A Block-Floating-Point-Based Realization of the Block LMS Algorithm

Mrityunjoy Chakraborty, *Senior Member, IEEE*, Rafiahamed Shaik, *Student Member, IEEE*, and Moon Ho Lee, *Senior Member, IEEE*

***Abstract*—An efficient scheme is proposed for implementing the block LMS (BLMS) algorithm in a block-floating-point framework that permits processing of data over a wide dynamic range at a processor complexity and cost as low as that of a fixed-point processor. The proposed scheme adopts appropriate formats for representing the filter coefficients and the data. Using these and a new upper bound on the step size, update relations for the filter weight mantissas and exponent are developed, taking care so that neither overflow occurs, nor are quantities which are already very small multiplied directly. It is further shown how the mantissas of the filter coefficients and also the filter output can be evaluated faster by suitably modifying the approach of the fast BLMS algorithm.**

***Index Terms*—Block floating point, block LMS (BLMS), fast BLMS (FBLMS), overflow.**

I. Introduction

The block floating point (BFP) format provides an elegant means of floating-point (FP) emulation on a simple, low-cost fixed-point (FxP) processor. In BFP, a common exponent is assigned to a group of variables. As a result, computations involving these variables can be carried out in simple FxP-like manner, while the presence of the exponent provides a FP-like high dynamic range. This has prompted several researchers in the recent past to use the BFP format for efficient realization of many signal processing systems and algorithms, including various forms of digital filters ([1]–[5]) and unitary transforms ([6], [7]). The BFP format has also been used in several digital audio data transmission standards like NICAM (stereophonic sound system for PAL TV standard), the audio part of MUSE (Japanese HDTV standard) and DSR (German Digital Satellite Radio System). However, almost all the research efforts in this area have focussed on systems having constant coefficients and not on systems like adaptive filters that have time-varying parameters. A BFP treatment to adaptive filters faces certain difficulties, not encountered in the fixed coefficient case, namely, the following.

- Unlike a fixed coefficient filter, the filter coefficients in an adaptive filter can *not* be represented in the simpler FxP form, as the coefficients in effect evolve from the data by a time update relation;
- The two principal operations in an adaptive filter—filtering and weight updating, are mutually coupled, thus requiring an appropriate arrangement for joint prevention of overflow.

Recently, a BFP-based approach has been proposed for efficient realization of the LMS-based transversal adaptive filters [8], which was later extended to the normalized LMS algorithm [9] and the gradient adaptive lattice [10]. The philosophy used in [8] employs block processing technique and can provide considerable savings in computational complexities when applied to the block LMS (BLMS) algorithm [11], as shown in this paper. For this, we first recast the BLMS algorithm using the framework of [8]. This requires adoption of appropriate BFP format for the filter coefficients which remains invariant as the coefficients are updated from block to block. Using this, along with the BFP representation of the data as used in [8] and a new upper bound on the algorithm step size, update relations for the filter weight mantissas and exponent are developed, maintaining overflow free operation all throughout. Note that the BLMS weight update relation is more complex than its LMS counterpart, as the former needs to sum up several products between data vectors and error samples. Special care had to be taken in its computation using the adopted BFP format so that neither overflow occurs, nor are quantities which are already very small multiplied directly. Next, we show how the filter output mantissas and the filter weight mantissas can be evaluated faster, by appropriately adjusting the approach of the FFT-based fast BLMS (FBLMS) algorithm [11]. Such an adjustment requires introduction of one extra inverse fast Fourier transform (IFFT) operation in the weight update loop in order to implement a time domain constraint. However, despite this, considerable gains in computational complexities are achieved, since all the FFT/IFFTs are based on BFP arithmetic only.

II. BFP Background

The BFP representation can be considered as a special case of the FP format, where every nonoverlapping block of N incoming data has a joint scaling factor corresponding to the data sample with the highest magnitude in the block. In other words, given a block $[x_1,\ldots,x_N]$, we represent it as $[x_1,\ldots,x_N] = [\overline{x}_1,\ldots,\overline{x}_N]2^{\gamma}$ where $\overline{x}_l (= x_l 2^{-\gamma})$ represents the mantissa for $l = 1,2,\ldots,N$ and the block exponent γ is defined as $\gamma = \lfloor \log_2 \mathrm{Max} \rfloor + 1 + S$ where $\mathrm{Max} = \max(|x_1|,\ldots,|x_N|)$, "$\lfloor . \rfloor$" is the so-called floor function, meaning rounding down to the closest integer and the integer S is a scaling factor which is needed to prevent overflow during filtering operation. Due to the presence of S, the range of each mantissa is given as $0 \leq |\overline{x}_l| < 2^{-S}$. The scaling factor S can be calculated from the

Manuscript received July 19, 2005. This work was supported in part by the Institute of Information Technology Assessment (IITA), South Korea. This paper was recommended by Associate Editor B. C. Levy.

