計測自動制御学会東北支部 第 257 回研究集会 (2010.5.31)
資料番号 257-10

# 分散演算形 LMS 適応フィルタの FPGA 実現

# FPGA Implementation of LMS Adaptive Digital Filter using Distributed Arithmetic

○細田晃史 *，佐々木拓郎 *，内田勝也 *，
高橋強 **，恒川佳隆 *

○ Akihumi Hosoda*, Takurou Sasaki*, Katsuya Uchida*,
Kyo Takahashi**, Yoshitaka Tsunekawa*

*岩手大学, **岩手県工業技術センター

*Iwate University, **Iwate Industrial Research Institute

**キーワード**： 分散演算 (distributed arithmetic), 並列計算 (parallel computing), 小規模ハードウエア (small hardware), 高サンプリングレート (high-sampling rate), FPGA(Field Programmable gate array)

**連絡先**： 〒 020-0852　岩手県盛岡市飯岡新田 3-35-2 岩手県工業技術センター電子情報技術部 高橋強,
Tel: (019)635-1115，Fax: (019)635-0311，E-mail: takahashi-kyou@pref.iwate.jp

## 1. はじめに

適応フィルタ (Adaptive Digital Filter, ADF) は，エコーキャンセラ，ノイズコントローラ，適応等化器などの様々な分野で用いられ，応用範囲を広げている．実現に際しては，高速サンプリングレート，良好な収束特性，短い出力滞在時間，小規模ハードウエア，低消費電力などが要求されるが，これらを同時に満たすことは困難であり，高性能なアルゴリズムや効果的なアーキテクチャが望まれている．

これまで，分散演算 (Distributed Arithmetic, DA) を LMS 適応フィルタ [1] に適用した分散演算形 LMS 適応フィルタ (LMS Adaptive Digital Filter using Distributed Arithmetic, DA-ADF) を提案した [2, 3, 4, 5]．分散演算は出力計算における係数ベクトルと入力信号ベクトルの内積演算に適用され，出力信号は，$N$ 次入力信号ベクトルのビットパターンにより指定される部分積のシフト加算により求められる．また，$2^N$ 個の部分積は適応プロセスにおいて逐次更新される．ここで，部分積の全集合を全適応関数空間（Whole Adaptive Function Space, WAFS）と呼ぶ．さらに，高次におけるハードウエア規模の増加と収束速度の劣化を補償する手法として，WAFS を分割したマルチメモリブロック構造 (Multi-memory Block Structure) を適用した DA-ADF(MDA-ADF) と，高サンプリングレートを実現するために，DA-ADF に特有の同時更新アルゴリズム [5], 同時更新アルゴリズムに準奇対称性を適用した新たな MDA-ADF を提案した [6]．MDA-ADF は乗算器を用いない “マルチプライヤレス” な構成が可能であり，小規模ハードウエア，低消費電力，良好な収束特性を有する高性能適応フィルタである。

ディジタルシステムの実現手段として，FPGA (Field Programmable gate array) が注目されている．FPGA は，書き換え可能なロジック・デバイスであり，設定の変更・開発が容易であるなどの利点があることから，通信基地局，大規

模ルータなど高度な処理を行う機器から，ディスプレイ，プロジェクタ，携帯端末など生活に身近な製品にまで幅広く採用されている．

本報告では，文献 [6] で提案した適応フィルタの FPGA 実装を行い性能を評価する．その際，高速サンプリングレート，短い出力滞在時間，小規模ハードウエアを達成するために，FPGA に適した加算方式の採用と内部メモリブロックの使用を検討する．その結果，大幅な改善効果を確認したので報告する．

## 2. 分散演算形 LMS 適応フィルタ

### 2.1 分散演算の適用

分散演算はベクトルの内積演算を効率よく計算する手法であり，原理的には処理時間が語長 $B$ にのみ依存する．マルチメモリブロック構造は，DA-ADF の高次における RAM 容量の増加抑制と更新確率の減少に伴う収束速度の劣化を補償する [2]．MDA-ADF は WAFS を $N$ 方向に $M$ 個に分割することにより，容量 $2^R (R = N/M)$ の分割された WAFS を $M$ 個有する．

