# REDUCED-COMPLEXITY PER-SURVIVOR ITERATIVE TIMING RECOVERY FOR CODED PARTIAL RESPONSE CHANNELS

*Piya Kovintavewat, John R. Barry*

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332

*M. Fatih Erden, Erozan M. Kurtas*

Seagate Technology
1251 Waterfront Place
Pittsburgh, PA 15222

**ABSTRACT**

A (full-complexity) per-survivor iterative timing recovery scheme, which jointly performs timing recovery, equalization, and error-correction decoding, was recently proposed [1] to deal with the problem of timing recovery operating at low signal-to-noise ratio. Although it outperforms other iterative timing recovery schemes, it has very high complexity. In this paper, we propose a *reduced-complexity* per-survivor iterative timing recovery scheme to make it implementable in real-life applications. Simulation results indicate that for low to moderate complexity, the reduced-complexity scheme provides a better performance than the full-complexity scheme.

## 1. INTRODUCTION

At some point in a digital communications receiver, the received analog signal must be sampled at the instants controlled by the *timing recovery* block. Sampling at the wrong times can have a devastating impact on overall performance. Therefore, the quality of synchronization is very important.

The large coding gains of iterative error-correction codes allow reliable operation at low signal-to-noise ratio (SNR). This means that timing recovery must also function at low SNR. Thus, a conventional receiver, which performs timing recovery and error-correction decoding separately, normally fails to work properly at low SNR. To solve this problem, three iterative timing recovery schemes have been proposed in the literature [1, 2]. However, it has been shown in [1] that (full-complexity) per-survivor iterative timing recovery (will be referred to as *a full-complexity scheme*) performs better than the others, especially when timing error is large.

The full-complexity scheme [1] is realized by first developing a per-survivor Bahl, Cocke, Jelinek, and Raviv (BCJR) [3] equalizer, denoted as "PSP-BCJR," by embedding the timing recovery step inside the BCJR equalizer using per-survivor processing (PSP) [4]. Hence, PSP-BCJR iteratively exchanges soft information with a soft-in soft-output (SISO) decoder. As will be seen later, this scheme has high complexity because of PSP-BCJR, thus preventing it from being employed in real-life applications.

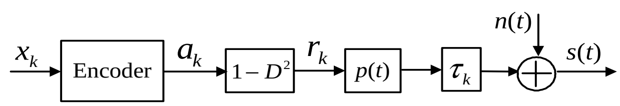


**Fig. 1**. Data encoding with a PR-IV channel model.

To reduce the complexity of PSP-BCJR, we apply the PSP concept to a soft-output Viterbi algorithm (SOVA) [5], resulting in a per-survivor SOVA equalizer, denoted as "PSP-SOVA." Then, we propose reduced-complexity per-survivor iterative timing recovery (will be referred to as *a reduce-complexity scheme*), which iteratively exchanges soft information between PSP-SOVA and the SISO decoder.

This paper is organized as follows. After explaining our channel model in Section 2, we propose and describe PSP-SOVA in Section 3. Section 4 compares the performance of different iterative timing recovery schemes. Finally, Section 5 concludes the paper.

## 2. CHANNEL DESCRIPTION

We consider the coded partial response (PR) channel model shown in Fig. 1. A message sequence $x_k \in \{0, 1\}$ is encoded by an error-correction encoder and is mapped to a binary sequence $a_k \in \{\pm 1\}$ with bit period $T$ of length $L$. The read-back signal, $s(t)$, can then be expressed as

$$s(t) = \sum_k a_k h(t - kT - \tau_k) + n(t), \qquad (1)$$

where $h(t) = p(t) - p(t - 2T)$ is a PR-IV pulse, $p(t) = \sin(\pi t/T)/(\pi t/T)$ is a 0% excess bandwidth pulse, and $n(t)$ is additive white Gaussian noise with two-sided power spectral density $N_0/2$. We model $\tau_k$ as a random walk [1] according to $\tau_{k+1} = \tau_k + \mathcal{N}(0, \sigma_w^2)$, where $\sigma_w$ determines the severity of the timing jitter. The random walk model is chosen because of its simplicity and its ability to represent

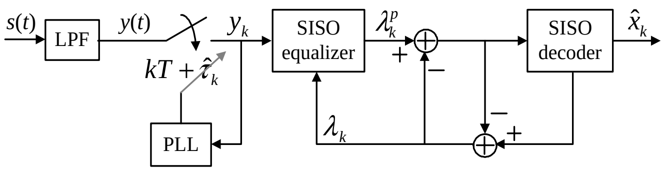


**Fig. 2**. Conventional receiver architecture.

a variety of channels by changing only one parameter. We also assume perfect acquisition by setting $\tau_0 = 0$.