M. Chakraborty and R. Shaik are with the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur 721302, India (e-mail: mrityun@ece.iitkgp.ernet.in; rafi_ahmed@ece.iitkgp.ernet.in).

M. H. Lee is with the Department of Information and Communication, Chonbuk National University, Chonju, Korea (e-mail: moonho@chonbuk.ac.kr).

Digital Object Identifier 10.1109/TCSII.2006.880333

1057-7130/$20.00 © 2006 IEEE

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR. Downloaded on June 24, 2009 at 06:22 from IEEE Xplore. Restrictions apply.

inner product computation representing filtering operation. An inner product is calculated in BFP arithmetic as

$$\begin{aligned} y(n) &= \mathbf{w}^t\mathbf{x}(n)\\ &= [w_0\overline{x}(n)+\ldots+w_{L-1}\overline{x}(n-L+1)]\,2^{\gamma}\\ &= \overline{y}(n)2^{\gamma} \end{aligned} \tag{1}$$

where $\mathbf{w}$ is a length L, FxP filter coefficient vector and $\mathbf{x}(n)$ is the data vector at the nth index, represented in the aforesaid BFP format. For no overflow in $y(n)$, we need $|\overline{y}(n)| < 1$ at every time index, which can be satisfied [2] by selecting $S \geq S_{\min} = \lceil \log_2(\sum_{k=0}^{L-1}|w_k|)\rceil$ where "$\lceil . \rceil$" is the so-called ceiling function, meaning rounding up to the closest integer.

III. Proposed Implementation

Consider a length-L BLMS-based adaptive filter that takes an input sequence $x(n)$, which is partitioned into nonoverlapping blocks of length P each, with the jth block $(j \in Z)$ consisting of $x(jP+r), r \in Z_P = 0, 1, \ldots, P-1$. The filter coefficients are updated from block to block as

$$\mathbf{w}(j+1) = \mathbf{w}(j) + \mu\Sigma_{r=0}^{P-1}\mathbf{x}(jP+r)e(jP+r) \tag{2}$$

where $\mathbf{w}(j) = [w_0(j)w_1(j)\ldots w_{L-1}(j)]^t$ is the tap weight vector corresponding to the jth block, $\mathbf{x}(jP+r) = [x(jP+r)x(jP+r-1)\ldots x(jP+r-L+1)]^t$ and $e(jP+r) = d(jP+r) - y(jP+r)$ is the output error at $n = jP+r$. The sequence $d(jP+r)$ is the so-called desired response available during the initial training period and $y(jP+r) = \mathbf{w}^t(j)\mathbf{x}(jP+r)$ is the filter output at $n = jP+r$, with μ denoting the so-called step size parameter.

The proposed scheme consists of two simultaneous BFP representations, one for the filter coefficient vector $\mathbf{w}(j)$ and the other for the given data, namely, $x(n)$ and $d(n)$. These are as follows.

A. BFP Representation of the Filter Coefficient Vector

Here, the tap weight vector $\mathbf{w}(j)$ is represented in a scaled format as

$$\mathbf{w}(j) = \overline{\mathbf{w}}(j)2^{\psi_j} \tag{3}$$

where $\overline{\mathbf{w}}(j)$ and ψ_j are respectively the filter mantissa vector and the filter block exponent which are updated separately over the block index j. Note that in the above representation, all components of $\mathbf{w}(j)$ are normalized by the same factor 2^{ψ_j}. In our treatment, the exponent ψ_j is a nondecreasing function of j with zero initial value and is chosen to ensure that $|\overline{w}_k(j)| < (1/2)$, $k \in Z_L = \{0, 1, \ldots, L-1\}$. If a data vector $\mathbf{x}(jP+r)$ is given in the aforesaid BFP format as $\mathbf{x}(jP+r) = \overline{\mathbf{x}}(jP+r)2^{\gamma}$, where $\gamma = ex+S, ex = \lfloor \log_2 M \rfloor + 1, M = \max(|x(jP+r-k)||k \in Z_L)$ and S is an appropriate scaling factor, then, the filter output $y(jP+r)$ can be expressed as $y(jP+r) = \overline{y}(jP+r)2^{\gamma+\psi_j}$ with $\overline{y}(jP+r) = \overline{\mathbf{w}}^t(j)\overline{\mathbf{x}}(jP+r)$ denoting the output mantissa. To prevent overflow in $\overline{y}(jP+r)$, it is required that $|\overline{y}(jP+r)| < 1$. However, in the proposed scheme, we restrict $\overline{y}(jP+r)$ to lie between $+(1/2)$ and $-(1/2)$, i.e., $|\overline{y}(jP+r)| < (1/2)$. Since $|\overline{y}(jP+r)| \leq \Sigma_{k=0}^{L-1}|\overline{w}_k(j)||\overline{x}(jP+r-k)|, 0 \leq |\overline{x}(jP+r-k)| < 2^{-S}$ and $|\overline{w}_k(j)| < (1/2)$, this implies a lower limit of S as $S_{\min} = \lceil \log_2 L \rceil$. The two conditions: $|\overline{w}_k(j)| < (1/2)$ and $|\overline{y}(jP+r)| < (1/2)$ ensure no overflow during updating of $\overline{\mathbf{w}}(j)$ and computation of output error mantissa, respectively, as shown later.