以下，フィルタのタップ数を $N$，入力信号の語長を $B$，分割数を $M$ とする． 図 1 に MDA-ADF の基本構成， 図 2 にタイミングチャートを示す．分散演算は，$N$ 次係数ベクトル $\boldsymbol{W}(k)$ と $N$ 次入力信号ベクトル $\boldsymbol{S}(k)$ の内積演算

$$
\begin{aligned}
y(k) &= \boldsymbol{W}^T(k)\,\boldsymbol{S}(k) \\
\boldsymbol{W}(k) &= [w_0(k), \cdots, w_{N-1}(k)]^T \\
\boldsymbol{S}(k) &= [s(k), s(k-1), \cdots, s(k-N+1)]^T \quad (1)
\end{aligned}
$$

に対して適用される．出力計算は，$B$ 個のアドレスベクトルにより順次指定される部分積を WAFS から読み出してシフト加算を実行する．更新動作は，誤差信号 $e(k)$ を用いて出力計算で用いた部分積を順に更新する．時刻 $k$ における $B$ 個の部分積は WAFS の部分集合であり，これを適応関数空間（Adaptive Function Space, AFS）と呼ぶ． 式 (1) の $N$ 次入力信号ベクトルは以下のように表わされる．

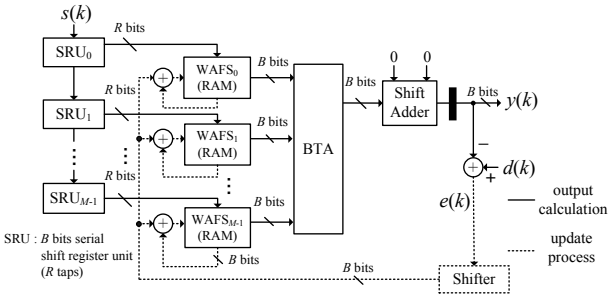


Fig. 1 Fundamental structure of MDA-ADF.

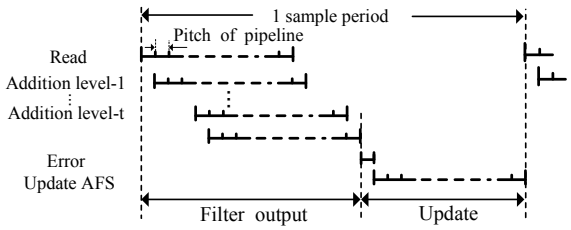


Fig. 2 Timing chart of MDA-ADF.

$$
\boldsymbol{S}(k) = \boldsymbol{A}^T(k)\,\boldsymbol{F}
$$

ここで，$T$ はマトリクスの転置を表す．bit パターンに分解した $N \times B$ 次アドレスマトリクスとスケーリングベクトルは

$$
\boldsymbol{A}(k) = \begin{bmatrix} b_0(k) & \cdots & b_0(k-N+1) \\ b_1(k) & \cdots & b_1(k-N+1) \\ \vdots & \ddots & \vdots \\ b_{B-1}(k) & \cdots & b_{B-1}(k-N+1) \end{bmatrix}, \quad (2)
$$

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

である．ここで，$b_i(k), i = 0, \cdots, B-1$，は $s(k)$ の $i$ 番目の bit を表す．式 (2) の列ベクトルを $M$ 個に分割し，$R$ 個のアドレスベクトルで指定される $m$ 番目の適応関数空間 $\boldsymbol{P}_m(k)$ は

$$
\begin{aligned}
\boldsymbol{P}_m(k) &= \left[p_{m0}(k), p_{m1}, \cdots, p_{m(B-1)}(k)\right]^T \\
&= \boldsymbol{A}_m^T(k)\,\boldsymbol{W}_m(k) \\
m &= 0, \cdots, M-1
\end{aligned}
$$

と表され，フィルタ出力 $y(k)$，更新式，誤差信号は，$d(k)$ を所望信号とすると次式で表される．

$$
y(k) = \sum_{m=0}^{M-1} \boldsymbol{F}^T\,\boldsymbol{P}_m(k) \qquad (3)
$$

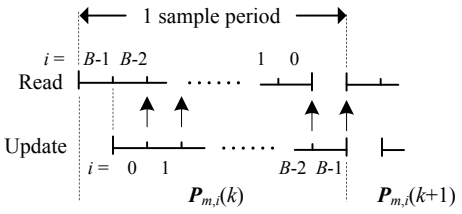


Fig. 3 Relation between Read and Update process.

