A New Pipelined Adaptive DFE Architecture with Improved Convergence Rate

Meng-Da Yang and An-Yeu (Andy) Wu
Graduate Institute of Electronics Engineering,
National Taiwan University, Taipei 106, Taiwan, R.O.C

Abstract

Among existing works of high-speed pipelined Adaptive Decision Feedback Equalizer (ADFE), the pipelined ADFE using Relaxed Look-ahead technique results in a substantial hardware saving than the parallel processing or Look-ahead approaches. However, it suffers from the SNR degradation and slow convergence rate. In this paper, a new pipelining ADFE architecture is developed. By employing the Post-processing Filter (PF) to cancel the most significant post-cursor Inter-Symbol Interference (ISI) terms, the proposed ADFE architecture can help to improve the convergence rate of ADFE compared with the Relaxed Look-ahead ADFE architecture while at the same hardware cost. It provides an alternative approach for the design of high-seed pipelining ADFE with large speedup factor.

I. Introduction

Adaptive Decision Feedback Equalizer (ADFE) using *Least Mean-Squared (LMS)* algorithm is a well-known equalization technique for magnetic storage and digital communication. However, the fine-grain pipelining of the ADFE is known to be a difficult problem for high-speed applications. This is due to the *Decision Feedback Loop (DFL)*. According to the Iteration Bound [1], the smallest clock period of ADFE is bounded by the DFL. Several approaches are proposed to solve aforementioned problems. For example, pipelining the ADFE can be achieved by precomputing all possible in DFL to open the DFL [2] . However, it results in a large hardware overhead as it transforms a serial algorithm into an equivalent (in the sense of input-output behavior) pipelined algorithm. Another algorithm is proposed in [3] . It maintains the functionality instead of input-output behavior by *Relaxed Look-ahead.* Although the hardware overhead in this algorithm is small, it suffers from some performance degradation such as output SNR and convergence rate. Nevertheless, from VLSI implementation point of view, the second approach is suited for a low cost design. In this paper, we propose an alternative architecture to improve the convergence rate of the Relaxed Look-ahead ADFE. The simulation result shows that we the proposed ADFE architecture can converge much faster than the Relaxed Look-ahead ADFE given the same SNR requirement, whereas the hardware complexity of both architectures are the same.

II. The Proposed Pipelined ADFE Architecture

In this section, we will introduce two pipelined ADFE architectures. The first algorithm, referred to as *Pipelined ADFE1 (PIPEADFE1)* [3] , is an extension of the ADFE algorithm proposed in [4] . Another architecture, referred to as *Pipelined ADFE2 (PIPEADFE2),* is our proposed architecture in this paper. The channel we consider in this paper is as follows. $x(n) = h(n) \otimes a(n) + v(n)$, where $a(n)$ is the channel input at time instance n, $h(n)$ is the channel coefficient vector. $v(n)$ is white Gaussian noise and $x(n)$ is the received sample.

A. Review of Pipelined ADFE Architecture

In [3] , the Delayed LMS [5] and the technique of Transfer Delay Relaxation [6] are employed to develop the PIPEADFE1. Then, *Sum Relaxation* [3] is applied to pipeline the updating circuit of ADFE. The equations describing the PIPEADFE1 (see Fig.1.) are summarized below.

$$\tilde{a}(n) = a_F(n) + a_B(n) , \tag{1a}$$

$$\mathbf{X}(n) = [x(n) \ldots\ldots x(n-N_f+1)], \tag{1b}$$

$$\mathbf{Y}(n) = [\hat{a}(n-1-D_1) \ldots\ldots \hat{a}(n-D_1-N_b)], \tag{1c}$$

$$a_F(n) = \mathbf{C}^T(n-D_4)\mathbf{X}(n), \tag{1d}$$

$$a_B(n) = \mathbf{D}^T(n-D_4)\mathbf{Y}(n), \tag{1e}$$

$$e(n) = \hat{a}(n) - \tilde{a}(n), \tag{1f}$$

$$\hat{a}(n) = Q[\tilde{a}(n)], \tag{1g}$$

$$\mathbf{C}(n) = \mathbf{C}(n-D_4) + \mu \sum_{i=0}^{M-1} e(n-D_2-i)\mathbf{X}(n-D_2-i), \tag{1h}$$

$$\mathbf{D}(n) = \mathbf{D}(n-D_4) + \mu \sum_{i=0}^{M-1} e(n-D_3-i)\mathbf{Y}(n-D_3-i), \tag{1i}$$