B. BFP Representation of the Given Data

The input data $x(n)$ and the desired response sequence $d(n)$ are partitioned jointly in nonoverlapping blocks of N samples each with the ith block, $i \in Z^+ = \{0, 1, 2, \ldots\}$, consisting of $x(n)$, $d(n)$ for $n \in Z_i' = \{iN, iN+1, \ldots, iN+N-1\}$. In our present treatment, we choose N based on the following constraints:

1) $N \geq L-1$, meaning that at any point of time, data from at most two adjacent blocks may come under filtering operation.
2) $N = KP$ for some integer K, meaning that in a block of duration N, the filter coefficients are updated a total of K times over K sub-blocks of length P each.

The data samples $x(n)$ and $d(n)$ constituting a block are jointly scaled so as to have a common BFP representation for the block under consideration. This means that, for $n \in Z_i'$, $x(n)$ and $d(n)$ are expressed as

$$x(n) = \overline{x}(n)2^{\gamma_i} \qquad d(n) = \overline{d}(n)2^{\gamma_i} \tag{4}$$

where γ_i is the common block exponent for the ith block and is given as $\gamma_i = ex_i + S_i$ where $ex_i = \lfloor \log_2 M_i \rfloor + 1$ and $M_i = \max\{|x(n)|, |d(n)||n \in Z_i'\}$. The scaling factor S_i is assigned as per the following exponent assignment algorithm:

Exponent Assignment Algorithm: Assign $S_{\min} = \lceil \log_2 L \rceil$ as the scaling factor to the first block and for any $(i-1)$th block, assume $S_{i-1} \geq S_{\min}$.

Then, if $ex_i \geq ex_{i-1}$

choose $S_i = S_{\min}$ (i.e., $\gamma_i = ex_i + S_{\min}$)

else (i.e., $ex_i < ex_{i-1}$)

choose $S_i = (ex_{i-1} - ex_i) + S_{\min}$, (i.e., $\gamma_i = ex_{i-1} + S_{\min}$).

Note that when $ex_i \geq ex_{i-1}$, we can either have $ex_i + S_{\min} \geq \gamma_{i-1}$ (Case A) implying $\gamma_i \geq \gamma_{i-1}$, or, $ex_i + S_{\min} < \gamma_{i-1}$ (Case B) meaning $\gamma_i < \gamma_{i-1}$. However, for $ex_i < ex_{i-1}$ (Case C), we always have $\gamma_i \leq \gamma_{i-1}$.

Additionally, we rescale the elements $\overline{x}(iN-L+1), \cdots, \overline{x}(iN-1)$ by dividing by $2^{\Delta\gamma_i}$, where $\Delta\gamma_i = \gamma_i - \gamma_{i-1}$. Equivalently, for the elements $x(iN-L+1), \cdots, x(iN-1)$, we change S_{i-1} to an effective scaling factor of $S_{i-1}' = S_{i-1} + \Delta\gamma_i$. This permits a BFP representation of the data vector $\mathbf{x}(n)$ with common exponent γ_i during block-to-block transition phase too, i.e., when part of $\mathbf{x}(n)$ comes from the $(i-1)$th block and part from the ith block. In practice, such rescaling is effected by passing each of the delayed terms

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR. Downloaded on June 24, 2009 at 06:22 from IEEE Xplore. Restrictions apply.

$\overline{x}(n-k)$, $k = 1,\ldots,L-1$, through a rescaling unit that applies $\Delta\gamma_i$ number of right or left shifts on $\overline{x}(n-k)$ depending on whether $\Delta\gamma_i$ is positive or negative respectively. This is, however, done only at the beginning of each block, i.e., at indices $n = iN$, $i \in Z$. Also, note that though for the case (A), $\Delta\gamma_i \geq 0$, for (B) and (C), however, $\Delta\gamma_i \leq 0$, meaning that in these cases, the aforesaid mantissas from the $(i-1)$th block are actually scaled up by $2^{-\Delta\gamma_i}$. It is, however, not difficult to see that the effective scaling factor S'_{i-1} for the elements $x(iN-L+1),\cdots,x(iN-1)$ still remains lower bounded by $S_{\min}$, thus ensuring no overflow during filtering operation.

