

A New Fast Householder-Based Fractionally-Spaced FIR MMSE-DFE Computation Algorithm and its Real-Time Implementation

K. M. Zahidul Islam, Naofal Al-Dhahir, Russell McKown, Robert Dawes, and Christopher Stillo

Abstract—We present a new parallel algorithm for computing the optimum coefficients of the *fractionally-spaced* FIR MMSE-DFE based on the Householder orthogonal transformation which avoids the back-substitution computational bottleneck in [1]. The resulting significant reductions in the algorithm's execution time are demonstrated through a real-time FPGA implementation.

Index Terms—MMSE-DFE, fractionally-spaced, QR decomposition, householder transformation.

I. INTRODUCTION

THE finite-impulse-response (FIR) minimum mean-squared error (MMSE) decision feedback equalizer (DFE) is a widely-used single-carrier equalization structure as it outperforms linear equalization while significantly reducing the detection complexity compared to maximum-likelihood sequence detection. Its applications include single-carrier digital TV receivers, high-speed backplane serial links, IEEE 802.11b WLAN, and 2G/3G digital cellular phones, among others. For a robust digital implementation in the presence of timing errors, the feedforward filter (FFF) of the DFE is implemented as a fractionally-spaced FIR filter.

In [1], a fast DFE computation algorithm based on the generalized Schur Algorithm was proposed. This algorithm first calculates the feedback filter (FBF) using Cholesky decomposition of a channel-dependent correlation matrix and then calculates the FFF by back-substitution. In [2], the Cholesky decomposition was formulated as a QR factorization using Givens rotations (GR) to parallelize the MMSE-DFE computation and improve its robustness to finite-precision effects. However, the back-substitution step was still a computational bottleneck which increased the overall algorithm execution time causing performance degradation especially for rapidly-changing channels. Based on the QR approach in [2], it was pointed out in [3] that it is possible to obtain the FFF and FBF simultaneously using the same GR and, hence, the back-substitution is not required. But the algorithm in [3] only applies to a symbol-spaced DFE and the extension to the fractionally-spaced case is not straightforward.

In this letter, we make the following 3 novel contributions

Manuscript received May 13, 2010. The associate editor coordinating the review of this letter and approving it for publication was H. Shafiq.

K. M. Zahidul Islam and N. Al-Dhahir are with the Department of Electrical Engineering, The University of Texas at Dallas, Richardson, TX, USA (e-mail: zislam@student.utdallas.edu, aldhahir@utdallas.edu).

R. McKown is with Advanced Receiver Technologies, Inc. Dallas, TX, USA (e-mail: russell.mckown@gmail.com).

R. Dawes is with QED Corporation, Bedford, TX, USA (e-mail: dawes@wedomath.com).

C. Stillo is with Xilinx, Inc., Dallas, TX, USA (e-mail: cstillo@magicfoo.net).

This work is supported by Advanced Receiver Technologies, Inc. Digital Object Identifier 10.1109/LCOMM.2010.11.100816

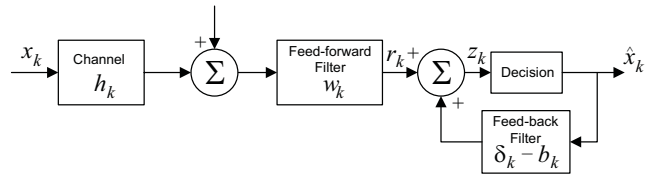


Fig. 1: Block diagram of MMSE-DFE.

- We generalize the algorithm in [3] to the practically-important fractionally-spaced FFF case.
- We reduce the execution time of the algorithm in [3] by designing a new algorithm based on Householder transformation (HT) instead of GR.
- We implement the proposed algorithm in real-time (for a system with symbol duration $T = 0.26\mu\text{s}$) and demonstrate a 50% reduction in the execution time for a $\frac{T}{2}$ -spaced FIR MMSE-DFE compared to the algorithm proposed in [1].

