論　文

# Delayed LMS アルゴリズムに基づくパイプライン適応フィルタ

松原　勝重[†]　　西川　清史[†]　　貴家　仁志[†]

# Pipelined Adaptive Filters Based on Delayed LMS Algorithm

Katsushige MATSUBARA[†], Kiyoshi NISHIKAWA[†], and Hitoshi KIYA[†]

**あらまし**　パイプライン化が可能なアルゴリズムとして Delayed Least Mean Square (DLMS) アルゴリズムが知られている．DLMS は Pipelined LMS (PIPLMS) の1特殊形とみなすことができ，PIPLMS に比べ速い収束速度をもつ．従来の DLMS に基づくパイプラインアーキテクチャは，フィルタ次数とは独立に入力信号のクロックレートを一定に保つために，フィルタの1タップを一つのセルとしている．しかしながら，フィルタ次数の増大と共に収束特性の劣化やレイテンシーの増大，および追従性の悪化を生じる．本論文では一つのセルに複数のタップを含ませるアーキテクチャを提案している．提案法により，収束特性およびレイテンシーの改善が可能であり，計算機シミュレーションによりその有効性を確認している．またセル内の演算に対しスケジューリングを行うことで，複数タップをまとめたことによる信号のクロックレートの低下を抑制でき，従来法に比べ約半数のハードウェア量で実現できることを示している．

**キーワード**　パイプライン処理，PIPLMS アルゴリズム，DLMS アルゴリズム

## 1. まえがき

本文では Least Mean Square (LMS) アルゴリズム [1] に基づくパイプライン適応フィルタのパイプライン処理の収束特性を，従来法に比べ改善できるアーキテクチャを示される．

LMS アルゴリズムは実現の容易さから適応信号処理に広く用いられている．近年この LMS に基づくパイプライン処理が研究されており，いくつかのアルゴリズムや構成法が提案されている [2], [4]～[6]．そのうち Pipelined LMS (PIPLMS) アルゴリズム [2] は LMS をパイプライン処理する一般的なアルゴリズムとしてよく知られている．このアルゴリズムは，再帰型アルゴリズムのパイプライン実現のための一方法であるルックアヘッド [8] に，近似 (relaxation) を施した近似ルックアヘッド (relaxed look-ahead) [2] に基づいている．近似は後に示す3の方法が代表的であり，それぞれ信号の統計的性質を利用して行われる．また近似の度合はパラメータにより調整可能であるが，ハードウェアとアルゴリズムの特性とのトレードオフにより決定しなければならない．すなわち近似の度合を大きくすることで入力信号のクロックレートの向上が可能である反面，同時に収束特性は悪くなることが知られている [2]．

一方 PIPLMS とは独立に，パイプライン化を可能とするアルゴリズムとして Delayed LMS (DLMS) が知られている [3], [4]．DLMS は適応エコーキャンセラの分野で従来から知られて，LMS の係数更新部にディレイを挿入することに相当する．LMS の係数更新部にディレイを挿入することで，DLMS は PIPLMS の近似の一つである遅延近似 (delay relaxation) を用いた特殊形となる [2]．DLMS は PIPLMS に比べ近似が少ない分，収束特性は良い．しかしディレイを大きくすることは近似の度合を大きくすることと等価であり，PIPLMS と同様に入力信号のクロックレートと収束特性のトレードオフが生じてしまう．

DLMS に基づくパイプライン処理ののハードウェアアーキテクチャが報告されている [5], [6]．従来法は適応フィルタの1タップを1機能ブロック (セル) としてパイプライン処理を行っている．このため入力信号のクロックレートはタップ数に依存せず高い値をもつことができる．しかし，ディレイはフィルタのタップ数と等しくなり，フィルタが高次のとき収束特性の劣化が生じる．更にディレイの増加はレイテンシーの増加やアルゴリズムの追従性の劣化も引き起こしてし

† 東京都立大学工学部電子情報工学科，八王子市
Faculty of Technology, Tokyo Metropolitan University, Hachioji-shi, 192-03 Japan

まう．

本文では，従来法の問題点を回避するためのDLMSに基づく新しいアーキテクチャを提案する．提案法は従来法とは異なり，複数のタップを一つのセルとしてパイプライン処理を行う．この構成によりディレイを減少でき，収束特性およびレイテンシーが改善できることを示す．複数タップをまとめたことによるクロックレートの劣化も，セルに含まれる演算器のスケジューリングを行うことで抑制できることを示す．その際，DLMSに基づく従来の構成法[5],[6]に比べハードウェア量を約半数に減少できることを示す．