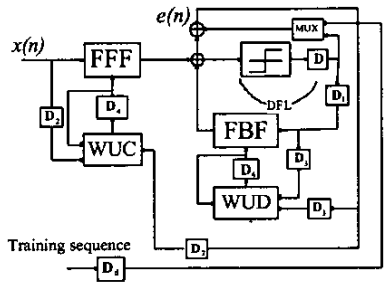

Fig.1. PIPEADFE1 architecture.

where $a_F(n)$ is the output of FFF, $a_B(n)$ is the output of FBF, $\mathbf{C}(n)$ is the vector of FFF coefficients, $\mathbf{D}(n)$ is the vector of FBF coefficients, $\mathbf{X}(n)$ is the vector of received samples, $\mathbf{Y}(n)$ is the vector of detected symbols, $a(n)$ is the input to $Q(\bullet)$, and $\hat{a}(n)$ is the quantizer decision.

B. New Architecture of Pipelined Adaptive DFE (PIPEADFE2)

In [3] , it is shown that the convergence rate of PIPEADFE1 is quiet slower than the *Serial ADFE (Non-pipelined ADFE)*. In this paper, we develop a new architecture of pipelined adaptive DFE (PIPEADFE2) to improve the convergence rate. In the case of PIPEADFE1, the FBF cannot cancel the first D_1 post-cursor ISI terms and the burden of canceling them falls on the FFF. Therefore, we apply some post-processing scheme (PF) into PIPEADFE1 to cancel the first D_1 post-cursor ISI terms instead of using FFF. Hence, the burden of FFF can be alleviated. That means the number of taps in FFF can be reduced. The overall block diagram of the proposed PIPEADFE2 is shown in Fig.2.

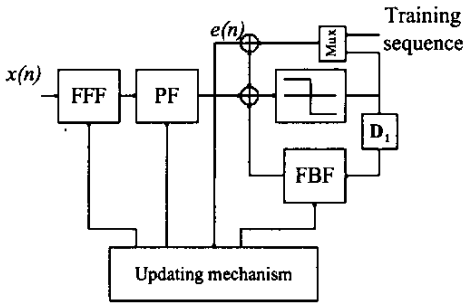

Fig.2. Overall block diagram of the PIPEADFE2.

The derivation of our proposed algorithm is as follows. We first show an example of PIPEADFE2 with a speedup factor of three. That is, iteration bound of serial ADFE is three times than the PIPEADFE2. Assume that the transmitted data $a(n)$ is an independent sequence, and input data of receiver are $x(n)$.

$$x(n) = h_{-1}a(n+1) + h_0 a(n) + h_1 a(n-1) + h_2 a(n-2) + v(n), \tag{2}$$

where $h_{-1}, h_0, \ldots\ldots$ are channel impulse response, $v(n)$ is the AWGN. The number of taps in FFF and FBF are three and two, respectively. Hence the number of taps in PF is three in this example. The coefficients of FFF, PF and FBF at time instance n are denoted as c_i, p_i and b_i, respectively. Then, the prediction error $e(n)$ can be expressed as

$$\begin{aligned} e(n) &= a(n) - F(n) - P(n) \\ &= a(n) - F(n) - p_1 F(n-1) - p_2 F(n-2) - B(n) \end{aligned} \tag{3a}$$

$$F(n) = \sum_{i=0}^{1} c_i x(n+i), \tag{3b}$$

$$B(n) = \sum_{i=1}^{2} b_i a(n-2-i), \tag{3c}$$

where c_i $i = 0 \sim 2$, is the coefficient of i-th taps in FFF, $F(n)$ is the output of FFF, $B(n)$ is the output of FBF, p_i is coefficient of i-th tap in PF. $P(n)$ denotes the total effect of PF and FBF in time instance n, and can be written as

$$\begin{aligned} P(n) &= p_1 F(n-1) + p_2 F(n-2) + B(n) \\ &= \sum_{i=-2}^{4} s_i a(n-i) + r(n). \end{aligned} \tag{4}$$

where $r(n)$ is noise component

In serial ADFE, the objective of the FFF is to minimize $\mathrm{E}\{ e^2(n) \}$. In the FFF of the PIPEADFE2, we intend to minimize $e^2(n)$ instead of $\mathrm{E}\{ e^2(n) \}$. In order to apply stochastic gradient-based algorithm, we must find the gradient of this cost function. Moreover, the total effect of PF and FBF in time instance n can be considered as a constant. Hence, the gradients corresponding to c_i are listed below:

$$\frac{\partial e(n)^2}{\partial c_0} = -2e(n)x(n), \tag{5a}$$

$$\frac{\partial e(n)^2}{\partial c_1} = -2e(n)x(n+1), \tag{5b}$$

$$\frac{\partial e(n)^2}{\partial c_2} = -2e(n)x(n+2), \tag{5c}$$

The results listed above are similar to the serial ADFE case. Hence, the main functionality of the FFF in PIPEADFE2 is the cancellation of precursor ISI terms. In PIPEADFE2, we employ a dedicated PF to decorrelated the correlation between the first two post-cursor ISI terms and ADFE output. The rest ISI terms are canceled by FFF and FBF. Let us consider the PF and FBF. The FFF output at time instance n can written as

$$F(n) = \sum_{i=0}^{2} c_i x(n+i) = \sum_{i=-3}^{2} r_i a(n-i). \tag{6}$$

$F(n)$ represents sum of the residual ISI terms that cannot be canceled by FFF at time instance n. Recall that, in the serial ADFE algorithm, the prediction error must be orthogonal to the observations in the steady state. This is so-called "*Orthogonal Principle*" in the literature of adaptive signal processing [7] . It implies that minimizing the prediction error is equivalent to decorrelate the correlation between observations and filter output. Therefore, the objective of PF is to minimize the following two expectation terms of

$$\min_{p_1} \{\mathrm{E}^2\{e(n)a(n-1)\}\}, \tag{7a}$$

$$\min_{p_2} \{\mathrm{E}^2\{e(n)a(n-2)\}\}, \tag{7b}$$

where p_1, p_2 are the coefficients of postprocessing filter.
Next, the gradients corresponding to these costs functions are

$$\frac{\partial \mathbf{E}^2\{a(n-1)e(n)\}}{\partial p_1} = 2r_0(r_1 + p_1 r_0 + p_2 r_{-1})\mathbf{E}^2\{a^2(n-1)\} \quad (8a)$$
$$= -2r_0\mathbf{E}\{e(n)a(n-1)\}\mathbf{E}\{a^2(n-1)\},$$
$$\frac{\partial \mathbf{E}^2\{a(n-2)e(n)\}}{\partial p_2} = 2r_0(r_2 + p_1 r_1 + p_2 r_0)\mathbf{E}^2\{a^2(n-2)\} \quad (8b)$$
$$= -2r_0\mathbf{E}\{e(n)a(n-2)\}\mathbf{E}\{a^2(n-2)\}.$$

Because the direction of gradient is more important than the magnitude of gradient in stochastic gradient-based algorithm, we can approximate the gradient of (8) as

$$\frac{\partial \mathbf{E}\{a^2(n-1)e^2(n)\}}{\partial p_1} \cong -2\mathbf{E}\{e(n)a(n-1)\}, \quad (9a)$$
$$\frac{\partial \mathbf{E}\{a^2(n-2)e^2(n)\}}{\partial p_2} \cong -2\mathbf{E}\{e(n)a(n-2)\}. \quad (9b)$$

In FBF, we intend to minimize the same cost function of FFF. The gradients corresponding to FBF are

$$\frac{\partial \mathbf{E}\{e^2(n)\}}{\partial b_1} = -2\mathbf{E}\{e(n)a(n-3)\}, \quad (10a)$$
$$\frac{\partial \mathbf{E}\{e^2(n)\}}{\partial b_2} = -2\mathbf{E}\{e(n)a(n-4)\}. \quad (10b)$$

The equations we derived for this particular example can be generalized to the general case with arbitrary taps and arbitrary speedup factor. Finally, combining with Delayed-LMS, Sum Relaxed Look-ahead and generalized cases of (5), (9), (10), the equations describing the proposed PIPEADFE2 algorithm are