This letter is organized as follows. In Section II, we present the fractionally-spaced FIR MMSE-DFE structure and describe the input-output model and assumptions. In Section III, first the QR-based approach for computing the DFE coefficients is briefly reviewed. Second, it is shown how to eliminate the back-substitution step in computing the fractionally-spaced FFF. Third, the new HT-based algorithm is presented and illustrated with an example. Finally, complexity comparisons based on real-time FPGA implementations of the proposed algorithm and the one in [1] are given in Section IV followed by conclusions.

II. MODEL AND ASSUMPTIONS

The structure of the discrete-time MMSE-DFE is shown in Fig. 1. Let the continuous-time channel output be sampled at some positive integer multiple l of the symbol rate $1/T$. Following the notation in [1], by collecting l such samples into vectors, we write the sampled channel output as

$$\mathbf{y}_k = \sum_{m=0}^{\nu} \mathbf{h}_m x_{k-m} + \mathbf{n}_k \quad (1)$$

where x_k , \mathbf{h}_k , ν and \mathbf{n}_k denote the k -th input symbol, k -th channel impulse response (CIR) coefficient vector (of size $l \times 1$), channel memory and k -th noise symbol vector, respectively. The complex input and noise sequences $\{x_k\}$ and $\{\mathbf{n}_k\}$ are zero-mean and uncorrelated with variances \mathcal{E}_x and σ_n^2 , respectively. Collecting N_f output symbols, we write (1) as

$$\mathbf{y}_{k+N_f-1:k} = \mathbf{H}\mathbf{x}_{k+N_f-1:k-\nu} + \mathbf{n}_{k+N_f-1:k} \quad (2)$$

where \mathbf{H} is the $N_f l \times (N_f + \nu)$ block Toeplitz channel matrix.

III. THE FRACTIONALLY-SPACED FIR MMSE-DFE

The fractionally-spaced MMSE-DFE computation algorithm calculates two FIR filters: a length- N_f symbol-spaced FBF \mathbf{b} and a fractionally-spaced length- $N_f l$ FFF \mathbf{w} . We consider \mathbf{b} and \mathbf{w} as column vectors in this letter. For convenience, define the augmented vector $\tilde{\mathbf{b}} \triangleq \begin{bmatrix} \mathbf{0}_{1 \times \Delta} & 1 & \mathbf{b}^H \end{bmatrix}^H$ where $(\cdot)^H$ denotes the Hermitian operator, $\mathbf{0}$ is the all-zero vector and Δ ($0 \leq \Delta \leq N_f + \nu - 1$) is the decision delay parameter. Consider the Cholesky factorization

$$\mathbf{C} \triangleq \frac{1}{SNR'} \mathbf{I}_{N_f + \nu} + \mathbf{H}^H \mathbf{H} \triangleq \mathbf{R}^H \mathbf{R} \quad (3)$$

where \mathbf{I} is the identity matrix, \mathbf{R} is a upper-triangular matrix and $SNR' = \frac{\epsilon_{\sigma}}{\sigma_n^2}$. The optimum $\tilde{\mathbf{b}}$ and \mathbf{w} are given by [2]

$$\tilde{\mathbf{b}} = \frac{\mathbf{R}^H \mathbf{e}_{(\Delta_{\text{opt}}+1)}}{R(\Delta_{\text{opt}}+1, \Delta_{\text{opt}}+1)}, \quad \mathbf{w}^H = \frac{\mathbf{e}_{(\Delta_{\text{opt}}+1)}^H \mathbf{R}^{-H} \mathbf{H}^H}{\sigma_n^2 l R(\Delta_{\text{opt}}+1, \Delta_{\text{opt}}+1)} \quad (4)$$

where \mathbf{e}_i and $R(i, i)$ denote i -th unit column vector and the (i, i) element of \mathbf{R} , respectively. Δ_{opt} is the optimized decision delay which maximizes the unbiased decision point SNR given by $SNR_{\text{MMSE-DFE,U}} = \frac{|R(\Delta_{\text{opt}}+1, \Delta_{\text{opt}}+1)|^2}{\sigma_n^2 l} - 1$. For the case $N_b = \nu$ assumed in this letter, it was shown in [1] that $\Delta_{\text{opt}} = N_f - 1$. The general case was treated in [4].