Formulation of the BLMS Algorithm in BFP Format: We begin by considering the lth sub-block, $l = 0,1,\ldots,K-1$, within the ith block. This consists of data at the indices $(iK+l)P+r$, $r = 0,1,\ldots,P-1$. Replacing $(iK+l)$ by j, one can then write $e(jP+r)$ as $e(jP+r) = \overline{e}(jP+r)2^{\psi_j+\gamma_i}$, where the mantissa $\overline{e}(jP+r)$ is given as

$$\overline{e}(jP+r) = \overline{d}(jP+r)2^{-\psi_j} - \overline{y}(jP+r) \qquad (5)$$

Clearly, computation of $\overline{e}(jP+r)$ involves an additional step of right-shift operation on $\overline{d}(jP+r)$—an operation that comes frequently in FP arithmetic. However, since in an adaptive filter, filter coefficients are derived from data and thus can not be represented in FxP format when data is given in a scaled form, such a step seems to be unavoidable. It is easy to check that $|\overline{e}(jP+r)| < 1$, since

$$\begin{aligned}|\overline{e}(jP+r)| &\leq \left|\overline{d}(jP+r)\right|2^{-\psi_j} + |\overline{y}(jP+r)|\\ &< 2^{-(S_i+\psi_j)} + \frac{1}{2} \leq \frac{2^{-\psi_j}}{L} + \frac{1}{2}\end{aligned} \qquad (6)$$

as $2^{-S_i} \leq 1/L$. Except for $\psi_j = 0$, $L = 1$, the R.H.S. is always less than or equal to 1.

For the above description of $e(jP+r)$, $\mathbf{x}(jP+r)$ and $\mathbf{w}(j)$, the weight update (2) can be written as

$$\mathbf{w}(j+1) = \overline{\mathbf{v}}(j)2^{\psi_j} \qquad (7)$$

where

$$\overline{\mathbf{v}}(j) = \overline{\mathbf{w}}(j) + \mu\Sigma_{r=0}^{P-1}\overline{\mathbf{x}}(jP+r)\overline{e}(jP+r)2^{2\gamma_i}. \qquad (8)$$

As stated earlier, $\overline{\mathbf{w}}(j+1)$ is required to satisfy $|\overline{w}_k(j+1)| < (1/2)$ for $k \in Z_L$, which can be realized in several ways. Our preferred option is to limit $\overline{\mathbf{v}}(j)$ so that $|\overline{v}_k(j)| < 1$, $k \in Z_L$. Then, if each $\overline{v}_k(j)$ happens to be lying within $\pm(1/2)$, we make the assignments

$$\overline{\mathbf{w}}(j+1) = \overline{\mathbf{v}}(j), \qquad \psi_{j+1} = \psi_j. \qquad (9)$$

Otherwise, we scale down $\overline{\mathbf{v}}(j)$ by 2, in which case

$$\overline{\mathbf{w}}(j+1) = \frac{1}{2}\overline{\mathbf{v}}(j), \qquad \psi_{j+1} = \psi_j + 1. \qquad (10)$$

In order to have $|\overline{v}_k(j)| < 1$, $k \in Z_L$ satisfied, we observe that $|\overline{v}_k(j)| \leq |\overline{w}_k(j)| + \mu\Sigma_{r=0}^{P-1}|\overline{x}(jP+r-k)||\overline{e}(jP+r)|2^{2\gamma_i}$. Since $|\overline{w}_k(j)| < 1/2$, $k \in Z_L$, it is sufficient to have $\mu\Sigma_{r=0}^{P-1}|\overline{x}(jP+r-k)||\overline{e}(jP+r)|2^{2\gamma_i} \leq 1/2$. Taking the upper bound of $|\overline{e}(jP+r)|$ as $[2^{-(S_i+\psi_j)} + (L/2)2^{-S_i}]$ and recalling that $|\overline{x}(jP+r-k)| < 2^{-S_i}$, this implies

$$\mu \leq \frac{2^{-2ex_i}}{P[2^{-\psi_j+1}+L]}. \qquad (11)$$

It is easy to verify that the above bound for μ is valid not only when each element of $\overline{\mathbf{x}}(jP+r)$ in (8) comes purely from the ith block, but also during transition from the $(i-1)$th to the ith block with $ex_i \geq ex_{i-1}$, for which, after necessary rescaling, we have $S'_{i-1} \geq S_i = S_{\min}$ implying $|\overline{x}(jP+r-k)| < 2^{-S_i}$ and thus $|\overline{y}(jP+r)| < (L/2)2^{-S_i}$. For $ex_i < ex_{i-1}$, however, the upper bound expression given by (11) gets modified with ex_i replaced by ex_{i-1}, as in that case, we have $\gamma_i = ex_{i-1} + S'_{i-1}$ with $S'_{i-1} = S_{\min} < S_i$ meaning $|\overline{x}(jP+r-k)| < 2^{-S'_{i-1}}$ and thus $|\overline{y}(jP+r)| < (L/2)2^{-S'_{i-1}}$, leading to $|\overline{e}(jP+r)| < [2^{-(S'_{i-1}+\psi_j)} + (L/2)2^{-S'_{i-1}}]$.