## 2. パイプライン LMS アルゴリズム

ここではパイプライン処理が可能なLMS型アルゴリズムについて述べる．はじめに近似ルックアヘッドに基づくPIPLMSを，次にDLMSについて述べる．

### 2.1 準　　備

LMSの更新式は以下の式で表せる[1]．

$$\boldsymbol{W}(n)=\boldsymbol{W}(n-1)+\mu e(n)\boldsymbol{U}(n) \tag{1}$$

$$e(n)=d(n)-\boldsymbol{W}^T(n-1)\boldsymbol{U}(n) \tag{2}$$

但し $\boldsymbol{W}(n)$，$\boldsymbol{U}(n)$ はそれぞれ適応フィルタ係数ベクトル，入力ベクトルで

$$\boldsymbol{W}(n)=[w_0(n),w_1(n),\cdots,w_{N-1}(n)]^T \tag{3}$$

$$\boldsymbol{U}(n)=[u(n),u(n-1),\cdots,u(n-N+1)]^T \tag{4}$$

である．また $[\cdot]^T$ は行列の転置，$\mu$ はステップサイズ，$d(n)$ は所望信号，$N$ はフィルタのタップ数を表す．LMSの更新式は式(1)からわかるように，再帰的に係数$\boldsymbol{W}(n)$を更新する．そのため式(1)に従来の非再帰アルゴリズムをパイプライン化する方法[7]を適用し，パイプライン化を行うことは不可能である．

このような再帰アルゴリズムをパイプライン化する方法として，ルックアヘッドが知られている[8]．ルックアヘッドは，再帰式を何段か展開することでパイプライン処理を可能とする．このとき，展開によってもとの式の入出力関係を変えないことに注意する．ここで式(1)のLMS更新式に $M$ 段のルックアヘッドを行うと

$$\boldsymbol{W}(n)=\boldsymbol{W}(n-M)+\mu\sum_{i=0}^{M-1}e(n-i)\boldsymbol{U}(n-i) \tag{5}$$

となる．但し $M$ は1以上の任意の整数である．式(5)は式(1)と等価な入出力関係を与える．しかし式(5)ではハードウェア実現時に，第2項目が大きなハードウェア量の増加を引き起こしてしまう．

この問題を回避するため式(5)に，信号の統計的性質に基づく近似（relaxation）を適用し，部分的なルックアヘッド（relaxed look-ahead）を行うことでLMSをパイプライン化するアルゴリズムが提案されている[2]．近似の方法としては遅延近似（delay relaxation），加算近似（sum relaxation）および乗算近似（product relaxation）が代表的である．ここでは後の議論で必要となる遅延近似と加算近似について述べる．

### 2.2 遅延近似

まず遅延近似について述べる．遅延近似は，係数更新に一定量のディレイ $D_1$ を受けた信号 $e(n-D_1)$，$\boldsymbol{U}(n-D_1)$ を用いる[4]．式(5)に遅延近似を用いると

$$\boldsymbol{W}(n)=\boldsymbol{W}(n-M)+\mu\sum_{i=0}^{M-1}e(n-D_1-i)\boldsymbol{U}(n-D_1-i) \tag{6}$$

となる．この近似が有効であるためには，$e(n)\boldsymbol{U}(n)$ が定常若しくは変化が緩やかでなければならない．

遅延近似により，ルックアヘッドでは行えないフィルタ部のパイプライン化が可能となる[4]．

### 2.3 加算近似

次に加算近似について述べる．まずLAを

$$1\leqq LA\leqq M \tag{7}$$

の定数とし，式(5)を次のように近似する[2]．

$$\boldsymbol{W}(n)=\boldsymbol{W}(n-M)+\mu\frac{M}{LA}\sum_{i=0}^{LA-1}e(n-i)\boldsymbol{U}(n-i) \tag{8}$$

この近似が有効であるためには，$e(n)\boldsymbol{U}(n)$ が $LA$ サンプル期間においてほぼ一定でなければならない．また $LA$ は近似の度合を表すパラメータである．すなわち $LA=1$ で最も大まかな近似を行い，$LA=M$ では全く近似を行わず式(8)は式(5)と一致する．

### 2.4 PIPLMS アルゴリズム

PIPLMSは式(1)に遅延近似および加算近似を施す

ことで導かれ，その更新式は以下の式で表せる[2].

