. Raghumath and K.K. Parhi, “Finite-precision error analysis of QRD-RLS and STAR-RLS adaptive filters,” IEEE Trans. Signal Processing, vol.45, no.5, pp.1193–1209, May 1997.

[8] H. Sakai, “A vectorized systolic array for parallel weight extraction of block RLS,” International Journal of Adaptive Control and Signal Processing, vol.8, pp.475–482, 1994.

[9] H. Sakai, “Recursive lease-squares algorithms of modified Gram-Schmidt type for parallel weight extraction,” IEEE Trans. Signal Processing, vol.SP-42, no.2, pp.429–433, Feb. 1994.

[10] S.Y. Kung, “VLSI Array Processing,” Prentice Hall, Englewood Cliffs, NJ 07632, 1988.

[11] K. Matsubara, K. Nishikawa, and H. Kiya, “Pipelined LMS adaptive filters using a new look-ahead transformation,” IEEE Trans. Circuits & Syst. II, to be appeared.

[12] Q. Zhu, S.C. Douglas, and K.F. Smith, “A pipelined architecture for LMS adaptive FIR filter without adaptation delay,” Proc. IEEE ICASSP’97, Munich, pp.1933–1936, 1997.

[13] 吉田智焼，飯国洋二，前田　肇，“LMS アルゴリズムのシストリックアレー実現に関する一考察,” 第 11 回ディジタル信号処理シンポジウム予稿集，pp.475–480, Nov. 1996.

[14] B. Widrow and S.D. Stearns, “Adaptive Signal Processing,” Prentice Hall, Englewood Cliffs, NJ 07632, 1985.

[15] G. Long, F. Ling, and J.G. Proakis, “The LMS algorithm with delayed coefficient adaptation,” IEEE Trans. Accoust., Speech & Signal Process., vol.37, no.9, pp.1397–1405, Sept. 1989.

[16] R.H. Cohen, H. Herzberg, and Y. Be’ery, “Delayed adaptive LMS filtering: current results,” Proc. IEEE ICASSP’90, pp.1273–1276, 1990.

[17] H. Herzberg, R. Haimi-Cohen, and Y. Be’ery, “A systolic array realization of an LMS adaptive filter and the effects of delayed adaptation,” IEEE Trans. Signal Processing, vol.40, no.11, pp.2799–2803, Nov. 1992.

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

[19] J. Thomas, “Pipelined systolic architectures for DLMS adaptvie filtering,” Journal of VLSI Signal Processing, vol.12, pp.223–246, 1996.

[20] 松原勝重，西川清史，貴家仁志，“ルックアヘッド 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.J80-A, no.11, pp.1902–1909, Nov. 1997.

[21] S.C. Douglas, Q. Zhu, and K.F. Smith, “A pipelined architecture for LMS adaptive FIR filter architecture without adaptation delay,” IEEE Trans. Signal Processing, vol.46, no.3, pp.775–779, March 1998.

[22] 原田昭男，西川清史，貴家仁志，“最小レイテンシーを持つパイプライン LMS 適応フィルタ,” DSP シンポジウム，pp.609–614, Nov. 1997.

[23] T. Kimijima, K. Nishikawa, and H. Kiya, “A pipelined architecture for DLMS algorithm considering both hardware complexity and output latency,” Proc. EUSIPCO-98, Island of Rhodes, Greece, 1998.

[24] P. Pirsch, “Architectures for Digital Signal Processing,” Wiley, New York, NY 10158-0012, 1998.

[25] N.R. Shanbhag and K.K. Parhi: “Relaxed look-ahead pipelined LMS adaptive filters and their application to ADPCM coder,” IEEE Trans. Circuits & Syst. Part II, vol.40, no.12, pp.753–766, Dec. 1993.

[26] 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.
[27] R.D. Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm," IEEE Signal Proc. Letters, vol.2, no.12, p.223, Dec. 1995.
[28] A. Harada, K. Nishikawa, and H. Kiya, "Pipelined architecture of the LMS adaptive digital filter with the minimm output latency," IEICE Trans., vol.E81-A, no.8, pp.1578–1585, Aug. 1998.
[29] K. Matsubara, K. Nishikawa, and H. Kiya, "2-d pipelined adaptive filters based on 2-d delayed lms algorithm," IEICE Trans. Fundamental, vol.80, no.6, pp.1009–1014, June 1997.
[30] T. Kimijima, K. Nishikawa, and H. Kiya, "Pipelining of 2-dimensional adaptive filters based on the LDLMS algorithm," Proc. IEEE ISCAS'98, Monterey, pp.V190–V193, May 1998.

（平成 10 年 9 月 9 日受付）

西川　清史　（正員）

1990 都立大・工・電気工卒．1992 同大大学院修士課程了．同年新日本製鉄エレクトロニクス研究所勤務．1993 都立大・工・電子・情報工学科助手．現在，同大学大学院工学研究科電気工学専攻助手．工博．適応信号処理及び信号処理の VLSI 実現に関する研究に従事．IEEE 会員．

貴家　仁志　（正員）

1980 長岡技科大・工・電気電子システム卒．1982 同大大学院修士課程了．同年都立大・工・電気工学科助手．現在，同大学大学院工学研究科電気工学専攻助教授．1995 年 10 月〜1996 年 3 月シドニー大（オーストラリア）客員研究員．工博．マルチレート信号処理，適応信号処理及び画像処理に関する研究に従事．IEEE Transaction on Signal Processing，本会和文論文誌（A）編集委員．著書「高速フーリエ変換とその応用」,「ディジタル信号処理技術」,「マルチレート信号処理」．電子画像学会，テレビジョン学会，IEEE 各会員．