From above, we obtain a general upper bound for μ by equating ψ_j to its lowest value of zero and replacing ex_i by $ex_{\max} = \max\{ex_i | i \in Z^+\}$ in (11). The general upper bound is given by

$$\mu \leq \frac{2^{-2ex_{\max}}}{P[L+2]}. \qquad (12)$$

The above bound is actually less than $2/[Ptr\mathbf{R}]$ which is the upper bound for μ for convergence of the BLMS algorithm. To see this, we note that $|x(n)| < 2^{ex_{\max}}$ and thus $E[x^2(n)] < 2^{2ex_{\max}}$. This implies $tr\mathbf{R} < L2^{2ex_{\max}}$ and thus $2/[Ptr\mathbf{R}] > 2^{-2ex_{\max}}/P(L+2)$.

Finally, for practical implementation of $\overline{\mathbf{v}}(j)$ as given by (8), we need to evaluate the update term: $\mu\Sigma_{r=0}^{P-1}\overline{x}(jP+r-k)\overline{e}(jP+r)2^{2\gamma_i}$, $k \in Z_L$ in such a way that no overflow occurs in any of the intermediate products, shifts, or the summation involved. At the same time, we need to avoid direct product of quantities which could be very small, as that may lead to loss of several useful bits via truncation. For this purpose, we proceed as follows: if $ex_i \geq ex_{i-1}$, then we have $S_i = S_{\min}$ and we express $2^{2\gamma_i}$ as $2^{2\gamma_i} = 2^{2ex_i+S_{\min}}2^{S_{\min}}$. If, instead, $ex_i < ex_{i-1}$, then, $S'_{i-1} = S_{\min}$, $\gamma_i = ex_{i-1} + S'_{i-1}$ and we decompose $2^{2\gamma_i}$ as $2^{2\gamma_i} = 2^{2ex_{i-1}+S_{\min}}2^{S_{\min}}$. The factors $2^{2ex_i+S_{\min}}$ (or, $2^{2ex_{i-1}+S_{\min}}$) and $2^{S_{\min}}$ are then distributed to compute the update term as follows:

Step 1) → $\mu_i = \mu 2^{2ex_i+S_{\min}}$ if $ex_i \geq ex_{i-1}$; if $ex_i < ex_{i-1}$, $\mu_i = \mu 2^{2ex_{i-1}+S_{\min}}$

Step 2) → $\mu_i\overline{e}(jP+r) = \overline{e}_1(jP+r)$ (say),

Step 3) → $\overline{x}(jP+r-k)2^{S_{\min}} = \overline{x}_1(jP+r-k)$ (say),

Step 4) → $\Sigma_{r=0}^{P-1}\overline{x}_1(jP+r-k)\overline{e}_1(jP+r)$.

It is easy to check that the operations described in Steps 1)–4) produce no intermediate overflow. Firstly, from (12), it follows that $\mu_i \leq 2^{S_{\min}}/P[L+2]$. Since, $L \leq 2^{S_{\min}} < 2L$, this implies $\mu_i < 2L/P[L+2]$. For the BLMS algorithm, sub-block length P is at least two, thus ensuring $\mu_i < 1$. Next, note that in all cases, $|\overline{e}(jP+r)| < [2^{-(S_{\min}+\psi_j)} + (L/2)2^{-S_{\min}}]$. Using this and the above observation that $\mu_i \leq 2^{S_{\min}}/P[L+2]$, it is easily seen that $|\overline{e}_1(jP+r)| < 1/2P$. Similarly, in step 3, $|\overline{x}_1(jP+r-k)| < 1$, since $|\overline{x}(jP+r-k)| < 2^{-S_{\min}}$ Finally, in Step 4), the