At the receiver, the signal $s(t)$ is filtered by a low-pass filter (LPF), whose impulse response is $p(t)/T$, to eliminate out-of-band noise, and is sampled at time $kT + \hat{\tau}_k$, creating

$$y_k = y(kT+\hat{\tau}_k) = \sum_i a_i h(kT+\hat{\tau}_k - iT - \tau_i) + n_k, \quad (2)$$

where $\hat{\tau}_k$ is the receiver's estimate of $\tau_k$, and $n_k$ is *i.i.d.* zero-mean Gaussian random variable with variance $\sigma_n^2 = N_0/(2T)$, i.e., $n_k \sim \mathcal{N}(0, \sigma_n^2)$.

Conventional timing recovery is based on a phase-locked-loop (PLL) [6]. Because perfect acquisition is assumed and our model has no frequency offset component, the sampling phase offset can be updated by a first-order PLL [6], i.e.,

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \mu\{y_k\tilde{r}_{k-1} - y_{k-1}\tilde{r}_k\}, \quad (3)$$

where $\mu$ is a PLL gain parameter, and $\tilde{r}_k$ is the $k$-th soft estimate of the *channel output* $r_k \in \{0, \pm 2\}$ given by [2]

$$\tilde{r}_k = E[r_k|y_k] = \frac{2\sinh(2y_k/\sigma_n^2)}{\cosh(2y_k/\sigma_n^2) + \exp(2/\sigma_n^2)}. \quad (4)$$

The soft estimate provides a better performance than the hard estimate [2], which is obtained by a memoryless three-level quantization of $y_k$.

In a conventional setting, conventional timing recovery is followed by a turbo equalizer [7] (see Fig. 2), which iteratively exchanges soft information between the SISO equalizer for the PR-IV channel and the SISO decoder.

## 3. PSP-SOVA

Because SOVA performs on the same trellis as the Viterbi algorithm [8] does, we then apply the PSP concept to develop PSP-SOVA by embedding the timing recovery step inside the SOVA equalizer so as to perform timing recovery and equalization jointly. Fig. 3 shows the PSP-SOVA algorithm, where the lines starting with * are the additional steps beyond the conventional SOVA. It should be noted that PSP-SOVA works in a *same* manner as *PSP-based timing recovery*[1] [9] does (i.e, from (A-1) to (A-10)), except that

[1] It is implemented based on the Viterbi algorithm.

(A-1) Initialize $\Phi_0(p) = 0$ for $\forall p$
*(A-2) Initialize $\hat{\tau}_0(p) = 0$ for $\forall p$
(A-3) For $k = 0, 1, \ldots, L+\nu-1+\delta$
(A-4) For $q = 0, 1, \ldots, Q-1$
*(A-5) $y_k(p) = y(kT + \hat{\tau}_k(p))$ for $\forall p$
(A-6) $\rho_k(p,q) = |y_k(p) - \hat{r}(p,q)|^2$ for $\forall p$
(A-7) $\pi_{k+1}(q) = \arg\min_p\{\Phi_k(p) + \rho_k(p,q)\}$
(A-8) $\Phi_{k+1}(q) = \Phi_k(\pi_{k+1}(q)) + \rho_k(\pi_{k+1}(q), q)$
(A-9) $\mathbf{S}_{k+1}(q) = [\,\mathbf{S}_k(\pi_{k+1}(q)) \mid \pi_{k+1}(q)\,]$
*(A-10) $\hat{\tau}_{k+1}(q) = \hat{\tau}_k(\pi_{k+1}(q)) + \mu\{y_k(\pi_{k+1}(q))\hat{r}(\pi_k(\pi_{k+1}(q)), \pi_{k+1}(q))$
$- y_{k-1}(\pi_k(\pi_{k+1}(q)))\hat{r}(\pi_{k+1}(q), q)\}$
(A-11) $\Delta_{k+1}(q) = \max_p\{\Phi_k(p) + \rho_k(p,q)\} - \Phi_{k+1}(q)$
(A-12) Initialize $\hat{L}_k(q) = +\infty$ [**Soft decision update** [5]]
For $j = k-\nu, \ldots, k-\delta$
Compare the two paths merging in state $q$ (i.e., $\Psi_{k+1} = q$)
If $\hat{a}_j^{(1)}(\Psi_{j+1}) \neq \hat{a}_j^{(2)}(\Psi_{j+1})$, update
$\hat{L}_j(\Psi_{j+1}) = \min(\hat{L}_j(\Psi_{j+1}),\, \Delta_{k+1}(q))$
End
(A-13) End
(A-14) If $k \geq \delta$
Output the soft decision according to $\lambda_{k-\delta}^p = \hat{a}_{k-\delta}\hat{L}_{k-\delta}$, which can be extracted from the survivor path that minimizes $\Phi_{k+1}$
(A-15) End
(A-16) End