A. Eliminating the Back-Substitution Bottleneck

The Cholesky factorization of \mathbf{C} in Equation (3) was formulated as a QR decomposition in [2] as follows

$$\mathbf{Q} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{SNR'}} \mathbf{I}_{N_f + \nu} \\ \mathbf{H} \end{bmatrix}}_{\tilde{\mathbf{H}}} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0}_{(N_f l) \times (N_f + \nu)} \end{bmatrix} \quad (5)$$

The complexity of the QR factorization can be reduced significantly based on the following observations

- As it can be seen from (4), we have to compute the $(\Delta_{\text{opt}} + 1)$ column of \mathbf{R}^H . Hence, we have to apply $(\Delta_{\text{opt}} + 1)$ unitary transformations (UT) on $\tilde{\mathbf{H}}$.
- Let $\mathbf{S}_i = \begin{bmatrix} \frac{1}{\sqrt{SNR'}} \mathbf{I}_i^T & \mathbf{H}_{il}^T & \cdots & \mathbf{H}_{i(l+1)}^T \end{bmatrix}^T : i = 0, 1, \dots, \Delta_{\text{opt}}$ denote a set of rows where \mathbf{I}_j and \mathbf{H}_j are the j -th row of \mathbf{I} and \mathbf{H} , respectively and $(\cdot)^T$ denotes the transpose. Since $\tilde{\mathbf{H}}$ is a block-Toeplitz matrix, the \mathbf{S}_i 's are identical except for a right shift by i positions. Hence, we can apply the same UT to all \mathbf{S}_i 's.
- Since $\tilde{\mathbf{H}}$ is a *banded* upper-triangular matrix with a band size of $\nu + 1$ rows, in each step, we only need to apply the UT to $\nu + 1$ rows.

It was shown in [3] that by applying the same \mathbf{Q} in Equation (5), we can compute \mathbf{w} in Equation (4) from the $(\Delta_{\text{opt}} + 1)$ row of \mathbf{F}_t which is obtained as

$$\mathbf{Q} \begin{bmatrix} \sqrt{SNR'} \mathbf{H}^H \\ \mathbf{0}_{(N_f l) \times (N_f l)} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_t \\ \mathbf{F}_l \end{bmatrix}$$

However, the algorithm in [3] is only applicable to a symbol-spaced FFF which limits its application in practice.

B. New Householder-Based Fractionally-Spaced DFE

The UT operator \mathbf{Q} is computed using the Gram-Schmidt procedure, GR or HT. The Gram-Schmidt procedure is numerically sensitive to rounding and, therefore, is not suitable for fixed-point implementation. In general, for fractionally-spaced channel estimates, to zero out each length- $(l + 1)$ channel coefficient vector \mathbf{h}_i , we need to apply l GR compared to a single HT operation. It is shown in [5] that for an $m \times n$ ($m \geq n$) matrix, it takes $2n^2(m - n/3)$ floating point operations (FLOPs) to obtain the upper-triangular \mathbf{R} matrix using HT compared to the $3n^2(m - n/3)$ FLOPs required by the GR. Hence, we choose the HT for our real-time implementation.

Given SNR , N_f , ν and l , we start by defining the initial matrices \mathbf{B}_0 and \mathbf{W}_0 . Let, $h_{in}, n = 1, \dots, l$ denote the n -th sample of i -th CIR vector coefficient $\mathbf{h}_i, i = 0, \dots, \nu$. We need to apply $\Delta_{\text{opt}} + 1$ iterations on \mathbf{B}_0 and \mathbf{W}_0 to compute \mathbf{b} and \mathbf{w} . Let \mathbf{B}_i and \mathbf{W}_i denote the updated \mathbf{B}_0 and \mathbf{W}_0 matrices at i -th iteration, respectively. In addition, let \mathbf{b}_k^i and \mathbf{w}_k^i denote the k -th column of \mathbf{B}_i and \mathbf{W}_i in the i -th iteration, respectively. In summary, we propose the following algorithm