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR. Downloaded on June 24, 2009 at 06:22 from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF THE BLMS ALGORITHM REALIZED IN BFP FORMAT
(INITIAL CONDITIONS: $\psi_0 = 0, |\overline{w}_k(0)| < 1/2, k \in Z_L$)

```
1. Preprocessing:
   Using the data for the $i$-th block, $x(n)$ and $d(n)$, $n \in Z_i'$, $i \in Z^+$
   (stored during the processing of the $(i-1)$-th block),
 (a) Evaluate block exponent $\gamma_i$ as per the Exponent Assignment
     Algorithm of Section 3 and express $x(n)$, $d(n)$, $n \in Z_i'$ as
     $x(n) = \overline{x}(n).2^{\gamma_i}$, $d(n) = \overline{d}(n).2^{\gamma_i}$,
 (b) Rescale the following elements of the $(i-1)$-th block:
     $\{\overline{x}(n) | n = iN - L + 1, ..., iN - 1\}$ as
     $\overline{x}(n) \rightarrow \overline{x}(n)\, 2^{-\Delta\gamma_i}$, $\Delta\gamma_i = \gamma_i - \gamma_{i-1}$; [Also, for Step 3
     of Section III, rescale the same separately by $2^{-\Delta\gamma_i + S_{min}}$.]
2. Processing for the $i$-th block:
   For the $l$-th sub-block within the $i$-th block, $l = 0, 1, \ldots, K-1$,
    Define $j = iK + l$.
    For $n = jP + r$, $r = 0, 1, \ldots, P-1$
     Filter output:
     $\overline{y}(n) = \overline{\mathbf{w}}^t(j)\overline{\mathbf{x}}(n)$,
     Output error (mantissa) computation:
     $\overline{e}(n) = \overline{d}(n)\, 2^{-\psi_j} - \overline{y}(n)$.
    end
   $ex\_out(j) = \gamma_i + \psi_j$.
   ($ex\_out(j)$ is the filter output exponent for the j-th sub-block)
   Filter weight updating:
   Compute $\overline{u}_k(j) = \mu \sum_{r=0}^{P-1} \overline{x}(jP + r - k)\overline{e}(jP + r) 2^{2\gamma_i}$ for all
   $k \in Z_L = \{0, 1, ..., L-1\}$ following $Step$ 1 $-$ $Step$ 4 of Section III.
   $\overline{\mathbf{v}}(j) = \overline{\mathbf{w}}(j) + \overline{\mathbf{u}}(j)$.
   (where $\overline{\mathbf{u}}(j) = [\overline{u}_0(j), \ldots, \overline{u}_{L-1}(j)]^t$)
   If $|\overline{v}_k(j)| < \frac{1}{2}$ for all $k \in Z_L$,
   then
       $\overline{\mathbf{w}}(j+1) = \overline{\mathbf{v}}(j)$,
       $\psi_{j+1} = \psi_j$,
   else
       $\overline{\mathbf{w}}(j+1) = \frac{1}{2}\overline{\mathbf{v}}(j)$,
       $\psi_{j+1} = \psi_j + 1$.
   end.
   end.
   $i = i + 1$.
   Repeat steps 1 to 2.
```

summation evaluates the update term, which is pre-constrained to be less than half in magnitude.

Noting that the vectors $\overline{\mathbf{x}}(jP + r)$ in (8) overlap with each other, in Step 3), we need to shift only the P terms: $\overline{x}(jP + r)$, $r \in Z_P$ by $2^{S_{\min}}$. [For $r = 0$, $j = iK$, $\overline{x}(jP - k)$, $k = 1, 2, \ldots, L-1$ correspond to the last $(L-1)$ mantissas of the $(i-1)$th block, rescaled by $2^{-\Delta\gamma_i}$. Further scaling of them by $2^{S_{\min}}$ can be carried out during the block formatting stage only.] The proposed BFP treatment to the BLMS algorithm is summarized in Table I.

Fast Implementation: A treatment similar to the one used in the derivation of the FBLMS algorithm [11] from the BLMS algorithm can be used in the above context for a faster evaluation of the filter output mantissa $\overline{y}(n)$ and the weight vector mantissa $\overline{\mathbf{w}}(j)$. For the lth sub-block within the ith block, $0 \le l \le K-1$, i.e., for $n = jP + r$, $r = 0, 1, \ldots, P-1$, $j = iK + l$, the filter output mantissa $\overline{y}(n) = \overline{\mathbf{w}}^t(j)\overline{\mathbf{x}}(n)$ is obtained by convolving the input data mantissa sequence $\overline{x}(n)$ with the filter coefficient mantissas $\overline{w}_0(j), \ldots, \overline{w}_{L-1}(j)$ and thus can be realized efficiently by the overlap-save method via $M = L + P - 1$ point FFT, where the first $L-1$ points come from the previous sub-block, for which the output is to be discarded. Similarly, the weight update term in Step 4 above, viz., $\Sigma_{r=0}^{P-1}\overline{x}_1(jP + r - k)\overline{e}_1(jP + r)$ can be obtained by the usual circular correlation technique, by employing M point FFT and setting the last $P-1$ output terms as zero. The resulting scheme for fast computation of $\overline{y}(n)$ and $\overline{\mathbf{w}}(j)$ is demonstrated in Fig. 1. Note that the following are true.

