AN EFFICIENT BLOCK FLOATING POINT IMPLEMENTATION OF THE LMS ALGORITHM

Mrityunjoy Chakraborty, Abhijit Mitra,

E. & E. C. E. Dept.,
I. I. T. Kharagpur, INDIA.
e-mail: (mrityun, abhijit)@ece.iitkgp.ernet.in

Hideaki Sakai

Graduate School of Informatics,
Kyoto University, JAPAN.
e-mail: hsakai@i.kyoto-u.ac.jp

ABSTRACT

An efficient scheme is presented for implementing the LMS-based transversal adaptive filter in block floating point (BFP) format which permits processing of data over a wide dynamic range at a processor cost marginally higher than that of a fixed point processor. Appropriate BFP formats for both the data and the filter coefficients have been adopted and adjustments made in filtering as well as weight updating operations in order to sustain the adopted format and also to prevent overflow in both these operations jointly. For the presented method to work properly, the algorithm step size is to be chosen below an upper limit, which is, however, not very restrictive when compared with the upper bound for convergence, thereby having marginal effect on convergence speed.

1. INTRODUCTION

High signal to quantization noise ratio (SNR) over a reasonably large dynamic range is one basic requirement for digital signal processing systems in several application areas. The block floating point (BFP) data format, proposed originally by Wilkinson [1], is one effective means to achieve this at a moderately low processor complexity and cost. Under this scheme, the given data is partitioned in non-overlapping blocks and based on the relative magnitudes of data samples in each block, a common exponent is assigned, which permits a floating point (FP) like representation of the data, with fixed point (FxP) like computation within every block. The advantages provided by this system, namely, a wide dynamic range like the FP scheme with hardware complexity, power requirement and cost nearly as low as that of a FxP processor, have prompted its usage in efficient implementation of many real time algorithms. In the context of digital filters, several attempts have been made in recent past to realize BFP based fixed coefficient digital filters ([2]- [3], [6]-[10]). Some studies ([3], [4], [5]) have also been made to investigate the associated numerical error behavior. However, to the best of our knowledge, no effort has so far been made to extend this treatment to adaptive filters which present more complex structures including error feedback. A BFP treatment to adaptive filters faces certain difficulties, not encountered in the fixed coefficient case, namely, (a) unlike a fixed coefficient filter, the filter coefficients in an adaptive filter can not be represented in the simpler fixed point form, as the coefficients in effect evolve from the data by a time update relation; (b) the two principal operations in an adaptive filter, namely, filtering and weight updating, are mutually coupled, thus requiring an appropriate arrangement for joint prevention of overflow.

In this paper, we present a novel scheme for BFP realization of the LMS-based transversal adaptive filter [11]. The proposed approach adopts appropriate BFP formats for the data and the filter coefficients separately and makes necessary adjustments in both the filtering as well as weight update equations so as to sustain the adopted format and also to prevent overlow jointly in both these operations. Special care also has been taken so that the proposed scheme works smoothly during block-to-block transition phase when part of the filter input vector comes from one block and the rest from an adjacent block with the two blocks having different exponents. The proposed method restricts the algorithm step size to a new upper bound which is less than its well known upper limit for convergence, i.e., $2/tr\mathbf{R}$ ($\mathbf{R}$: input correlation matrix). However, the new bound is not overly restrictive when compared with the above theoretical limit and its effect on convergence speed has been marginal.

2. THE BFP ARITHMETIC

The BFP representation can be considered as a special case of FP format, where every non-overlapping block of N incoming data has a joint scaling factor corresponding to the largest (magnitude) data sample in the block. In other words, given a block $[x_1, ..., x_N]$, we represent it as

$$[x_1, ..., x_N] = [\overline{x}_1, ..., \overline{x}_N].2^{\gamma} \qquad (1)$$

where $\overline{x}_l (= x_l.2^{-\gamma})$ represents the mantissa for $l = 1, 2, ..., N$ and the block exponent γ is defined as

$$\gamma = \lfloor log_2 Max \rfloor + 1 + S \qquad (2)$$

where $Max = max(|x_1|, ..., |x_N|)$, '$\lfloor . \rfloor$' 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. For the presence of S, the range of each mantissa is given as $|\overline{x}_l| \in [0, 2^{-S})$. The scaling factor S can be calculated from the inner product computation representing filtering operation. An inner product is calculated in BFP format as

