# Multirate Digital Filters for Symbol Timing Synchronization in Software Defined Radios

Fredric J. Harris, *Senior Member, IEEE,* and Michael Rice, *Senior Member, IEEE*

***Abstract*—This paper describes the use of a polyphase filterbank to perform the interpolations required for symbol timing synchronization in a sampled-data receiver. The polyphase filterbank possesses advantages over architectures based on separate matched and interpolation filters. Interpolations are realized by filterbank index selection and a separate interpolating filter following the matched filter is not required. Maximum likelihood timing synchronization techniques can be easily incorporated into the polyphase filter bank in a natural way. An $M$-stage polyphase filterbank with input data sampled at approximately $N$ samples/symbol can be used in a loop that operates at $MN$ samples/symbol, $N$ samples/symbol, or 1 sample/symbol. When operating at 1 sample/symbol, auxiliary control must also be included to adjust the clocking of data into the filter bank to account for small differences in the sample clock and $N$ times the data clock. Examples are presented to illustrate loop performance and control.**

***Index Terms*—Digital communication, digital filters, synchronization.**

## I. Introduction

DIGITAL signal processing (DSP) has become the standard method of signal conditioning and signal processing in receivers of many synchronous communication systems. These systems require acquisition and tracking of a carrier and a timing clock from the received signal when neither carrier nor clock is present. Synchronization techniques abound with clever *ad hoc* methods developed by clever designers. These suboptimal synchronization techniques emerged concurrently with the development of the theoretical basis of synchronization processes with many techniques developed prior to the heavy reliance upon DSP. As the transition to DSP occurred, analog-based synchronization schemes often survived the change and were implemented as digitized versions of the analog prototype technique. Digital systems designed this way often contain legacy implementation compromises that can be avoided in DSP-based solutions. In addition, DSP offers a number of options and solutions not previously available to the analog system designer.

Digital signal processing can be applied to symbol timing synchronization for MQASK systems where the samples of the received signal are not aligned with the data clock used to generate the analog waveform. Gardner formulated the problem in [1] where the use of an interpolation filter either preceding or following a sampled version of the matched filter was examined. This approach was explored fully in [2], [3, Chap. 7], and [4, Chaps. 4, 9]. In this paper, we consider a different approach for performing the interpolations required for symbol timing synchronization. We use an upsampled version of the matched filter as the interpolation filter and use a polyphase decomposition of the upsampled matched filter to gain implementation advantages. This approach eliminates the need for a separate interpolation filter and provides a natural way to incorporate maximum likelihood (ML) timing estimation into the loop.

The use of a polyphase filterbank as digital phaseshifter has been known for some time. To our knowledge, this was first observed by Crochiere *et al.* [5] who suggested applications to echo-canceller simulations, multiple signal processing in a phased array antenna system, or for synchronous synthesis of speech. Selectable fractional sample delays are now a common application of polyphase filterbanks [6]–[8]. Special cases of the use of polyphase filterbanks for symbol timing synchronization have been described in [9]–[12]. In this paper, we present generalizations to these ideas and describe three loop architectures based on the polyphase filter. In Section II, the basic problem is defined to establish notation. Continuous-time maximum likelihood and early–late gate synchronization are briefly reviewed for use later in the polyphase filter development. Discrete-time approaches are also reviewed to provide a basis for comparison with the polyphase filterbank implementation. In Section III, the polyphase filterbank version of the interpolator is developed in the context of symbol timing synchronization. Some implementation details, including different architecture options and ML timing phase detectors, are discussed in Section IV. Examples are presented in Section V together with simulation results. The paper ends with some concluding remarks.

## II. Background

Let the transmitted signal be

$$s(t) = \sum_k a_k p(t - kT) \tag{1}$$

where $a_k$ is the $k$th complex valued $M$-ary symbol drawn from a constellation with average symbol energy $E$ and $p(t)$ is the unit-energy pulse shape that spans $2L$ symbols. The received signal is

$$r(t) = s(t - \tau) + w(t) \tag{2}$$

where $w(t)$ is the additive noise term which is modeled as a zero-mean white Gaussian random process with power spectral

Manuscript received January 30, 2001; revised June 29, 2001.