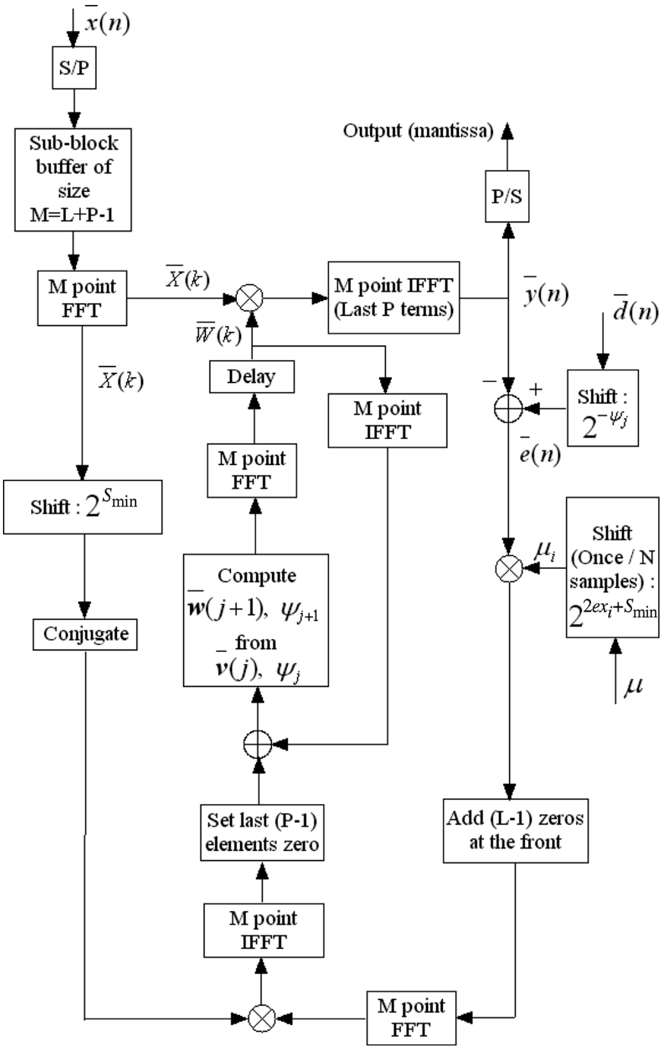

Fig. 1. Fast implementation of the proposed BFP-based BLMS algorithm.

a) The weight update loop in Fig. 1 is different from the weight update loop of the conventional FBLMS scheme [11], as, an additional IFFT is used here to get the filter weights back to the time domain, in order to implement the weight update relations (9) and (10). This is needed, since, in our proposed scheme, weight updating requires checking the condition: $|\overline{v}_k(j)| < 1/2$ for all k, $0 \le k \le L-1$, which is a purely time domain constraint and has no equivalent frequency domain counterpart. However, as the FFT and IFFT computations are FxP based, the overall computational cost of the proposed fast implementation scheme still remains much less than a conventional FP-based FBLMS realization, as shown in the next section.
b) Each FFT/IFFT in Fig. 1 can be implemented using BFP arithmetic [6]. For a M point FFT, this means that in each of the $\log_2 M$ stages, both the real and the imaginary parts of all input samples are jointly scaled up/down by the same factor to prevent overflow and at the same time, to

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR. Downloaded on June 24, 2009 at 06:22 from IEEE Xplore. Restrictions apply.

TABLE II
A COMPARISON BETWEEN THE BFP VIS-À-VIS THE FP-BASED REALIZATIONS OF THE BLMS ALGORITHM. NUMBER OF OPERATIONS REQUIRED PER ITERATION FOR (a) WEIGHT UPDATING, AND (b) FILTERING ARE SHOWN. [MAC: MULTIPLY AND ACCUMULATE, MC: MAGNITUDE CHECK, EC: EXPONENT COMPARISON, EA: EXPONENT ADDITION.]

(a)	MAC	Shift	MC	EC	EA
BFP	$(L+1)P$	$P+L$	L	Nil	1
FP	$(L+1)P$	$(2L+1)P$	Nil	LP	$(2L+2)P$

(b)	MAC	Shift	EC	EA
BFP	L	Nil	Nil	1
FP	L	$2L$	L	$2L$

TABLE III
A COMPARISON BETWEEN THE BFP VIS-À-VIS THE FP-BASED REALIZATIONS OF THE FBLMS ALGORITHM. NO. OF OPERATIONS PER SUB-BLOCK ARE SHOWN ($M = L + P - 1$, $r = \log_2 M$)

	MAC	Shift	EC	EA	Add	MC
BFP	$12Mr$ $+8M$	$12Mr$ $+L$	Nil	$6r$	$12Mr$ $+2M$	L
FP	$10Mr$ $+8M$	$35Mr$ $+16M$	$15Mr$ $+6M$	$30Mr$ $+18M$	$10Mr$ $+2M$	Nil

make better usage of the available dynamic range, at the output of the stage. The shift on $\overline{X}(k), k = 0, 1, \ldots, M - 1$ by $2^{S_{\min}}$ as shown in Fig. 1 can be absorbed in the up/down scaling processes present in the M point FFT preceding it and the M point IFFT following it.