$$\boldsymbol{W}(n) = \boldsymbol{W}(n-D_2) + \mu \sum_{i=0}^{LA-1} e(n-D_1-i)\boldsymbol{U}(n-D_1-i) \tag{9}$$

$$e(n) = d(n) - \boldsymbol{W}^T(n-D_2)\boldsymbol{U}(n) \tag{10}$$

ここで $D_2$ は式(5)の $M$ と同じくルックアヘッドを行う段数を表す．近似の度合を大きく行うほど，すなわち $D_1$ を大きく $LA$ を小さくするほど，細かいパイプライン化が可能となるが，収束特性は劣化することが知られている[2]．入力信号のクロックレートと，収束特性や追従性とのトレードオフが生じる．

### 2.5 DLMSアルゴリズム

DLMSは適応エコーキャンセラの分野において従来から知られていた[3],[4]．DLMSに基づくパイプラインのハードウェア実現法はいくつか報告されている[5],[6].

DLMSの更新式は以下の式で表せる[4].

$$\boldsymbol{W}(n) = \boldsymbol{W}(n-1) + \mu e(n-D_1)\boldsymbol{U}(n-D_1) \tag{11}$$

$$e(n) = d(n) - \boldsymbol{W}^T(n-1)\boldsymbol{U}(n) \tag{12}$$

DLMSはPIPLMSで $LA=1$，$D_2=1$ としたときと一致する．すなわちPIPLMSにおいて遅延近似のみを行うことと等価とみなすことができる．DLMSはPIPLMSより，高いクロックレートの入力信号は扱えないが，収束特性は良い．

### 2.6 評価基準

パイプライン処理の単位時間当りにおける処理量を評価する基準として，スループットが広く用いられている．しかしパイプライン化を必要とするディジタル信号処理において，単位時間に処理できる処理量を示すスループットよりも，扱う入力信号のクロックレートが応用上直接的であると考えられる．そこで入力信号のクロックレートを示すものとして速度係数を以下に定義し，本論文では評価基準として用いる．

$$S = \frac{1}{\tau} \tag{13}$$

ここで $\tau$ はパイプラインフィルタの1機能ブロック（セル）における最大処理遅延（クリティカルパス）である．

次にレイテンシーを以下に定義する．レイテンシーは対応する入出力間の時間差を表す．適応信号処理においてレイテンシーの増大は，非定常環境での追従性の劣化などを引き起こしてしまう．

パイプライン適応フィルタのレイテンシーを以下に定義する．

$$L = D_1\tau \tag{14}$$

ここで $D_1$ は遅延近似によるディレイである．

以下のフィルタの評価には式(13)，(14)で示される速度係数とレイテンシーを用いる．

## 3. パイプライン適応フィルタ

従来法に基づくパイプライン処理構成とその問題点を示し，従来法の拡張として提案法を示す．また提案法の有効性をシミュレーションを通じて確認する．

### 3.1 従来法による構成

従来法ではパイプライン処理において1タップを一つの機能ブロック（セル）とすることを提案している．この1タップ相当のセルをここではモジュールと呼ぶことにする．図1にモジュールのシグナルフローグラフを示す[6]．図1で **F**-block と **WUD**-block はそれぞれフィルタ部と係数更新部である．図より各々のモジュールにおけるクリティカルパスは，1組の積和演算にかかる時間となることがわかる．すなわち

$$\tau = \tau_{mlt} + \tau_{add} \tag{15}$$

である．但し $\tau_{mlt}$，$\tau_{add}$ は各々乗算および加算1回にかかる時間である．

$i$ 番目のモジュールにおける入出力関係と係数更新は，それぞれ

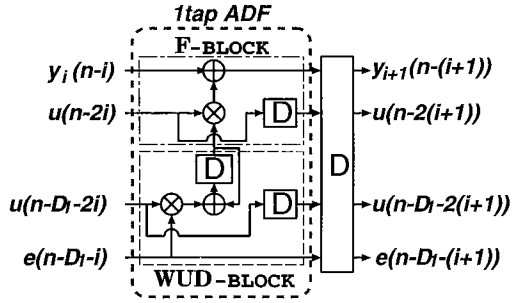


図1 $i$th モジュールシグナルフローグラフ
Fig. 1 Signal flowgraph for the $i$th module.