**Fig. 3**. PSP-SOVA algorithm, where the lines starting with * are the additional steps beyond the conventional SOVA.

PSP-SOVA has *extra* steps for approximating the *a posteriori* log-likelihood ratio (LLR) of each data bit (or a soft decision), which can be briefly explained as follows.

Following the notations in [9], at each $k$-th stage, the metric difference for state $q$ at time $k+1$, $\Delta_{k+1}(q)$, is computed by (A-11). Then, the *tentative* LLR is updated based on (A-12), where $\hat{L}_k(q)$ is the $k$-th LLR associated with state $q$, $\hat{a}_j^{(m)}(\Psi_{j+1})$ is the $j$-th estimated data bit associated with the $m$-th path that passes $\Psi_{j+1}$, and $m \in \{1, 2\}$ is used to indicate the two paths that merge in state $q$ (where $m = 1$ represents a correct path and $m = 2$ represents a wrong path). After a *decoding depth*, $\delta$, the *a posteriori* LLR of $a_k$, $\lambda_k^p$, is produced by (A-14). Note that $\delta = 5(\nu + 1)$, where $\nu$ is channel memory, is employed in this paper.

Beyond the conventional SOVA, PSP-SOVA needs new storage requirements for (i) the sampling phase offsets and (ii) the sampler outputs. However, only sampling phase offsets and sampler outputs of the *current* and *previous* stages need to be stored, thus minimizing extra memory. Like PSP-based timing recovery, PSP-SOVA requires one PLL for each survivor path. Thus, for a PR-IV channel, the complexity of timing recovery is four times the complexity of conventional timing recovery. Additionally, instead of storing $y(t)$, we could uniformly sample $y(t)$ at symbol rate to obtain a set of samples $\{y_k\}$. Then, we can only store this set of samples because the bandlimited nature of $y(t)$ makes it sufficient statistics. Therefore, PSP-SOVA can perform the timing update operation using $\{y_k\}$ and a digital interpolation filter, thus decreasing its complexity. In this paper, a 21-tap sinc interpolation filter (i.e., $N_s = 21$) is used.

To help quantify how much computational complexity PSP-SOVA contains if compared to PSP-BCJR, we measure complexity by counting the total number of additions and multiplications (*per bit*). For other mathematical functions, e.g., $\log(x)$, $\exp(x)$, etc., we assume they are implemented as lookup tables, and that we ignore their complexity. It can be shown that PSP-SOVA has $(8 + 4N_s)Q + \frac{\delta^2+9\delta+9}{2} + 1$ additions and $(9 + N_s)Q + 1$ multiplications, whereas PSP-BCJR has $(14 + 8N_s)Q - 2$ additions and $(26 + 2N_s)Q + 1$ multiplications, where $Q = 2^\nu$ is the number of trellis states. Clearly, PSP-SOVA has lower complexity than PSP-BCJR. Furthermore, we also found that PSP-SOVA needs less memory than PSP-BCJR. Specifically, based on our channel model, PSP-BCJR requires $2(L+\nu+8)Q+4$ memory units, whereas PSP-SOVA needs only $(16+2\delta)Q+8+\delta$ memory units. This suggests that PSP-SOVA is preferable to PSP-BCJR in terms of cost implementation.

## 4. NUMERICAL RESULTS

Reduced-complexity per-survivor iterative timing recovery is obtained by discarding the front-end PLL in Fig. 2 and replacing the SISO equalizer with PSP-SOVA.

Consider a rate-8/9 system in which a block of 3640 message bits is encoded by a regular $(3, 27)$ low-density parity-check (LDPC) code [10], resulting in a coded block length of 4095 bits. The parity-check matrix has 3 ones in each column and 27 ones in each row. The SISO equalizer is implemented based on SOVA, whereas the SISO decoder is implemented based on the message passing algorithm [10] with $N_i = 5$ internal iterations. The PLL gain parameters for different iterative timing recovery schemes were optimized based on minimizing the RMS timing error, $\sigma_\epsilon = \sqrt{E[(\tau_k - \hat{\tau}_k)^2]}$, at a per-bit SNR, $E_b/N_0$, of 5 dB.