$$y(n) \quad = \quad \mathbf{w}^T \mathbf{x}(n)$$

$$= [w_0\overline{x}(n) + ... + w_{L-1}\overline{x}(n-L+1)].2^{\gamma}$$
$$= \overline{y}(n).2^{\gamma} \quad (3)$$

where $\mathbf{w}$ is an Lth order fixed point filter coefficient vector and $\mathbf{x}(n)$ is the data vector at the nth index, represented in BFP format. For no overflow in $y(n)$, we need $|\overline{y}(n)| \leq 1$ at every time index, which can be satisfied by selecting [3]

$$S \geq S_{min} = \lceil log_2(\sum_{k=1}^{L} |w_k|) \rceil \quad (4)$$

where '$\lceil . \rceil$' is the so-called ceiling function, meaning rounding up to the closest integer.

Note that, if (B_d + one sign) bits are used to represent each mantissa within the block and if (B_γ + one sign) bits are used to account for the block exponent, then effectively, under BFP system, each sample can be equivalently represented with $(B_d + 1) + (B_\gamma + 1)/N$ bits because the block exponent is assigned only once for the whole block. This particular strength makes this format more attractive than FxP or FP systems.

3. THE PROPOSED IMPLEMENTATION

Consider an Lth order LMS based adaptive filter that takes an input sequence $x(n)$ and updates the weights as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{x}(n)e(n) \quad (5)$$

where $\mathbf{w}(n) = [w_0(n)w_1(n)...w_{L-1}(n)]^T$ is the tap weight vector at the nth index, $\mathbf{x}(n) = [x(n)x(n-1)...x(n-L+1)]^T$, $e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n)$, with $d(n)$ being the so-called desired response available during the initial training period and μ denoting the so-called step size paramemter.

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

(a) **BFP format of the filter coefficient vector**:

In this, at each index of time, we have a scaled representation of the filter coefficient vector as

$$\mathbf{w}(n) = \overline{\mathbf{w}}(n).2^{\psi_n} \quad (6)$$

where ψ_n is a time-varying block exponent that needs to be updated at each index n and is chosen to ensure that each $|\overline{w}_k(n)| < \frac{1}{2}$ for $k = 0, 1, ..., L-1$. If a data vector $\mathbf{x}(n)$ is given in the aforesaid BFP format as $\mathbf{x}(n) = \overline{\mathbf{x}}(n).2^{\gamma}$, where $\gamma = ex + S$, $ex = \lfloor log_2 M \rfloor + 1$, $M = max(|x(n-k)| \mid k = 0, 1, ..., L-1)$ and S is an appropriate scaling factor, then, the filter output $y(n)$ can be expressed as

$$y(n) = \overline{y}(n).2^{\gamma+\psi_n} \quad (7)$$

with $\overline{y}(n) = \overline{\mathbf{w}}^T(n)\overline{\mathbf{x}}(n)$ denoting the output mantissa. To prevent overflow in $\overline{y}(n)$, it is required that $|\overline{y}(n)| < 1$. However, in the proposed scheme, we restrict $\overline{y}(n)$ to lie between $+\frac{1}{2}$ and $-\frac{1}{2}$, i.e., $|\overline{y}(n)| < \frac{1}{2}$ for reasons explained later. Since $|\overline{y}(n)| \leq \sum_{k=0}^{L-1} |\overline{w}_k(n)||\overline{x}(n-k)|$, $0 \leq |\overline{x}(n-k)| < 2^{-S}$ and $|\overline{w}_k(n)| < \frac{1}{2}$, this implies a lower limit of S as

$$S_{min} = \lceil log_2 L \rceil. \quad (8)$$

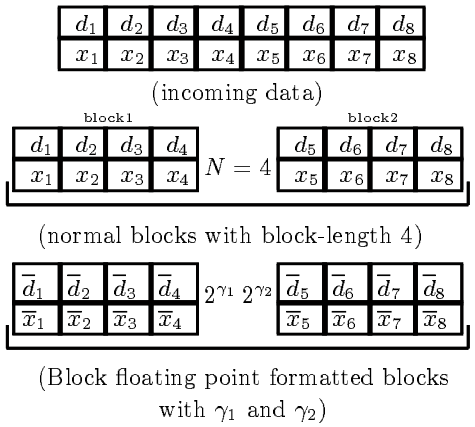

Fig. 1. Block floating point joint scaling mechanism.