F. J. Harris is with the Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92182 USA (e-mail: fred.harris@sdsu.edu).

M. Rice is with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT 84602 USA (e-mail: mdr@ee.byu.edu).

Publisher Item Identifier S 0733-8716(01)10296-9.

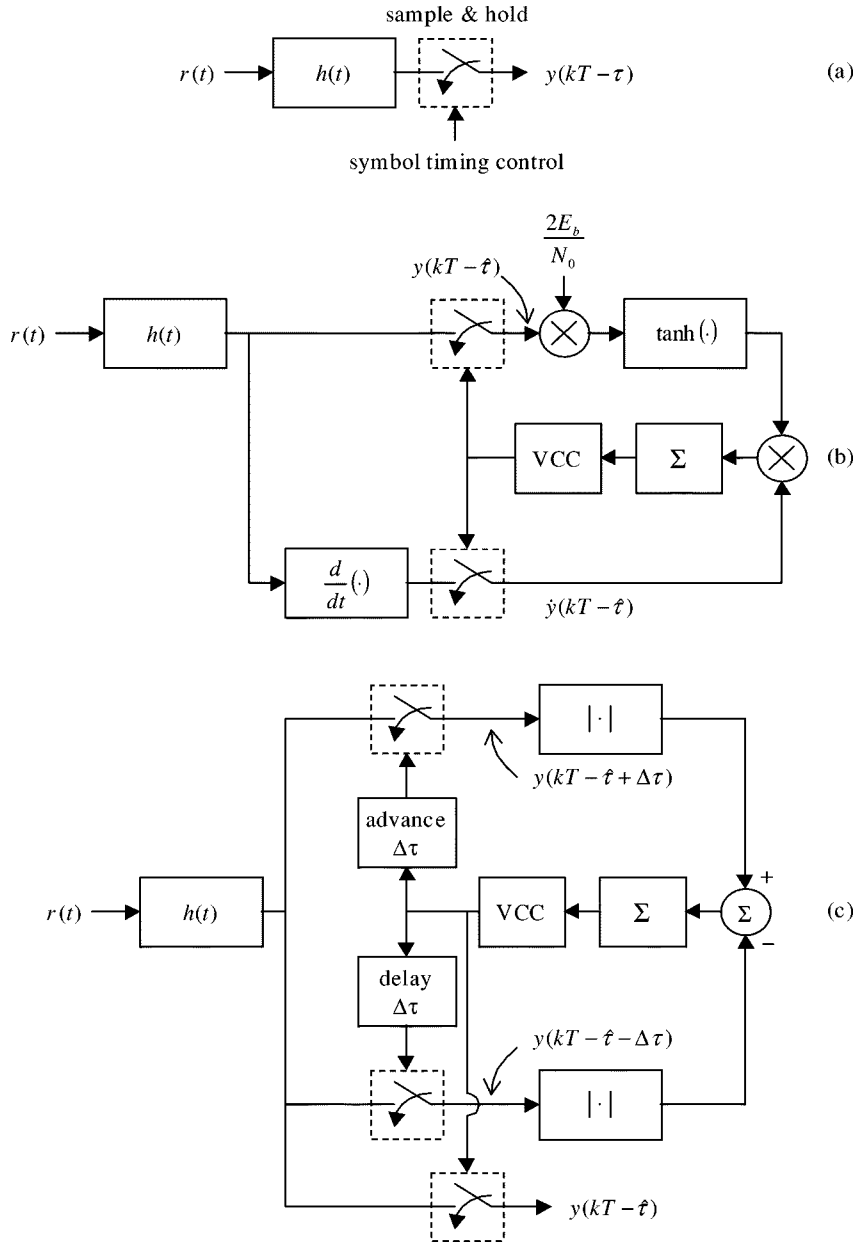


Fig. 1. Continuous-time symbol timing synchronization. (a) Sampling control of continuous-time matched filter output. (b) Maximum-likelihood timing synchronization phase locked loop. (c) Early–late gate approximation to ML timing synchronization phase locked loop.

density $N_0$ W/Hz and $\tau$ is a delay relative to the detector's time axis. The optimum detector uses the output of a matched filter with impulse response $h(t) = p(-t)$ as the basis for decisions. The matched filter output $y(t) = r(t) * h(t)$ is sampled at the end of the symbol interval to produce the required decision variable. This requires knowledge not only of the symbol rate $1/T$ but also the exact sampling instant within the symbol interval (i.e., the *timing phase*). Extracting this knowledge may be done either with continuous-time processing, discrete-time processing, or both.