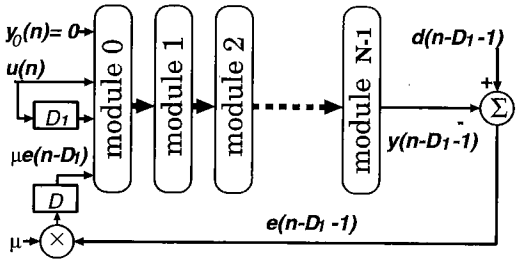


図2 従来法によるパイプライン適応フィルタ
Fig. 2 A pipelined adaptive filter base on the conventional method.

$$y_i(n-i)=y_{i-1}(n-i)+w_i(n-1-i)u(n-2i) \tag{16}$$

$$w_i(n-i)=w_i(n-1-i) + \mu e(n-D_1-i)u(n-D_1-2i) \tag{17}$$

となる．このモジュールを用いて構成されるパイプライン適応フィルタを図2に示す[5],[6]．図に示すように各モジュールは縦続に接続されるため，誤差信号は各モジュールで一つのディレイを受ける．また図より誤差ループに誤差信号を生成するための加算器と，ステップサイズを掛けるための乗算器が存在する．誤差ループからモジュールの係数更新部まで信号が伝わる遅延を，各モジュールのクリティカルパス以下にするには，誤差ループにディレイを一つ挿入する必要がある．よってフィルタ全体のディレイ $D_1$ は

$$D_1 = N+1 \tag{18}$$

としなければならない．

従来法に基づく構成において速度係数は式(13)，(15)より

$$S_{\text{module}} = \frac{1}{\tau_{mlt}+\tau_{add}} \tag{19}$$

となる．式(19)より従来法での速度係数は，乗算器および加算器の処理速度のみに依存し，構成に依存しない．しかしタップ数の増加と共に $D_1$ も増加するため，収束特性が劣化してしまうことが知られている[4]．また式(14)，(18)よりレイテンシーは

$$L_{\text{module}} = (N+1)\tau \tag{20}$$

となり，タップ数の増加に伴い劣化してしまう．従来法では式(18)よりタップ数 $N$ が決まると $D_1$ も決定するため，レイテンシーの改善は不可能である．

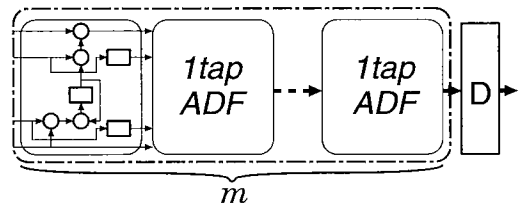


図3 ステージ ブロック図
Fig. 3 Block diagram of a stage.

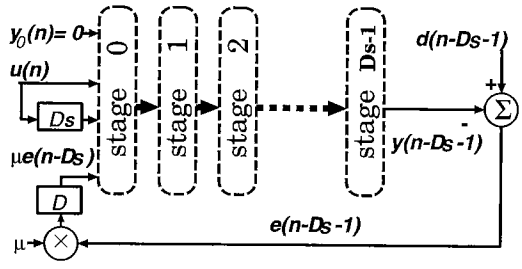


図4 提案法に基づくパイプライン適応フィルタ
Fig. 4 A pipelined adaptive filter based on the proposed method.

### 3.2 提案する構成

従来法ではパイプライン処理する上で1タップを1セルとしていた．そのため，タップ数 $N$ が大きいときに収束特性劣化や，レイテンシー劣化という問題が生じる．

そこで $m$ タップをひとまとめにして一つのセルを構成することを提案する[9],[10]．このセルをモジュールと区別してステージと呼ぶことにする．図3にステージのブロック図を示す．図3の $1tapADF$ は図1の破線内と一致し，適応フィルタ1タップに相当する．またステージを用いて構成されたパイプライン適応フィルタを図4に示す．

この構成での遅延近似におけるディレイを $D_s$ とすると，

$$D_s = \left\lceil \frac{N}{m} \right\rceil + 1 \tag{21}$$

と表せる．但し $\lceil x \rceil$ は $x$ を下回らない最小の整数を表す．また $j$ 番目のステージにおける入出力関係と係数更新はそれぞれ，

$$y_j(n-j) = y_{j-1}(n-j) + \boldsymbol{w}_j^T(n-j-1)\boldsymbol{u}_j(n-j) \tag{22}$$

$$\boldsymbol{w}_j(n-j) = \boldsymbol{w}_j(n-j-1) + \mu e(n-D_s-j)\boldsymbol{u}_j(n-D_s) \tag{23}$$

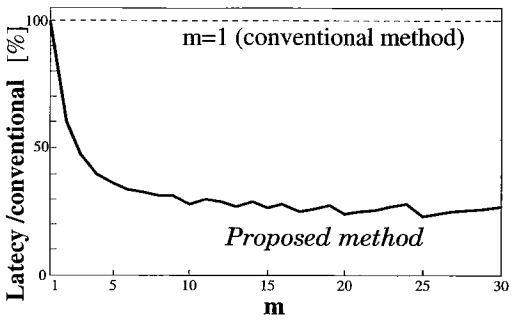


図5 レイテンシー特性
Fig. 5 Latency versus tap-length of stage.