- **Initial condition:** Initialize \mathbf{B}_0 and \mathbf{W}_0 according to Equation (6).
- **Iterations:** $i = 1, \dots, \Delta_{\text{opt}} + 1$
 - Compute UT operator \mathbf{Q}_i as follows

$$\mathbf{v} = \mathbf{b}_0^{(i-1)} + \text{sign}(b_{00}^{(i-1)}) \|\mathbf{b}_0^{(i-1)}\| \mathbf{e}_1$$

$$\mathbf{Q}_i = \mathbf{I}_{l+1} - 2 \frac{\mathbf{v} \mathbf{v}^H}{\mathbf{v}^H \mathbf{v}}$$

- Update: $\mathbf{B}_i = \mathbf{Q}_i \mathbf{B}_{i-1}, \mathbf{W}_i = \mathbf{Q}_i \mathbf{W}_{i-1}$.
- Unless $i = \Delta_{\text{opt}} + 1$, linearly shift the first column of \mathbf{B}_i and the first l columns of \mathbf{W}_i to the left by 1 and l positions, respectively, *except* the first row of each matrix and append $(l \times 1)$ and $(l \times l)$ zero matrices to the right end, respectively.

- **Output :** Compute \mathbf{b} and \mathbf{w} as follows

$$\mathbf{b} = \frac{1}{b_{00}^{(\Delta_{\text{opt}}+1)}} \begin{bmatrix} b_{01}^{(\Delta_{\text{opt}}+1)} & \cdots & b_{0\nu}^{(\Delta_{\text{opt}}+1)} \end{bmatrix}^H$$

$$\mathbf{w}^H = \frac{1}{b_{00}^{(\Delta_{\text{opt}}+1)}} \begin{bmatrix} w_{01}^{(\Delta_{\text{opt}}+1)} & \cdots & w_{0(N_f l - 1)}^{(\Delta_{\text{opt}}+1)} \end{bmatrix}$$

We illustrate our proposed fast algorithm with an example in Fig. 2 for $N_f = 4, \nu = 2$ and $l = 2$. We chose $N_b = \nu = 2$ and; therefore, $\Delta_{\text{opt}} = N_f - 1 = 3$. The general algorithm flowchart is given in Fig. 3.

IV. EXPERIMENTAL RESULTS

We have implemented the proposed new algorithm and the algorithm in [1] on the Xilinx Virtex-4 XC4VVSX35 FPGA platform for $N_f = 20, l = 2$, and $N_b = \nu = 19$. We found by simulations that this choice of parameters results in only 0.3 dB performance loss from the matched filter bound at an input SNR level of 20 dB for the considered CIR.

In Table I, we list the maximum number of hardware clocks per design function call, defined as throughput and the total number of function calls per receiver cycle for these 2 algorithms. The receiver cycle in this implementation is defined as $256T$ where $T = \frac{1}{3.84\text{MHz}} = 0.26\mu\text{s}$. At 3840

$$\mathbf{B}_0 \triangleq \underbrace{\begin{bmatrix} 1/\sqrt{SNR'} & 0 & \cdots & 0 \\ h_{01} & h_{11} & \cdots & h_{\nu 1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{0l} & h_{1l} & \cdots & h_{\nu l} \end{bmatrix}}_{(l+1) \times (\nu+1)}, \mathbf{W}_0 \triangleq \underbrace{\begin{bmatrix} 0 & \cdots & 0 & \sqrt{SNR'}h_{\nu 1}^* & \cdots & \sqrt{SNR'}h_{\nu l}^* & \cdots & \sqrt{SNR'}h_{01}^* & \cdots & \sqrt{SNR'}h_{0l}^* \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}}_{(l+1) \times (N_f l)} \quad (6)$$