IV. COMPLEXITY ISSUES

The proposed schemes rely mostly on FxP arithmetic, resulting in computational complexities much less than that of their FP-based counterparts. For example, to compute the filter output in Table I, L "Multiply and Accumulate (MAC)" operations (FxP) are needed to evaluate $\overline{y}(n)$ and at the most, one exponent addition operation to compute the exponent $ex\_out(j)$. In FP, this would require L FP-based MAC operations. Note that given three numbers in FP (normalized) format: $A = \overline{A}2^{e_a}$, $B = \overline{B}2^{e_b}$, $C = \overline{C}2^{e_c}$, the MAC operation $A + BC$ requires the following steps: i) $e_b + e_c$, i.e., exponent addition (EA); ii) exponent comparison (EC) between e_a and $e_b + e_c$; iii) shifting either $\overline{A}$ or $\overline{B}/\overline{C}$; iv) FxP-based MAC; and finally, v) renormalization, requiring shift and exponent addition. In other words, in FP, computation of $y(n)$ will require the following additional operations over the BFP-based realization: a) $2L$ shifts (assuming availability of single cycle barrel shifters); b) L EC; and c) $2L - 1$ EA. Similar advantages exist in weight updating also. Table II provides a comparative account of the two approaches in terms of number of operations required per iteration. Note that the number of additional operations required under FP increases linearly with both the filter length L and the sub-block length P. It is easy to verify from Table II that given a low cost, simple FxP processor with single cycle MAC and barrel shifter units, the proposed scheme is about *six times faster* than a FP-based implementation, for moderately large values of L and P.

For the algorithm proposed in Fig. 1, similar computational advantages exist over the conventional FP-based FBLMS algorithm. As the major computational block here is M-point FFT/IFFT, we consider a typical butterfly computation stage that takes as input $X_m(p)$ and $X_m(q)$ and performs the following computation: $X'_m(q) = W_M^s X_m(q)$; $X_{m+1}(p) = X_m(p) + X'_m(q)$, $X_{m+1}(q) = X_m(p) - X'_m(q)$, where $W_M = e^{-(j2\pi/M)}$. In a FP treatment, both the real and the imaginary parts of $X_m(p)$, $X_m(q)$ and W_M^s are represented in normalized FP format, resulting in a total of 4 MAC (FxP), 14 shift, 12 EA, 6 EC, and 4 addition (FxP) operations per butterfly. In BFP [6], however, both the real and imaginary parts of the above quantities are in FxP format and the input quantities of all the butterflies in each stage are scaled up/down by the same number. This gives rise to 4 MAC (FxP), 4 additions, and 4 shifts per butterfly, along with one EA for each stage of the FFT. Assuming M to be a power of 2, i.e., $M = 2^r$, there are r stages in each M-point FFT/IFFT, each having $M/2$ butterflies. From this and also taking into account the complexities involved in FFT addition and multiplication, we obtain a comparative account of the two approaches in terms of number of operations required per sub-block. This is given in Table III. Once again, for moderately large values of M, it is easily seen that the proposed scheme of Fig. 1 is between *three to four times faster* than a FP-based FBLMS algorithm.

V. CONCLUSION

The BLMS algorithm is presented in a BFP framework that ensures simple FxP operations in most of the computations while maintaining a FP-like wide dynamic range via a block exponent. Care is also taken to prevent overflow by a new upper bound on the step size μ and a dynamic scaling of the data. A faster realization of the proposed scheme is developed by suitable modification of the FFT-based FBLMS algorithm.

REFERENCES

[1] K. R. Ralev and P. H. Bauer, "Realization of block floating point digital filters and application to block implementations," *IEEE Trans. Signal Process.*, vol. 47, no. 4, pp. 1076–1086, Apr. 1999.

[2] K. Kalliojärvi and J. Astola, "Roundoff errors in block floating point systems," *IEEE Trans. Signal Process.*, vol. 44, no. 4, pp. 783–790, Apr. 1996.

[3] P. H. Bauer, "Absolute error bounds for block floating point direct form digital filters," *IEEE Trans. Signal Process.*, vol. 43, no. 8, pp. 1994–1996, Aug. 1995.

[4] S. Sridharan and G. Dickman, "Block floating point implementation of digital filters using the DSP56000," *Microprocess. Microsyst.*, vol. 12, no. 6, pp. 299–308, Jul.–Aug. 1988.

[5] F. J. Taylor, "Block floating point distributed filters," *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 300–304, Mar. 1984.

[6] D. Elam and C. Lovescu, A block floating point implementation for an N-point FFT on the TMS320C55X DSP Texas Instruments, Dallas, TX, Texas Instrum. Appl. Rep. SPRA948, Sep. 2003.

[7] A. Erickson and B. Fagin, "Calculating FHT in hardware," *IEEE Trans. Signal Process.*, vol. 40, pp. 1341–1353, Jun. 1992.

[8] A. Mitra, M. Chakraborty, and H. Sakai, "A block floating point treatment to the LMS algorithm: efficient realization and a roundoff error analysis," *IEEE Trans. Signal Process.*, pp. 4536–4544, Dec. 2005.

[9] A. Mitra and M. Chakraborty, "The NLMS algorithm in block floating point format," *IEEE Signal Process. Letters*, pp. 301–304, Mar. 2004.

[10] M. Chakraborty and A. Mitra, "A block floating point realization of the gradient adaptive lattice filter," *IEEE Signal Process. Letters*, pp. 265–268, Apr. 2005.

[11] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR. Downloaded on June 24, 2009 at 06:22 from IEEE Xplore. Restrictions apply.