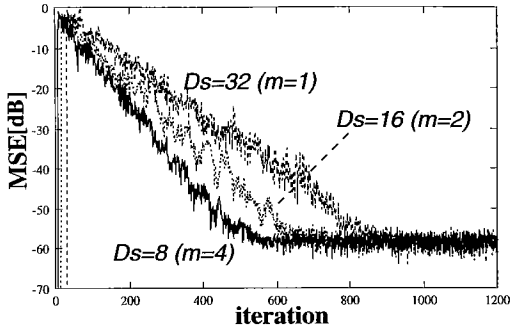


図6 DLMSの学習曲線
Fig. 6 Learning curves of DLMS.

と表せる．但し

$$\boldsymbol{w}_j(n) = [w_{jm}(n), w_{jm+1}(n), \cdots, w_{(i+j)m-1}(n)]^T \tag{24}$$

$$\boldsymbol{u}_j(n) = [u(n-j(m+1)), u(n-1-j(m+1)), \cdots, u(n-m+1-j(m+1))]^T \tag{25}$$

である．

従来法は式(22)～(25)において $m=1$ の特別な場合とみなせる．式(18)と式(21)を比較することで $1<m$ のときは $D_s<D_1$ となることがわかり，従来法に比べレイテンシーや収束特性の改善が可能となる．このことを以下に示す．

### 3.3 レイテンシーの改善

ステージ内のタップをトランスバーサルフィルタ構成にすると，ステージのクリティカルパスは

$$\tau = \tau_{mlt} + m\tau_{add} \tag{26}$$

となる．ここで $m$ はステージ内に含まれるタップ数である．式(14)，(21)，(26)より提案法でのレイテンシーは

$$L_{stage} = D_s\tau = \left(\left\lceil \frac{N}{m} + 1 \right\rceil\right)(\tau_{mlt} + m\tau_{add}) \tag{27}$$

となる．図5はステージタップ数 $m$ を変化させたときのレイテンシーを表す．図よりレイテンシーは $m$ の増加に伴う $\tau$ の増加より，$D_s$ の減少による影響を強く受ける．従来法では困難なレイテンシーの改善を，提案法では式(27)より $m$ を変化させることで可能であることがわかる．

### 3.4 シミュレーション

提案法は，従来法より小さなディレイを選択でき，収束特性を改善できることをシミュレーションにより示す．まず収束特性のシミュレーションを示す．条件として未知システムは32タップのローパスフィルタ，入力に分散1の定常白色ガウス信号，適応フィルタも32タップとした．所望信号 $d(n)$ には $-60$[dB]の白色雑音を加えた．ステップサイズ $\mu$ は，誤差が平均2乗で収束を保証する範囲で最大の値を用いた[4]．結果は独立な20回の試行の集合平均である．図6は $D_s=8$ $(m=4)$，$D_s=16$ $(m=2)$，$D_s=32$ $(m=1)$ それぞれの場合の結果である．従来法は $D_s=32$ のときと等しい．図6より提案法の構成により小さなディレイを選択でき，収束特性を改善できることがわかる．

## 4. スケジューリング

図4の構成において速度係数は原理的には，式(13)，(26)より

$$S_{stage} = \frac{1}{\tau_{mlt} + m\tau_{add}} \tag{28}$$

と表現される．上式は，式(19)との比較からステージ内タップ数 $m$ が大きいほど，速度係数が悪化することを示している．しかし複数タップをまとめたことでハードウェア構成の自由度が生じる．ここではステージ内の加算のスケジューリングを行うことで，速度係数の劣化が抑制できることを示す．更に乗算のスケジューリングを行うことにより，抑制した劣化において約半数のハードウェア量で実現可能なことを示す[10]．

### 4.1 準　　備

ステージのフィルタ部において，クリティカルパス

は各タップの1回の乗算と，その $m$ 個の結果の加算にかかる時間となる．係数更新部では各タップでの1回の乗算と，各係数を更新するための1回の加算にかかる時間となる．