(b) **BFP representation of the given data**:

The input data $x(n)$ and the desired response sequence $d(n)$ are partitioned jointly in non-overlapping blocks of N samples each ($N \geq L-1$) as shown in Fig. 1, with the ith block ($i \in Z$) consisting of $x(n)$, $d(n)$ for $n \in Z_i = \{iN, iN+1, ..., iN+N-1\}$. Further, both $x(n)$ and $d(n)$ are jointly scaled so as to have a common BFP representation within each block. This means that, for $n \in Z_i$, $x(n)$ and $d(n)$ are expressed as

$$x(n) = \overline{x}(n).2^{\gamma_i}, \quad d(n) = \overline{d}(n).2^{\gamma_i} \quad (9)$$

where γ_i is the common block exponent for the ith block and is given as

$$\gamma_i = ex_i + S_i \quad (10)$$

where

$$ex_i = \lfloor log_2 M_i \rfloor + 1 \quad (11)$$

and

$$M_i = max\{|x(n)|, |d(n)| \mid n \in Z_i\}. \quad (12)$$

The scaling factor S_i is assigned as per the following algorithm :

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 $S_i = (ex_{i-1} - ex_i + S_{min})$, s.t. $\gamma_i = ex_{i-1} + S_{min}$.
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 $\gamma_i < \gamma_{i-1}$.
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 $S'_{i-1} =$

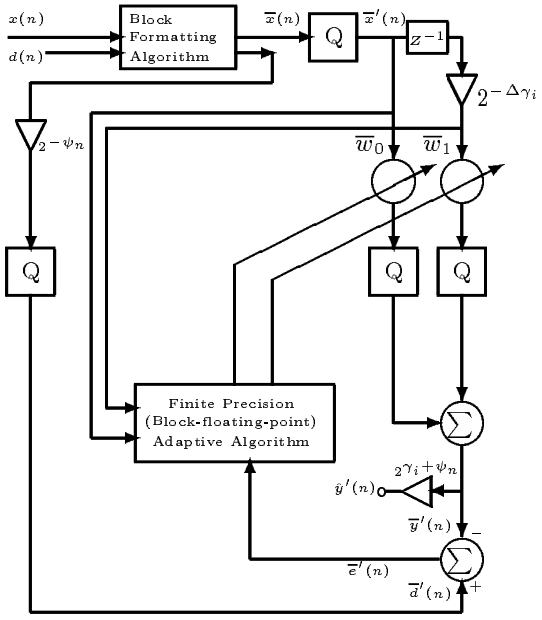

Fig. 2. Block floating point implementation of a 2-tap LMS based adaptive filter.

$S_{i-1} + \Delta\gamma_i$. 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 $\overline{x}(n-j)$, $j = 1, ..., L-1$, through a rescaling unit that applies $\Delta\gamma_i$ number of right or left shifts on $\overline{x}(n-j)$ 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 $n = iN$, $i \in Z$. For a 2-tap filter, this is shown in Fig. 2, where an additional unit 'Q' is used to indicate quantization operation. Also, note that though for the case (A), $\Delta\gamma_i \geq 0$, for (B) and (C), however, $\Delta\gamma_i < 0$, 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. The output error $e(n)$ is then evaluated as $e(n) = \overline{e}(n).2^{\gamma_i+\psi_n}$ where the mantissa $\overline{e}(n)$ is given as

$$\overline{e}(n) = \overline{d}(n).2^{-\psi_n} - \overline{y}(n). \tag{13}$$

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

$$\begin{aligned} |\overline{e}(n)| &\leq |\overline{d}(n)|.2^{-\psi_n} + |\overline{y}(n)| \\ &< 2^{-(S_i+\psi_n)} + \frac{1}{2} \leq \frac{2^{-\psi_n}}{L} + \frac{1}{2} \end{aligned} \tag{14}$$

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

For the above description of $e(n)$, $\mathbf{x}(n)$, $d(n)$ and $\mathbf{w}(n)$, the weight update equation (5) takes the following form:

$$\mathbf{w}(n+1) = \overline{\mathbf{v}}(n).2^{\psi_n} \tag{15}$$

where

$$\overline{\mathbf{v}}(n) = \overline{\mathbf{w}}(n) + \mu\overline{\mathbf{x}}(n)\overline{e}(n).2^{2\gamma_i}. \tag{16}$$