Initialization:	$\mathbf{B}_0 \triangleq \begin{bmatrix} 1/\sqrt{SNR'} & 0 & 0 \\ h_{01} & h_{11} & h_{21} \\ h_{02} & h_{12} & h_{22} \end{bmatrix}$	$\mathbf{W}_0 \triangleq \begin{bmatrix} 0 & 0 & \sqrt{SNR'}h_{21}^* & \sqrt{SNR'}h_{22}^* & \sqrt{SNR'}h_{11}^* & \sqrt{SNR'}h_{12}^* & \sqrt{SNR'}h_{01}^* & \sqrt{SNR'}h_{02}^* \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
Iter. 1- Update:	$\mathbf{B}_1 = \mathbf{Q}_1 \mathbf{B}_0 = \begin{bmatrix} b_{00}^{(1)} & b_{01}^{(1)} & b_{02}^{(1)} \\ 0 & b_{11}^{(1)} & b_{12}^{(1)} \\ 0 & b_{21}^{(1)} & b_{22}^{(1)} \end{bmatrix}$	$\mathbf{W}_1 = \mathbf{Q}_1 \mathbf{W}_0 = \begin{bmatrix} 0 & 0 & w_{02}^{(1)} & w_{03}^{(1)} & w_{04}^{(1)} & w_{05}^{(1)} & w_{06}^{(1)} & w_{07}^{(1)} \\ 0 & 0 & w_{12}^{(1)} & w_{13}^{(1)} & w_{14}^{(1)} & w_{15}^{(1)} & w_{16}^{(1)} & w_{17}^{(1)} \\ 0 & 0 & w_{22}^{(1)} & w_{23}^{(1)} & w_{24}^{(1)} & w_{25}^{(1)} & w_{26}^{(1)} & w_{27}^{(1)} \end{bmatrix}$	
Iter. 1- Shift:	$\mathbf{B}_1 = \begin{bmatrix} b_{00}^{(1)} & b_{01}^{(1)} & b_{02}^{(2)} \\ b_{11}^{(1)} & b_{12}^{(1)} & 0 \\ b_{21}^{(1)} & b_{22}^{(1)} & 0 \end{bmatrix}$	$\mathbf{W}_1 = \begin{bmatrix} 0 & 0 & w_{02}^{(1)} & w_{03}^{(1)} & w_{04}^{(1)} & w_{05}^{(1)} & w_{06}^{(1)} & w_{07}^{(1)} \\ w_{12}^{(1)} & w_{13}^{(1)} & w_{14}^{(1)} & w_{15}^{(1)} & w_{16}^{(1)} & w_{17}^{(1)} & 0 & 0 \\ w_{22}^{(1)} & w_{23}^{(1)} & w_{24}^{(1)} & w_{25}^{(1)} & w_{26}^{(1)} & w_{27}^{(1)} & 0 & 0 \end{bmatrix}$	
Iter. 2- Update:	$\mathbf{B}_2 = \mathbf{Q}_2 \mathbf{B}_1 = \begin{bmatrix} b_{00}^{(2)} & b_{01}^{(2)} & b_{02}^{(2)} \\ 0 & b_{11}^{(2)} & b_{12}^{(2)} \\ 0 & b_{21}^{(2)} & b_{22}^{(2)} \end{bmatrix}$	$\mathbf{W}_2 = \mathbf{Q}_2 \mathbf{W}_1 = \begin{bmatrix} w_{00}^{(2)} & w_{01}^{(2)} & w_{02}^{(2)} & w_{03}^{(2)} & w_{04}^{(2)} & w_{05}^{(2)} & w_{06}^{(2)} & w_{07}^{(2)} \\ w_{10}^{(2)} & w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} & w_{14}^{(2)} & w_{15}^{(2)} & w_{16}^{(2)} & w_{17}^{(2)} \\ w_{20}^{(2)} & w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(2)} & w_{24}^{(2)} & w_{25}^{(2)} & w_{26}^{(2)} & w_{27}^{(2)} \end{bmatrix}$	