$$\begin{aligned} \boldsymbol{P}_m(k+1) &= \boldsymbol{P}_m(k) + 0.5\mu Re(k)\,\boldsymbol{F} \\ e(k) &= d(k) - y(k) \end{aligned}$$

## 2.2 高速アルゴリズムと準奇対称性の利用

ディレードアップデートは出力計算と更新動作を並列に実行する高速手法であるが，過去の誤差信号を用いることにより収束速度が劣化することが知られている [7]．しかし，先に提案した同時更新アルゴリズムでは収束速度の劣化が小さい [5]．

本構成において，出力計算は重みの小さいアドレスベクトル，更新動作は重みの大きいアドレスベクトルから順次実行する．WAFS の読み出し・更新を図 3 に示す．ここで，$\boldsymbol{P}_{m,i}(k)$ は $M$ 分割された $m$ 番目の WAFS であり，$i(=0,1,\cdots,B-1)$ は時刻 $k$ における更新タイミングを表す．次いで，リタイミングにより更新動作の一連処理である “WAFS の読み出し処理（アドレスデコードとセレクト）”と “更新値計算・書込み動作 ”を分離して並列化し，読み出し処理を 1 ピッチだけ先行して開始する．WAFS の構成を 図 4 (b) に示す．処理の分離は，更新値計算用加算器の入力にラッチを挿入して実現する．さらに，最上位のアドレスベクトルが読み出す要素に対する符号反転処理を不要とし，ピッチ時間を短縮する．これは， 図 4 (c) のように，あらかじめ MSB を反転した信号を用いることにより実現する．本構成では例外処理をなくして出力計算のパイプライン処理をスムーズにすることにより出力滞在時間を短縮する．

更新動作においては，ハードウエアの増加を許容して，通常の更新要素と準奇対称の関係にある要素も同時に更新する．まず， 図 4 (a) のように $e(k)$ と $-e(k)$ を同時に計算する．この $-e(k)$ は準奇対称性の関係にある要素を更新する誤差信号である．異符号の誤差信号を用いて求められた更新値により， 図 4 (b) のように WAFS を更新する．この更新手法を用いることによりハーフメモリアルゴリズムと同様に収束速度を向上させることが可能となる．本構成のアルゴリズムを以下に示す．フィルタ出力 $y(k)$ と分割された適応関数空間の出力 $y_m(k)$ は次式で表される．

$$\begin{aligned} y(k) &= \sum_{m=0}^{M-1} y_m(k), \\ y_m(k) &= \sum_{i=0}^{B-1} \boldsymbol{F}'^{T}_{B-1-i}\,\boldsymbol{P}_{m,B-1-i}(k). \end{aligned}$$

ここで，スケーリングベクトル $\boldsymbol{F}'_i$ は，

$$\begin{aligned} \boldsymbol{F}'_0 &= [2^0, 0, 0, \cdots, 0, 0]^T, \\ \boldsymbol{F}'_1 &= [0, 2^{-1}, 0, \cdots, 0, 0]^T, \\ &\vdots \\ \boldsymbol{F}'_{B-1} &= [0, 0, \cdots, 0, 2^{-B+1}]^T, \\ i &= 0, 1, \cdots, B-1. \end{aligned}$$

$\boldsymbol{P}_{m,i}(k)$ は $M$ 分割された $m$ 番目の AFS である．また，

$$\boldsymbol{P}_m(k) \equiv \boldsymbol{P}_{m,B-1}(k-1)$$

の関係がある．以上より，更新式は次式で表わされる．なお，$\overline{\boldsymbol{P}}$ は $\boldsymbol{P}$ の準奇対称の関係にある AFS である．

$$\boldsymbol{P}_{m,i}(k+1) = \boldsymbol{P}_{m,i}(k) + 0.5\mu Re(k-1)\boldsymbol{F}'_i, \quad (4)$$

$$\overline{\boldsymbol{P}}_{m,i}(k+1) = \overline{\boldsymbol{P}}_{m,i}(k) - 0.5\mu Re(k-1)\boldsymbol{F}'_i. \quad (5)$$