As stated earlier, $\overline{\mathbf{w}}(n+1)$ is required to satisfy $|\overline{w}_k(n+1)| < \frac{1}{2}$ for $k = 0, 1, ..., L-1$, which can be realized in several ways. Our preferred option is to limit $\overline{\mathbf{v}}(n)$ so that $|\overline{v}_k(n)| < 1$, $k = 0, 1, ..., L-1$. Then, if each $\overline{v}_k(n)$ happens to be lying within $\pm\frac{1}{2}$, we make the assignments:

$$\overline{\mathbf{w}}(n+1) = \overline{\mathbf{v}}(n), \quad \psi_{n+1} = \psi_n. \tag{17}$$

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

$$\overline{\mathbf{w}}(n+1) = \frac{1}{2}\overline{\mathbf{v}}(n), \quad \psi_{n+1} = \psi_n + 1. \tag{18}$$

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

$$\mu \leq \frac{2^{-2ex_i}}{2^{-\psi_n+1} + L}. \tag{19}$$

It is easy to verify that the above bound for μ is valid not only when each element of $\overline{\mathbf{x}}(n)$ in (16) comes purely from the i-th block, but also during transition from the (i-1)-th to the i-th block with $ex_i \geq ex_{i-1}$, for which, after necessary rescaling, we have $S'_{i-1} \geq S_i = S_{min}$ implying $|\overline{x}(n-k)| < 2^{-S_i}$ and thus $\overline{y}(n) < \frac{L}{2}.2^{-S_i}$. For $ex_i < ex_{i-1}$, however, the upper bound expression given by eq. (19) 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 $|\overline{x}(n-k)| < 2^{-S'_{i-1}}$ and thus $\overline{y}(n) < \frac{L}{2}.2^{-S'_{i-1}}$, leading to $|\overline{e}(n)| < [2^{-(S'_{i-1}+\psi_n)} + \frac{L}{2}.2^{-S'_{i-1}}]$.

From above, we obtain a general upper bound for μ by equating ψ_n to its lowest value of zero and replacing ex_i by $ex_{max} = max\{ex_i \mid i \in Z\}$ in eq. (19). The general upper bound is given by :

$$\mu \leq \frac{2^{-2ex_{max}}}{L+2}. \tag{20}$$

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

Finally, for practical implementation of $\overline{\mathbf{v}}(n)$ as given in eq. (16), we need to evaluate the product $\mu\overline{x}(n-k)\overline{e}(n)2^{2\gamma_i}$ in such a way that no overflow occurs in any of the intermediate products. This is realized via the following steps :

$step1 \rightarrow \mu.2^{2ex_i} = \mu_1$ (say),

$step2 \rightarrow \mu_1[\overline{x}(n-k)2^{S_i}] = \overline{x}_1(n-k)$ (say), and

$step3 \rightarrow [\overline{x}_1(n-k)\overline{e}(n)].2^{S_i} = \overline{e}_1(n)$ (say).

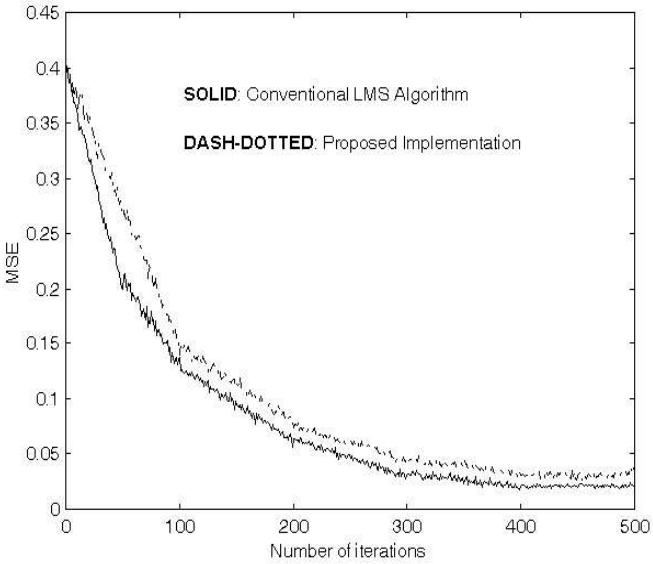

Fig. 3. Learning curves for conventional LMS algorithm and proposed BFP implementation scheme.