Iter. 2- Shift:	$\mathbf{B}_2 = \begin{bmatrix} b_{00}^{(2)} & b_{01}^{(2)} & b_{02}^{(3)} \\ b_{11}^{(2)} & b_{12}^{(2)} & 0 \\ b_{21}^{(2)} & b_{22}^{(2)} & 0 \end{bmatrix}$	$\mathbf{W}_2 = \begin{bmatrix} w_{00}^{(2)} & w_{01}^{(2)} & w_{02}^{(2)} & w_{03}^{(2)} & w_{04}^{(2)} & w_{05}^{(2)} & w_{06}^{(2)} & w_{07}^{(2)} \\ w_{12}^{(2)} & w_{13}^{(2)} & w_{14}^{(2)} & w_{15}^{(2)} & w_{16}^{(2)} & w_{17}^{(2)} & 0 & 0 \\ w_{22}^{(2)} & w_{23}^{(2)} & w_{24}^{(2)} & w_{25}^{(2)} & w_{26}^{(2)} & w_{27}^{(2)} & 0 & 0 \end{bmatrix}$	
Iter. 3- Update:	$\mathbf{B}_3 = \mathbf{Q}_3 \mathbf{B}_2 = \begin{bmatrix} b_{00}^{(3)} & b_{01}^{(3)} & b_{02}^{(3)} \\ 0 & b_{11}^{(3)} & b_{12}^{(3)} \\ 0 & b_{21}^{(3)} & b_{22}^{(3)} \end{bmatrix}$	$\mathbf{W}_3 = \mathbf{Q}_3 \mathbf{W}_2 = \begin{bmatrix} w_{00}^{(3)} & w_{01}^{(3)} & w_{02}^{(3)} & w_{03}^{(3)} & w_{04}^{(3)} & w_{05}^{(3)} & w_{06}^{(3)} & w_{07}^{(3)} \\ w_{10}^{(3)} & w_{11}^{(3)} & w_{12}^{(3)} & w_{13}^{(3)} & w_{14}^{(3)} & w_{15}^{(3)} & w_{16}^{(3)} & w_{17}^{(3)} \\ w_{20}^{(3)} & w_{21}^{(3)} & w_{22}^{(3)} & w_{23}^{(3)} & w_{24}^{(3)} & w_{25}^{(3)} & w_{26}^{(3)} & w_{27}^{(3)} \end{bmatrix}$	
Iter. 3- Shift:	$\mathbf{B}_3 = \begin{bmatrix} b_{00}^{(3)} & b_{01}^{(3)} & b_{02}^{(4)} \\ b_{11}^{(3)} & b_{12}^{(3)} & 0 \\ b_{21}^{(3)} & b_{22}^{(3)} & 0 \end{bmatrix}$	$\mathbf{W}_3 = \begin{bmatrix} w_{00}^{(3)} & w_{01}^{(3)} & w_{02}^{(3)} & w_{03}^{(3)} & w_{04}^{(3)} & w_{05}^{(3)} & w_{06}^{(3)} & w_{07}^{(3)} \\ w_{12}^{(3)} & w_{13}^{(3)} & w_{14}^{(3)} & w_{15}^{(3)} & w_{16}^{(3)} & w_{17}^{(3)} & 0 & 0 \\ w_{22}^{(3)} & w_{23}^{(3)} & w_{24}^{(3)} & w_{25}^{(3)} & w_{26}^{(3)} & w_{27}^{(3)} & 0 & 0 \end{bmatrix}$	
Iter. 4- Update:	$\mathbf{B}_4 = \mathbf{Q}_4 \mathbf{B}_3 = \begin{bmatrix} b_{00}^{(4)} & b_{01}^{(4)} & b_{02}^{(4)} \\ 0 & b_{11}^{(4)} & b_{12}^{(4)} \\ 0 & b_{21}^{(4)} & b_{22}^{(4)} \end{bmatrix}$	$\mathbf{W}_4 = \mathbf{Q}_4 \mathbf{W}_3 = \begin{bmatrix} w_{00}^{(4)} & w_{01}^{(4)} & w_{02}^{(4)} & w_{03}^{(4)} & w_{04}^{(4)} & w_{05}^{(4)} & w_{06}^{(4)} & w_{07}^{(4)} \\ w_{10}^{(4)} & w_{11}^{(4)} & w_{12}^{(4)} & w_{13}^{(4)} & w_{14}^{(4)} & w_{15}^{(4)} & w_{16}^{(4)} & w_{17}^{(4)} \\ w_{20}^{(4)} & w_{21}^{(4)} & w_{22}^{(4)} & w_{23}^{(4)} & w_{24}^{(4)} & w_{25}^{(4)} & w_{26}^{(4)} & w_{27}^{(4)} \end{bmatrix}$	
Compute \mathbf{b} and \mathbf{w} :	$\mathbf{b} = \frac{1}{b_{00}^{(4)}} \begin{bmatrix} b_{01}^{(4)} & b_{02}^{(4)} \end{bmatrix}^H$	$\mathbf{w}^H = \frac{1}{b_{00}^{(4)}} \begin{bmatrix} w_{00}^{(4)} & w_{01}^{(4)} & w_{02}^{(4)} & w_{03}^{(4)} & w_{04}^{(4)} & w_{05}^{(4)} & w_{06}^{(4)} & w_{07}^{(4)} \end{bmatrix}$	