## 2.3 アーキテクチャ

入力信号レジスタは $(2N+1) \times B$ 個 のレジスタを内部に有し，WAFS の要素を指定するアドレスマトリクスを $M$ 個に分割された適応関

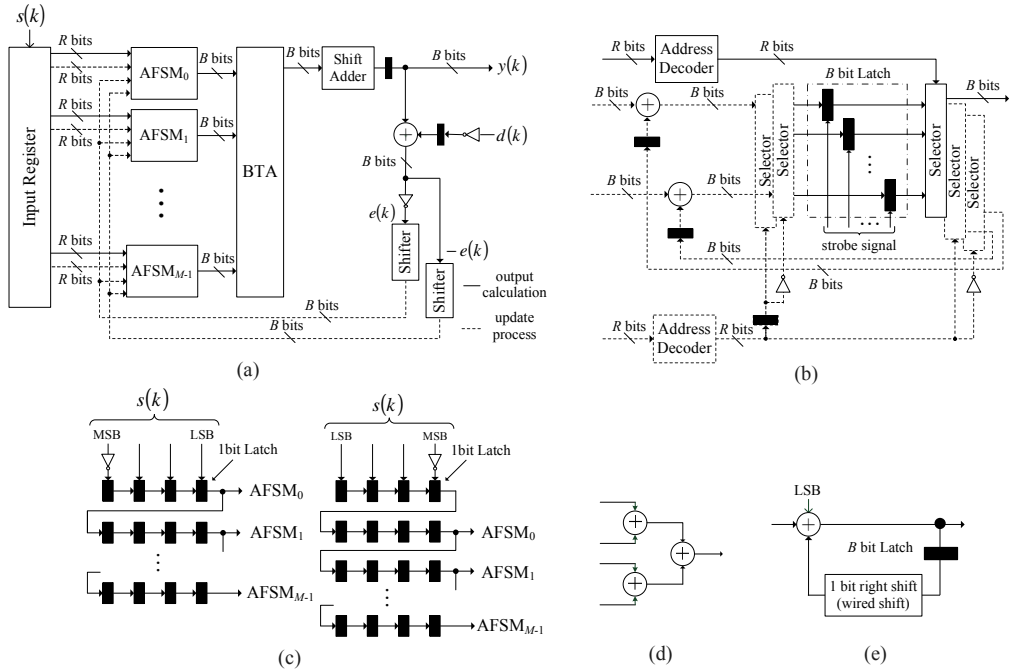


Fig. 4 Proposed architecture[6]. (a)Whole architecture (b)AFSM(Adaptive Function Space Module) (c)Input Register (d)BTA(Binary Tree Adder) (e)Shift Adder

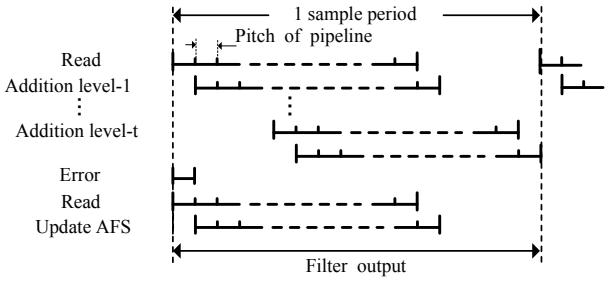


Fig. 5 Timing chart of the proposed architecture[6].

数空間モジュール (AFSM) に供給する．$M$ 個の AFSM は，$B$bit のレジスタ $2^R+2$ 個と $R$bit のレジスタ 1 個，デコーダ，セレクタ，加算器から構成され，セレクタはフィルタ出力に用いる AFS の読み出しと，更新のための読み出し書き込みを行う．デコーダは，アドレスマトリクスから 1 つのアドレスベクトルを選択し，選択信号をセレクタに供給する．$M$ 個の WAFS からの出力はバイナリツリーアダー (BTA) により加算され，シフト加算器を用いて出力を計算する．また，誤差計算は更新における先行開始の動作と並列に実行し，$e(k-1)$ と $-e(k-1)$ を同時に求める．次いで，WAFS は先行開始により読み出された値とシフトされた誤差信号を用いて更新される．

## 3. FPGA 実現

文献 [6] の構成を Altera 社の FPGA である CycloneII に実装して性能を評価した．CycloneII の特徴として， 図 6 のように，内部にエンベデッド・マルチプライヤ（乗算器），M4K メモリ・ブロック（以下，M4K），PLL(Phase Locked Loop)，LAB(Logic Array Block)，LAB 内に 16 個の LE(Logic Element) を有している．そして，LAB 間の通信は，横に隣接する LAB 間を高速通信するダイレクト・リンク接続，横方向の通信を行うロウ・インタコネクト，縦方向の通信を行うカラム・インタコネクトを有する．また，LAB 内には LE 間の通信を高速に行うローカル接続，隣接する LE 間を高速通信するキャリー・チェインを保持している．これらの機能を利用してより高性能な構成を提案する．