$\tau_{fil}$，$\tau_{wud}$ をそれぞれフィルタ部，係数更新部のクリティカルパスとすると

$$\tau_{fil} = \tau_{mlt} + s\tau_{add} \tag{29}$$

$$\tau_{wud} = \tau_{mlt} + \tau_{add} \tag{30}$$

となる．ここで $\tau_{mlt}$，$\tau_{add}$ は各々1回の乗算と加算にかかる時間で，$s$ は $m$ 個の加算に要する加算器の段数で構成に依存し，

$$s \geq 1 \tag{31}$$

である．これより

$$\tau_{wud} \leqq \tau_{fil} \tag{32}$$

となり，速度係数に関してはフィルタ部のクリティカルパスのみ考慮すればよいことがわかる．

以下ではステージ内の演算のスケジューリングを行う．

### 4.2 加 算 器

はじめに加算のスケジューリングを検討する．ここでは 図7 に示す (a) トランスバーサルフィルタ構成と，(b) 二分木の構成を提案し，両者を比較する．それぞれの構成において $m$ 個の加算にかかる時間を $\tau_{(a)}$，$\tau_{(b)}$ とすると

$$\tau_{(a)} = \tau_{mlt} + m\tau_{add} \tag{33}$$

$$\tau_{(b)} = \tau_{mlt} + \lceil \log_2(m+1) \rceil \tau_{add} \tag{34}$$

となる．

構成 (a)，(b) 各々で $m$ を変化させたときの速度係数を図8に示す．但し実際の設計例から $\tau_{mlt} = 4\tau_{add}$ と仮定した[2],[10]．図8より二分木の構成を用いることでトランスバーサルフィルタ構成より，速度係数の劣化を抑えられることがわかる．また図9に各々の構成でのレイテンシーを示す．この場合にも二分木構成が優れていることがわかる．

### 4.3 乗 算 器

次にステージ内の乗算のスケジューリングを検討する．スケジューリングの条件として，フィルタ部および係数更新部の各々で乗算器の個数を $h_{fil}$，$h_{wud}$ と表し，$m$ 以下に制限する（$h_{fil} \leqq m, h_{wud} \leqq m$）．このとき一つの乗算器を複数回の乗算に効率良く使うため，パイプライン乗算器を用いることを考える．また加算のスケジューリングには二分木構成を用いる．

乗算器の個数に制約を定めたことにより，必ずしも式(32) は満たされるとは限らなくなってしまうことに注意しなければならない．すなわち係数更新部のクリティカルパスがフィルタ部より大きくなってしまう

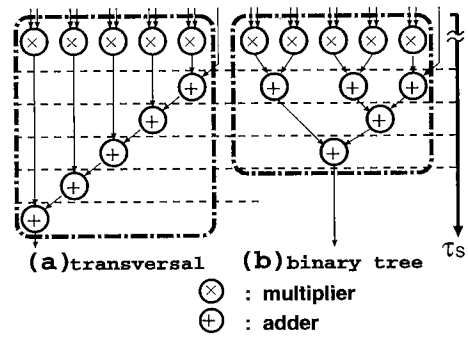


図7 加算のスケジューリング
Fig. 7 Scheduling of add operation.

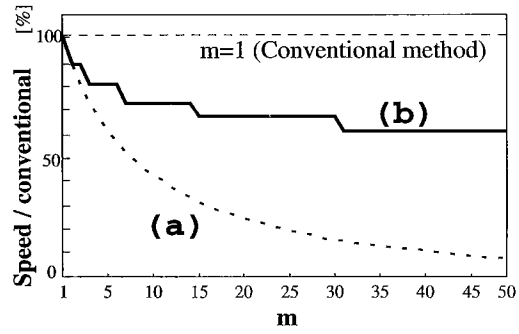


図8 速度係数特性
Fig. 8 Speed versus tap-lengh of stage.

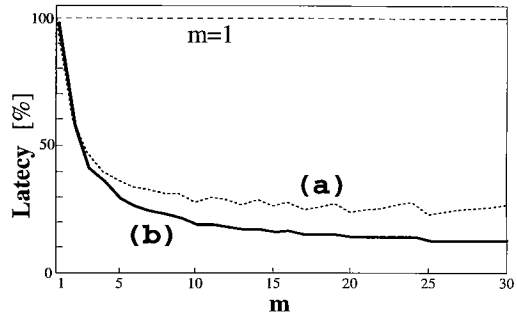


図9 レイテンシー特性
Fig. 9 Latency versus tap-lengh of stage.