Fig. 4 compares the BER performance of different iterative timing recovery schemes at the 10-th iteration for the systems with $\sigma_w/T = 0.5\%$ (imply a *low* probability of occurrence of a cycle slip) and $\sigma_w/T = 1\%$ (imply a *high* probability of occurrence of a cycle slip). Apparently, the reduced-complexity scheme outperforms the conventional receiver, especially when $\sigma_w/T$ is large. This is because the reduced-complexity scheme can correct a cycle slip (same as the full-complexity scheme) as opposed to the conventional receiver. It is evident that for a given number of iterations, the full-complexity scheme yields a better perfor-

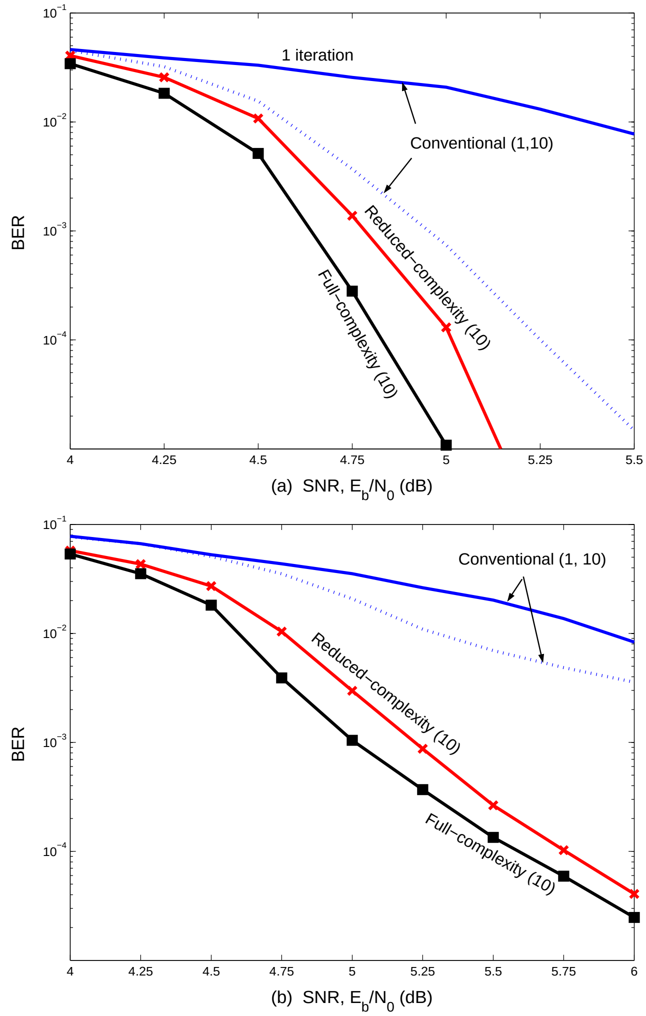


**Fig. 4**. Performance comparison at the 10-th iteration when (a) $\sigma_w/T = 0.5\%$ and (b) $\sigma_w/T = 1\%$.

mance than the reduced-complexity scheme. Nonetheless, we will show that the reduced-complexity scheme will perform better than the full-complexity scheme if we compare their performances when they have same complexity.

To do so, we count the number of operations (*per bit*) of different schemes, including an LDPC decoder. Note that it can be shown the LDPC decoder requires $(j + (k-1)(1 - R))N_i + 1$ additions and $(1 - R)N_i$ multiplications, where $(j, k) = (3, 27)$ is an LPDC parameter, and $R = 1 - j/k$ is a code rate. Let $N$ be the number of iterations. Then, by using $N_s = 21$, $Q = 4$, $\delta = 15$, and $N_i = 5$, we can show that the conventional receiver has $86 + 245.94N$ additions and $27 + 25.56N$ multiplications; the full-complexity scheme has $758.44N$ additions and $273.56N$ multiplications; and the reduced-complexity scheme has $585.94N$ additions and $121.56N$ multiplications. However, it should be pointed out that multiplication has much more complexity than addi-