### 3.1 加算方式の検討

本構成はメモリ・アキュムレータ構成とも呼ばれるように，加算器を多用した構成であるた

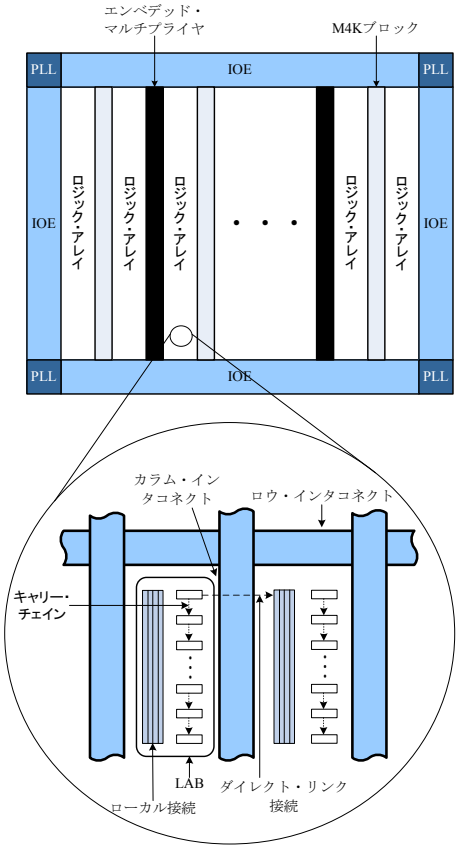


Fig. 6 Block diagram of CycloneII.

め加算器の性能が適応フィルタの性能に大きく影響する．加算器の方式として，桁上げ先見加算器 (Carry Look-Ahead Adder, CLAA) が知られている．CLAA は高速演算が可能ではあるがゲート規模が大きい．FPGA 実現では，面積の増加に伴う LAB 間の配線遅延が増加し，実装レベルでの高速性は低下する．

一方，桁上げ伝播加算器 (Ripple Carry Adder, RCA) は LSB から MSB 方向へのキャリー伝播を行う方式であり，長語長では伝播時間の増加に伴い高速性が低下する．しかし，CycloneII の LE 間，あるいはカラム方向の LAB 間には高速伝播が可能なキャリー・チェインが配線されており，RCA におけるキャリー伝播を高速に実行可能である．CLAA と RCA を演算語長 24bit で比較した結果，LE 数，演算時間共に RCA が良好な結果を示した．本構成は，$N = 32$，$R = 2$ に対して 49 個もの加算器を使用しており，RCA の採用により LE 数，サンプリングレート，出力滞在時間の大幅な改善が可能である．

Table 1 Condition of the Evaluations.

| | |
|---|---|
| Description Language | VHDL |
| Synth. and Fit. Tool | QuartusII v.8 |
| Synth. Option | Balance |
| Fitter Option | Auto |
| Target Device | CycloneII C70F672C6 |
| Filter Taps [taps] | 32 |
| Filter Division | 16 |
| Word Length [bits] | 16 |
| Calculation [bits] | 24 |

### 3.2 M4K メモリ・ブロックの使用

CycloneII には，LE の他にもメモリ機能として使用することができる M4K を有している．M4K の通信は，LAB 間と同様にダイレクト・リンク接続，ロウ・インタコネクト，カラム・インタコネクトを利用可能であり，利用可能な LAB を減らすことなく，LAB と同等の速度でメモリ機能を使用することができる．また，M4K はメモリ機能に加えてシフト機能も有するため，本構成における入力レジスタを M4K に割り当てる．これにより，FPGA においても入力レジスタを効率的に構成することが可能である．

## 4. 評価

乗算器を使用した LMS-ADF[1]，LMS-ADF に分散演算とマルチメモリブロック構造を適用した MDA-ADF[2]，文献 [6] に基づいて今回検討した Type1 と Type2 を評価する．LMS-ADF で使用する乗算器は Booth アルゴリズムに Wallace Tree 方式と CLAA を用いた乗算器，加算器は CLAA を使用した．Type1 は CLAA と入力レジスタを LAB 上に配置，Type2 は RCA は LAB，入力レジスタは M4K 上に配置した．FPGA の開発環境を 表 1 に，QuartusII による評価結果を 表 2 に，適応フィルタの性能評