場合がある．しかしながら後の議論を簡単にするため式(32)を満足するための条件を求める．

フィルタ部，係数更新部の各クリティカルパスは

$$\tau_{fil} = \tau_{mlt} + (\alpha + \beta - 1)\tau_{add} \tag{35}$$

$$\tau_{wud} = \tau_{mlt} + \left\lceil \frac{m}{h_{wud}} \right\rceil \tau_{add} \tag{36}$$

と表せる．ここで

$$\alpha = \left\lceil \frac{m}{h_{fil}} \right\rceil \tag{37}$$

$$\beta = \lceil \log_2(m - (\alpha - 1)h_{fil} + \lceil a_{\alpha-1} \rceil) \rceil \tag{38}$$

$$a_i = \frac{1 - h_{fil}}{2^i} + h_{fil} \tag{39}$$

である．これらの式より式(32)を満たすには，

$$h_{fil} \leqq h_{wud} \tag{40}$$

であればよいことがわかる．また式(32)で等号が成り立つのは $h_{fil} = h_{wud} = 1$ のときである．以下ではフィルタ部および係数更新部の乗算器は同数個の制約をおくものとし，

$$h_{mlt} \equiv h_{fil} = h_{wud} \tag{41}$$

とする．図10に実現例を示す．(a)は必要となる $m$ 個の乗算器をすべて用意した場合で，(b)は2個の乗算器で $m$ 回の乗算を実行する場合の例である．

式(13)，(35)より加算および乗算のスケジューリングを行ったときの速度係数は

$$S_{proposed} = \frac{1}{\tau_{mlt} + (\alpha + \beta - 1)\tau_{add}} \tag{42}$$

となる．但し $\alpha$，$\beta$ は各々式(37)～(39)で表される．図11は1ステージのタップ数 $m$ を5，10，20各々で，乗算器の個数を変化させたときの速度係数を表す．但し $\tau_{mlt} = 4\tau_{add}$ とした．図11より図8に示した二分木構成の速度係数を実現するのに，乗算器を $m$ 個用意する必要はなく，従来法では不可能な約半数個の乗算器で実現できることがわかる．

またレイテンシーは

$$L_{proposed} = \left( \left\lceil \frac{N}{m} + 1 \right\rceil \right) (\tau_{mlt} + (\alpha + \beta - 1)\tau_{add}) \tag{43}$$

と表せる．図12に乗算器の個数の制約によるレイテンシーへの影響を示す．ここでステージ内タップ数は $m = 30$ とした．図12より提案法において乗算器の個数を変化させても，従来法に比べ小さなレイテンシーとなることがわかる．

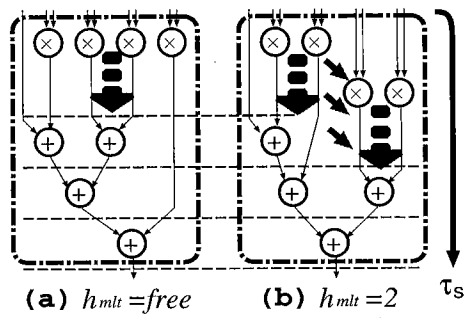


図10 乗算のスケジューリング
Fig. 10 Scheduling of multiply operations.

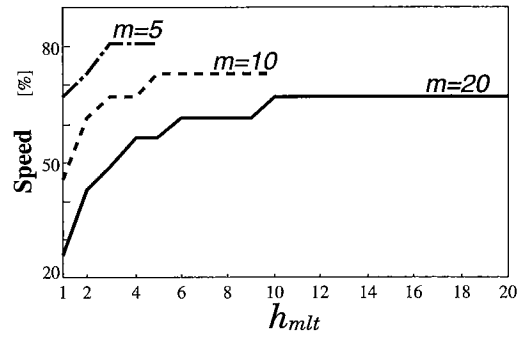


図11 速度係数特性
Fig. 11 Speed versus $h_{mlt}$.

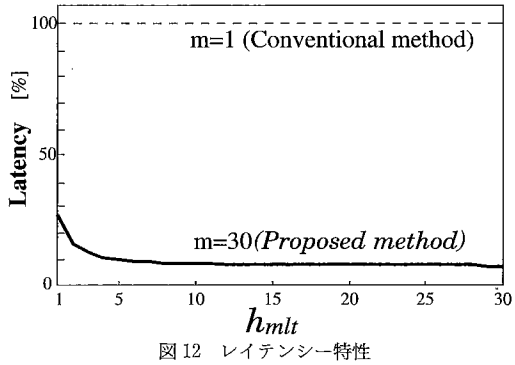


図12 レイテンシー特性
Fig. 12 Latency versus $h_{mlt}$.