$$\mathbf{X}(n) = [x(n)..... .....x(n-N_f+1)], \quad (11a)$$
$$\mathbf{Y}(n) = [\hat{a}(n-D_1-1).....\hat{a}(n-D_1-N_b)] \quad (11b)$$
$$\mathbf{Z}(n) = [\hat{a}(n-1) \cdots \cdots \hat{a}(n-D_1)], \quad (11c)$$
$$\mathbf{P}(n) = [p_1(n) \cdots \cdots p_{D_1}(n)], \quad (11d)$$
$$F(n) = \mathbf{C}^T(n-D_4)\mathbf{X}(n), \quad (11e)$$
$$B(n) = \mathbf{D}^T(n-D_4)\mathbf{Y}(n), \quad (11f)$$
$$\bar{a}(n) = \sum_{j=0}^{D_1} p_j(n-D_4)F(n-j) + B(n), p_0 = 1, \quad (11g)$$
$$e(n) = \hat{a}(n) - \bar{a}(n), \quad (11h)$$
$$\hat{a}(n) = Q[\bar{a}(n)], \quad (11i)$$
$$\mathbf{C}(n) = \mathbf{C}(n-D_4) + \mu\sum_{i=0}^{LA-1} e(n-D_2-i)\mathbf{X}(n-D_2-i), \quad (11j)$$
$$\mathbf{D}(n) = \mathbf{D}(n-D_4) + \mu\sum_{i=0}^{LA-1} e(n-D_3-i)\mathbf{Y}(n-D_3-i), \quad (11k)$$
$$\mathbf{P}(n) = \mathbf{P}(n-D_4) + \mu\sum_{i=0}^{LA-1} e(n-D_5-i)\mathbf{Z}(n-D_5-i), \quad (11l)$$

where p_j is the postprecessing filter coefficients.

The corresponding hardware architecture of PIPEADFE2 is shown in Fig.3. , where D_m is the dummy delay in order to pipeline feed-forward part in PIPEADFE2.

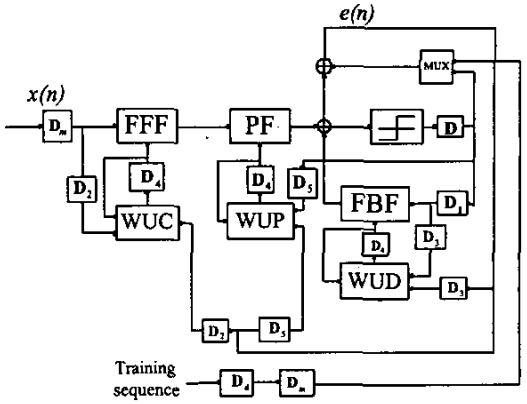

Fig.3. Architecture of PIPEADFE2.

III. Simulation Results and Comparison

In this section, we will show that the convergence rate of PIPEADFE2 is faster than PIPEADFE1 by simulation results. This is due to the extra post-processing scheme in PIPEADFE2. Moreover, the hardware complexity of PIPEADFE1 and PIPEADFE2 are almost the same if steady state prediction error of PIPEADFE1 and PIPEADFE2 are close. In our simulation, we assume channel impulse response, h=[0.2 0.6 1.0 –1.0 –0.6 –0.2] is obtained from a Lorentian pulse model[8] , and the transmitted data $a(n)$ are a **PAM5** sequence, $a(n) \in \{-1, -0.5, 0, 0.5, 1\}$.

A. Convergence Rate of PIPEADFE1 and PIPEADFE2

In the first simulation, we evaluate the convergence performance of equalizer with channel SNR=31.46dB. In serial ADFE case, N_f=6 and N_b=3, were chosen to achieve the required output SNR. The detailed parameters of the PIPEADFE1 are: $D_2 = 2D_1 = 12, D_3 = 3$, $LA=0, D_4 = 1$, step_size= 2^{-7} , N_f=12 , and N_b=3 . The parameters of PIPEADFE2 are: $D_2 = 4D = 2D_5 = 24$, , $D_3 = 3$, $LA = 0, D_4 = 1$, step_size= 2^{-7} , N_f=6 , N_b=3 , and $N_P = D_1 + 1 = 7$ .With the parameter setting, the learning curves of PIPEADFE1 and PIPEADFE2 are shown in Fig. 4. .

Based on the results shown in Fig. 4, we observe that PIPEADFE2 achieves the same steady-state SNR as PIPEADFE1 at iteration number of 4000, while PIPEADFE1 converge to the target SNR in about 8000 iterations. That is, the convergence performance is much improved by the introduced Post-processing filter.