Table 2　Comparison of Hardware Performances on the FPGA ($B$=16,$N$=32,$R$=2).

| | LMS[1] | MDA[2] | Type1[6] | Type2[6] |
|---|---|---|---|---|
| Clock Rate [MHz] | 19.4 | 55.2 | 68.0 | 84.2 |
| Amount of Hardware [gates] | 42,064 | 8,084 | 8,456 | 5,108 |

Table 3　Comparison of ADF Performances on FPGA ($B$=16,$N$=32,$R$=2).

| | LMS[1] | MDA[2] | Type1[6] | Type2[6] |
|---|---|---|---|---|
| Required Clocks [clocks] | 1 | 38 | 21 | 21 |
| Sampling Rate [MHz] | 19.4 | 1.45 | 3.24 | 4.01 |
| Output Latency [ns] | 40.2 | 380.6 | 308.7 | 249.3 |

価を 表 3 に示す．なお，評価結果にはピンから LAB までの遅延時間も含まれている．

ハードウエアの評価結果である 表 2 より，分散演算を用いた MDA，Type1，Type2 は乗算器を用いる LMS のわずか 20% 程度の回路規模で実現可能である．小規模実現が可能であることは，マルチプライヤレス構造の分散演算形 LMS 適応フィルタの特徴である．文献 [6] に基づく Type1 と Type2 は MDA の改良型であり，出力計算と更新動作の同時実行，そして，処理のリタイミングによりクロックレートが向上している．Type2 は小規模な RCA の使用により，CLAA を使用する Type 1 よりも回路規模が約 60% に減少している．

次いで，適応フィルタの評価である 表 3 より，Type1 と MDA を比較すると，サンプリングレートは 123% 向上し，出力滞在時間は 18.9% 減少している．これは，高速アルゴリズムによるサンプリングレートの向上効果であり，同時にスムーズなパイプライン処理の効果として出力滞在時間が減少している．次に，Type2 は Type1 に比較してサンプリングレートは 23.8% 向上し，出力滞在時間は 19.2% 減少していることがわかる．これは，回路規模，高速性に優れた RCA 加算器と M4K の使用による回路規模の削減，高速化の効果である．

Table 4　Improvement of the Performances by employing RCA Adders and M4K Memory Block. Type1: CLAAs and Input Registers on LABs, Type2: RCAs on LABs and Input Registers on M4K

| | Ratio (Type2/Type1) |
|---|---|
| Clock Rate | 1.238 |
| Amount of Hardware | 0.604 |
| Sampling Rate | 1.238 |
| Output Latency | 0.808 |

## 5.　まとめ

分散演算 LMS 適応フィルタの効果的 FPGA 実現について検討した．その結果，配線遅延と回路規模の小さい桁上げ伝播加算器と，M4K メモリ・ブロックの使用により回路規模を 0.6 倍，サンプリングレートを 1.24 倍，出力滞在時間を 0.81 倍に改善した．今後は，より高性能化を目指して配線遅延を考慮したレイアウトの検討を進める．

## 参考文献


1) B. Widrow and M. E. Hoff,“Adaptive Switching Circuit,” IRE EWSCON Conv. Rec., pp96-104, 1960.

2) 恒川, 高橋, 豊田, 三浦, “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アーキテクチャ, ” 信学論 (A), vol.J-82-A, no.10, pp.1518-1528, Oct. 1999.

3) 高橋，恒川，豊田，三浦，“ハーフメモリアルゴリズムに基づく分散演算形 LMS 適応フィルタの高性能アーキテクチャ”，信学論 (A)，Vol. J-84-A No.6 pp.777-787, June 2001.

4) 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.

5) 橘内慎次郎, 内田勝也, 佐藤慎悟, 高橋 強, 恒川佳隆, “分散演算を用いたパイプライン LMS 適応ディジタルフィルタの高性能アーキテクチャ”, 計測自動制御学会東北支部第 242 回研究集会, 242-2, May 13, 2008.

6) 佐々木拓郎, 高橋 強, 内田勝也, 恒川佳隆, “出力計算と更新過程を並列化した分散演算形 LMS 適応フィルタの高性能 VLSI パイプラインアーキテクチャ”, 計測自動制御学会東北支部 45 周年記念学術講演会, 1308, Sep 7, 2009.

7) 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.