### 4.4 ハードウェア量の検討

ここでは論理合成ツールを用いて設計した結果を示し，スケジューリングによるハードウェア量への影響を検討する．ハードウェア量の比較として，ここではゲート数を用いる．設計において，論理合成ツールにPARTHENON [11] を，合成に用いるライブラリには 1.5 μm ルール CMOS テクノロジーの実ライブラリを用いた．演算は2の補数表現による符号付き固定

表1 ゲート数
Table 1 Number of gates.

| | 加算器 | 乗算器 |
|---|---|---|
| 8 ビット | 70 | 1527 |
| 16 ビット | 138 | 4734 |

小数点演算とした．また加算器はけた上げ伝搬加算器（CPA: Carry Propagate Adder）で，乗算器はけた上げ保存加算器（CSA: Carry Save Adder）のアレーで構成し[7]，1回の演算にかかる時間は加算器1に対し乗算器4とした[2]．表1に8ビット，16ビット各々における設計結果を示す．表より乗算器のハードウェア量は加算器に比べ多大であり，フィルタ全体のハードウェア量の大半を乗算器が占めることがわかる．提案法は，従来法に比べ乗算器が約半数個に減少可能であるため，必要となるハードウェア量を大幅に削減することが可能となる．

## 5. むすび

本論文ではDLMSに基づくパイプライン適応フィルタの収束特性を改善できるアーキテクチャを提案した．提案法は一つのセルに複数のフィルタタップを含ませることで，遅延近似の度合を縮小することが可能となることを示した．これにより収束特性およびレイテンシーが改善でき，計算機シミュレーションにより有効性を確認した．また複数タップをまとめたことによる入力信号のクロックレートの低下も，ステージ内の演算にスケジューリングを行うことで抑制でき，かつハードウェア量も約半数に減少できることを示した．

今後は，PIPLMSへの本アーキテクチャの適用について考察を進める予定である．

**謝辞** 本研究において PARTHENON（パルテノン）を御提供して下さいましたNTTコミュニケーション科学研究所の皆様に感謝致します．

### 文　献

[1] B. Widrow and S.D. Stearms, "Adaptive Signal Processing," Prentice Hall, 1985.
[2] N.R. Shanbhag and K.K. Parhi, "Relaxed look-ahead pipelined LMS adaptive filters and their application to ADPCM coder," IEEE Trans. Circuits & Syst. II, vol.40, no.12, pp.753–766, Dec. 1993.
[3] R.H.-Cohen, H. Herzberg, and Y. Be'ery, "Delayed adaptive LMS filtering: current results," Proc. IEEE ICASSP'90, pp.1273–1276, April 1990.
[4] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," IEEE Trans. Acoust., Speech & Signal Process., vol.37, no.9, pp.1397–1405, Sept. 1989.
[5] 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 Process., vol.40, no.11, pp.2799–2803, Nov. 1992.
[6] 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.
[7] A.R. Omondi, "Computer Arithmetic Systems," Prentice Hall, 1994.
[8] H.H. Loomis and B. Sinha, "High speed recursive digital filter realization," Circuits, Syst., Signal Processing, vol.3, no.3, pp.267–294, March 1984.
[9] 松原勝重，西川清史，貴家仁志，"DLMSに基づくパイプライン適応フィルタ，" 1995信学春季全大，vol.A-211, March 1995.
[10] 松原勝重，西川清史，貴家仁志，"DLMSに基づくパイプライン適応フィルタの実現と評価，" 信学技報，CAS95–36, June 1995.
[11] Y. Nakamura, K. Oguri, A. Nagoya, M. Yukishita, and R. Nomura, "High-level synthesis design at NTT systems labs," IEICE Trans. Inf. & Syst., vol.E76-D, no.9, pp.1047–1054, Sept. 1993.

（平成7年8月11日受付，12月21日再受付）

**松原　勝重**　（学生員）

平7東京都立大・工・電子・情報工卒．現在同大大学院修士課程在学中．適応信号処理に関する研究に従事．

**西川　清史**　（正員）

平2東京都立大・工・電気工卒．平4同大大学院修士課程了．同年新日本製鉄（株）エレクトロニクス研究所勤務．平5東京都立大工学部電子情報工学科助手．現在に至る．適応信号処理および情報源符号化に関する研究に従事．IEEE会員．

**貴家　仁志**　（正員）

昭55長岡技科大・工・電気電子システム卒．昭57同大大学院修士課程了．同年東京都立大工学部電気工学科助手．現在，同大電子・情報工学科助教授．工博．マルチレート信号処理，適応信号処理および画像処理に関する研究に従事．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入門」，「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．