Fig. 2: Illustration of the proposed fast fractionally-spaced FIR MMSE-DFE computation algorithm.

TABLE I: Real-time complexity comparison

	[1]	Proposed algorithm
Maximum Throughput (MT)	6	11
No. of function calls per receiver cycle (C)	1575	565
Total clocks (TC=MT×C)	9450	6215

clocks per receive cycle, the new algorithm takes 2 receive cycles while it takes 3 cycles for the algorithm in [1] to compute the DFE coefficients resulting in a 50% reduction in the algorithm's execution time.

V. CONCLUSIONS

We presented a new algorithm for computing the feed-forward and feedback coefficients of the FIR fractionally-spaced MMSE-DFE by applying the same Householder transformation, thus eliminating the back-substitution computational bottleneck in [1]. Our real-time hardware measurements demonstrate a 50% reduction in the algorithm's execution time compared to [1] making it the fastest fractionally-spaced FIR MMSE-DFE coefficient computation algorithm in the literature, to the best of our knowledge.

REFERENCES

[1] N. Al-Dhahir and J. Cioffi, "Fast computation of channel-estimate based equalizers in packet data transmission," *IEEE Trans. Signal Process.*, vol. 43, no. 11, pp. 2462–2473, Nov. 1995.

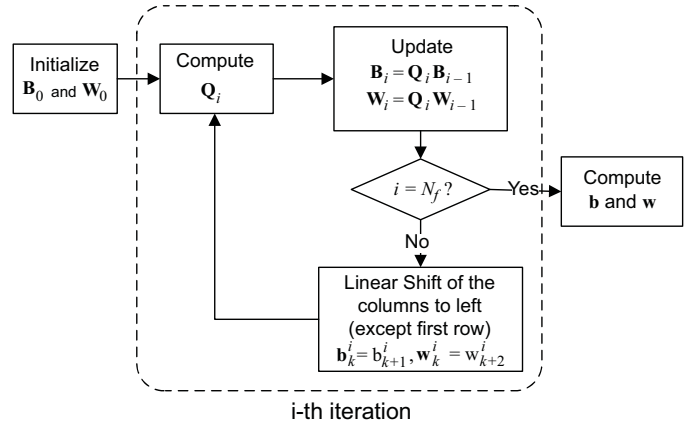


Fig. 3: Flowchart of the proposed algorithm.

- [2] N. Al-Dhahir and A. H. Sayed, "CORDIC-based MMSE-DFE coefficient computation," *Digital Signal Processing*, vol. 9, no. 3, pp. 178–194, 1999.
- [3] B. Yang, "Method for channel equalization," Patent US 6,954,509 B2, Oct. 11, 2005.
- [4] N. Al-Dhahir and J. Cioffi, "Efficient computation of the delay-optimized finite length MMSE-DFE," *IEEE Trans. Signal Process.*, vol. 44, no. 5, pp. 1288–1292, May 1996.
- [5] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd edition. Baltimore, MD: Johns Hopkins University Press, 1996.