Once again, for the block-to-block transitional case with $ex_i < ex_{i-1}$, ex_i in step 1 and S_i in steps 2 and 3 are to be replaced by ex_{i-1} and $S'_{i-1}(= S_{min})$ respectively. It is then easy to check that each of the intermediate products computed in steps 1-3 above has magnitude less than one and thus there is no intermediate overflow.

4. DISCUSSIONS AND CONCLUSIONS

In this paper, we have presented a method for efficient implementation of the LMS algorithm using block floating point arithmetic whose usage in digital filter realization has so far remained limited to fixed coefficient filters only. The proposed scheme requires mostly FxP like operations and thus enjoys computational simplicity, less cost and low power requirement like a FxP based implementation. However, presence of an exponent with every block gives it the additional strength of processing capability over a wide dynamic range as is common with a FP processor. Simulation of the proposed method in finite precision in a variety of contexts has also yielded satisfactory results. We present here the simulation results for a system identification problem, where the signal $y(n)$ observed at a system output was generated as $y(n) = x(n)+0.65x(n-1)+0.25x(n-2)+\nu(n)$, with $x(n)$: system input with variance $\sigma_x^2 = 1$ and $\nu(n)$: observation noise (white) with variance $\sigma_\nu^2 = 0.01$. To decide about the value for μ, we observed that $M_{max} = max\{x(n), y(n)|n \in Z\}$ can be safely taken as $\pm 1.99\sigma_x$ so as to contain almost 95% of the samples of $x(n)$ and $y(n)$. This gives rise to $ex_{max} = 1$ and with $L = 3$, we then have $\mu = 0.05$. With this value of μ, the algorithm was first simulated in 'infinite' precision and then in finite precision with 9 (i.e., 1+8) bits for the mantissa and 4 (i.e., 1+3) bits for the exponent. Fig. 3 demonstrates the simulation results by plotting MSE versus number of iterations for both the infinite precision and the finite precision cases. As is clearly seen, the algorithm convergence has been largely unaffected by moving from infinite to finite precision – however, the excess MSE is more in latter due to the presence of numerical errors. Efforts are now underway to analyse the effects of numerical errors on the proposed implementation.

5. REFERENCES

[1] J. H. Wilkinson, Rounding Errors in Algebraic Processes. Englewood Cliffs, NJ: Prentice-Hall, 1963.

[2] K. R. Ralev and P. H. Bauer, "Realization of Block Floating Point Digital Filters and Application to Block Implementations," IEEE Trans. Signal Processing, vol. 47, no. 4, pp. 1076-1086, April 1999.

[3] K. Kalliojärvi and J. Astola, "Roundoff Errors in Block-Floating-Point Systems," IEEE Trans. Signal Processing, vol. 44, no. 4, pp. 783-790, April 1996.

[4] P. H. Bauer, "Absolute Error Bounds for Block Floating Point Direct form Digital Filters," IEEE Trans. Signal Processing, vol. 43, no. 8, pp. 1994-1996, Aug. 1995.

[5] P. Bauer, "Asymptotic behavior of digital filters with block floating point arithmetic," in Proc. 1994 Int. Conf. Accoust., Speech, Signal Processing, Adelaide, Australia, Apr. 1994, pp. III.609-III.612.

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

[7] S. Sridharan and D. Williamson, "Implementation of high order direct form digital filter structures," IEEE Trans. Circuits Syst., vol. CAS-33, pp. 818-822, Aug. 1986.

[8] D. Williamson, S. Sridharan and P. G. McCrea, "A new approach to block floating point arithmetic in recursive digital filters," IEEE Trans. Circuits Syst., vol. CAS-32, pp. 719-722, July 1985.

[9] F. J. Taylor, "Block Floating Point Distributed Filters," IEEE Trans. Circuits Syst., vol. CAS-31, pp. 300-304, Mar. 1984.

[10] A. V. Oppenheim, "Realization of digital filters using block floating point arithmetic," IEEE Trans. Audio Electroaccoust., vol. AE-18, no. 2, pp. 130-136, June 1970.

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