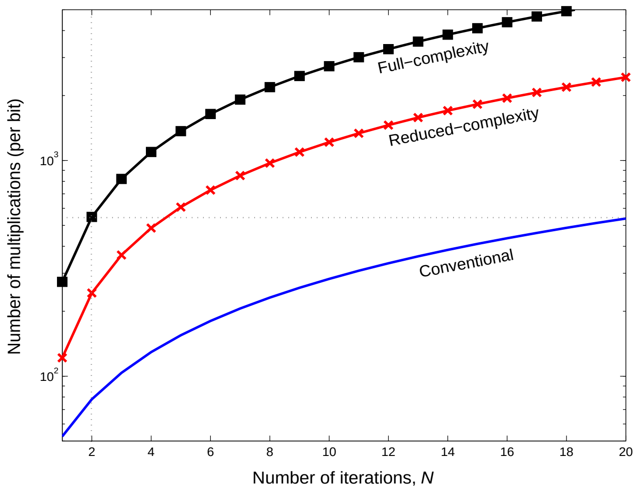


**Fig. 5**. Complexity comparison (based on a PR-IV channel).

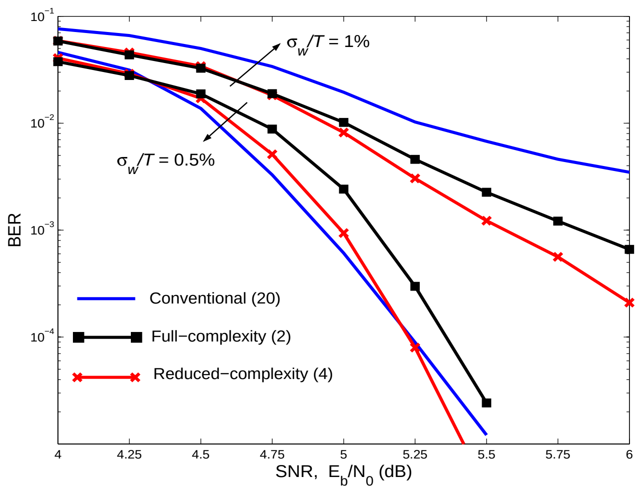


**Fig. 6**. Performance comparison with same complexity.

tion in terms of circuit implementation. Consequently, we consider only the number of multiplications when comparing the performance of different iterative timing recovery schemes. Fig. 5 compares the number of multiplications of each scheme. Clearly, the full-complexity scheme has very high complexity if compared to the others.

In addition, we also assume that current technology can support the total number of multiplications equal to 2 iterations of the full-complexity scheme, which is approximately equal to 4 iterations of the reduced-complexity scheme and 20 iterations of the conventional receiver (see Fig. 5). Fig. 6 compares the performance of different iterative timing recovery schemes when they have same complexity. It is evident that the reduced-complexity scheme performs better than both per-survivor iterative timing recovery and the conventional receiver, especially at high SNR.

## 5. CONCLUSION

We proposed reduced-complexity per-survivor iterative timing recovery, which jointly performs timing recovery, equalization, and error-correction decoding, to make it implementable in real-life applications. Simulation results have shown that for low to moderate complexity, the reduced-complexity scheme performs better than the full-complexity scheme and the conventional receiver.

## 6. REFERENCES


[1] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. M. Kurtas, "Per-survivor iterative timing recovery for coded partial response channels," to appear in *Proc. of Globecom'04*, Dallas, Texas, Nov 29 – Dec 3, 2004.

[2] J. R. Barry, A. Kavcic, S. W. McLaughlin, A. R. Nayak, and W. Zeng, "Iterative timing recovery," *IEEE Signal Processing Magazine*, vol. 21, pp. 89–102, Jan 2004.

[3] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, no. 2, pp. 248–287, Mar 1974.

[4] R. Raheli, A .Polydoros, and C. K. Tzou, "The principle of per-survivor processing: a general approach to approximate and adaptive MLSE," in *Proc. of Globecom'91*, vol. 2, pp. 1170–1175, Dec 1991.

[5] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," in *Proc. of Globecom'89*, pp. 1680–1686, Nov 1989.

[6] J. W. M. Bergmans, *Digital baseband transmission and recording*, Kluwer Academic Publishers, Boston, Massachusetts, 1996.

[7] T. Souvignier, A. Friedmann, M. Öberg, P. H. Siegel, R. E. Swanson, and J. K. Wolf, "Turbo decoding for PR4: parallel vs. serial concatenation," in *Proc. of ICC'99*, vol. 3, pp. 1638–1642, Jun 1999.

[8] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 3, pp. 363–378, May 1972.

[9] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. M. Kurtas, "Per-survivor processing (PSP) -based timing recovery for uncoded partial response channels," in *Proc. of ICC*, vol. 5, pp. 2715–2719, Jun 2004.

[10] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan 1962.