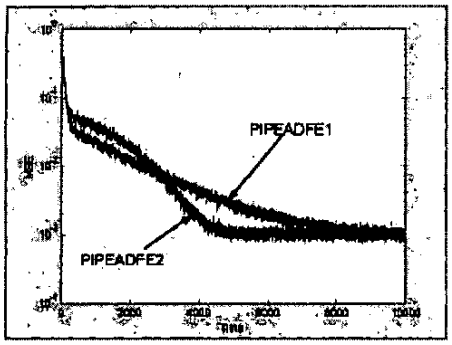

Fig. 4. Learning curves of PIPEADFE1 and PIPEADFE2

B. Output SNR versus Speedup Factor

Both PIPEADFE1 and PIPEADFE2 suffer from output SNR degradation. The purpose of this simulation is to show how the output SNR of PIPEADFE1 and PIPEADFE2 degrade as speed-up is increased. The parameters of PIPEADFE1 and PIPEADFE2 are the same as *Sec.IV.B.* except that $D_1 = speedup - 1$ and channel output SNR=28.451. The number of transmitted data samples in both architectures is 10000. The output SNR versus speedup factor is shown in Fig. 5. . We can see that both PIPEADFE1 and PIPEADFE2 have an output SNR loss of about 0.5 dB per unit increase in speedup factor.

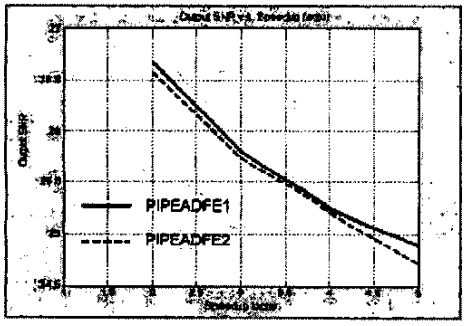

Fig. 5. Output SNR v.s. Speedup factor.

C. Comparison of Hardware Complexity

Because the output SNR depends on the number of taps in FFF and PF, we choose the number of taps in FFF and PF in order to make both architectures to achieve the same output SNR. Here, the taps of FFF in PIPEADFE1 is $N_f + D_1$, the taps of FFF in PIPEADFE2 is N_f, and the taps of PF in PIPEADFE2 is D_1. Moreover, the taps in FBF is fixed on N_b (Note: N_f and N_b is the taps of FFF and FBF in serial ADFE). Due to the multipliers in FFF and FBF have different wordlength assignment, we will have two kind of multiplier in our comparison. The speedup factor versus hardware complexity is shown in Table II. . It can be seen that the hardware complexity of PIPEADFE1 and PIPEADFE2 are the same.

	Speedup	2	N
PIPEADFE1	Mult. in FFF	$2N_f + 2$	$2(N_f + N - 1)$
	Mult. in FBF	$2N_b$	$2N_b$
	TotalAdder	$2N_f + 2N_b + 1$	$2(N_f + N_b + N) - 3$
PIPEADFE2	Mult. in FFF	$2N_f + 2$	$2(N_f + N - 1)$
	Mult. in FBF	$2N_b$	$2N_b$
	TotalAdder	$2N_f + 2N_b + 1$	$2(N_f + N_b + N) - 3$

Table II. Hardware complexity of PIPEADFE1 and PIPEADFE2.

IV. Conclusions

In this paper, a new pipelined ADFE using the post-proccessing technique is presented. Compared with the algorithm in [3] , we show the convergence rate of the proposed algorithm can be increased, while the hardware overhead is the same as the algorithm in [3] . It provides an alternative approach for the design of high-seed pipelining ADFE with large speedup factor.

Reference

[1] K. K. Parhi, VLSI Digital Signal Processing System, Wiley-Interscience, 1999.

[2] K. K. Parhi, "Pipelining in algorithm with quantizer loops," *IEEE Trans. Circ. Syst.*, vol. 38, pp.745-754, July 1991.

[3] Naresh R. Shanbhag, Keshab K. Parhi, "Pipelined adaptive DFE architectures using relaxed look-ahead," *IEEE Trans. Signal Processing*, vol. 43, No. 6, pp. 1368-1385,June 1995.

[4] M. Schobinger *et al.*, "CMOS digital adaptive decision feedback equalizter chip for multilevel QAM digital radio modems," In *Proc. IEEE Int. Symp. Cir. Syst.* ,vol. 28, pp. 330-338, Mar, 1993.

[5] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 37, pp. 1397-1405, Sept. 1989.

[6] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985

[7] S. Haykin, *Adaptive Filter Theory*, 2nd Edition, Englewood Cliffs, NJ: Prentice-Hall, 1991.

[8] I. Megory-Cohen, C. M. Melas, M. Hassner, and T. Howell, "An analytical model for magnetic readback pulses," *IBM Res. Rep. RJ* 5236, July 1986