### A. Continuous-Time Techniques

The continuous-time synchronizer controls a sample-and-hold circuit located at the output of the analog matched filter as illustrated in Fig. 1(a). The goal of the synchronizer is to sample the matched filter output $y(t)$ at the optimum instant for the $k$th symbol which is $kT+\tau$. It is well known that the log-likelihood function for the unknown timing phase $\tau$ for equally likely $a_k \in \{-1, +1\}$ is [13] or [4, Chap. 6]

$$\Lambda(\tau) = \sum_k \ln\left(\cosh\left(\frac{2E}{N_0}\, y(kT+\tau)\right)\right) \tag{3}$$

where

$$y(kT+\tau) = \frac{1}{\sqrt{E}} \int_{(k-L)T}^{(k+L)T} r(t)p(t - kT - \tau)\, dt \tag{4}$$

is the normalized matched filter output corresponding to the $k$th symbol. The estimate $\hat{\tau}$ that maximizes $\Lambda(\tau)$ is the timing phase that forces the derivative $\Lambda(\tau)$ to be zero

$$\frac{d}{d\tau}\Lambda(\tau) = \sum_k \tanh\left(\frac{2E}{N_0}y(kT+\tau)\right)\frac{d}{d\tau}y(kT+\tau) = 0. \tag{5}$$

Equation (5) suggests the timing recovery loop illustrated in Fig. 1(b) [13], [14]. The output of the matched filter and its derivative are combined and summed over successive symbol periods to form an error signal which is the middle term in (5). The phase of the voltage controlled clock (VCC) adjusted to force the error term to zero. This algorithm is often interpreted as a process that seeks the location of the maximum eye opening in the eye diagram. The loop controls the phase of the sampling clock at the output of the analog matched filter using the voltage controlled clock.

Often, the $\tanh(\cdot)$ function is replaced by its small signal approximation $\tanh(x) \approx x$ for low SNRs and by the large signal approximation $\tanh(x) \approx \mathrm{sgn}(x)$ for high SNRs. Another popular approximation to the ML phase detector is the early–late gate synchronizer shown in Fig. 1(c). The derivative is approximated by

$$\frac{d}{d\tau}\Lambda(\tau) \approx \frac{\Lambda(\tau+\Delta\tau)-\Lambda(\tau-\Delta\tau)}{2\Delta\tau} \tag{6}$$

which, for large $E/N_0$, is well approximated by[1]

$$\frac{d}{d\tau}\Lambda(\tau) \approx \sum_k \left|\frac{2E}{N_0}y(kT+\tau+\Delta\tau)\right| - \left|\frac{2E}{N_0}y(kT+\tau-\Delta\tau)\right| \tag{7}$$

where $\Delta\tau$ is an adjustable advance/delay parameter that satisfies $0 < \Delta\tau < T/2$.

### B. Discrete-Time Techniques

When the matched filter is implemented as a discrete-time filter, an analog-to-digital converter (ADC) preceding the filter is required. The ADC samples the bandlimited signal $r(t)$ every $T_s$ seconds where the sampling rate satisfies the Nyquist rate condition (e.g., $T_s \leq T/2$ for the square-root raised cosine pulse shape). These samples $r(nT_s)$ are then filtered by the discrete-time matched filter with impulse response $h(nT_s) = p(-nT_s)$. In this case, the desire is to produce $N$ samples of the matched filter output during each symbol interval such that one of the samples is as close to $y(kT+\tau)$ as possible. This sample is held for detection while the other $N-1$ are ignored.[2] The parameter $N$ is the number of samples/symbol and is usually a small integer (1 or 2).

There are two basic approaches to the problem. The first approach is illustrated in Fig. 2(a). This approach computes a timing error value which is used to adjust the phase of the voltage

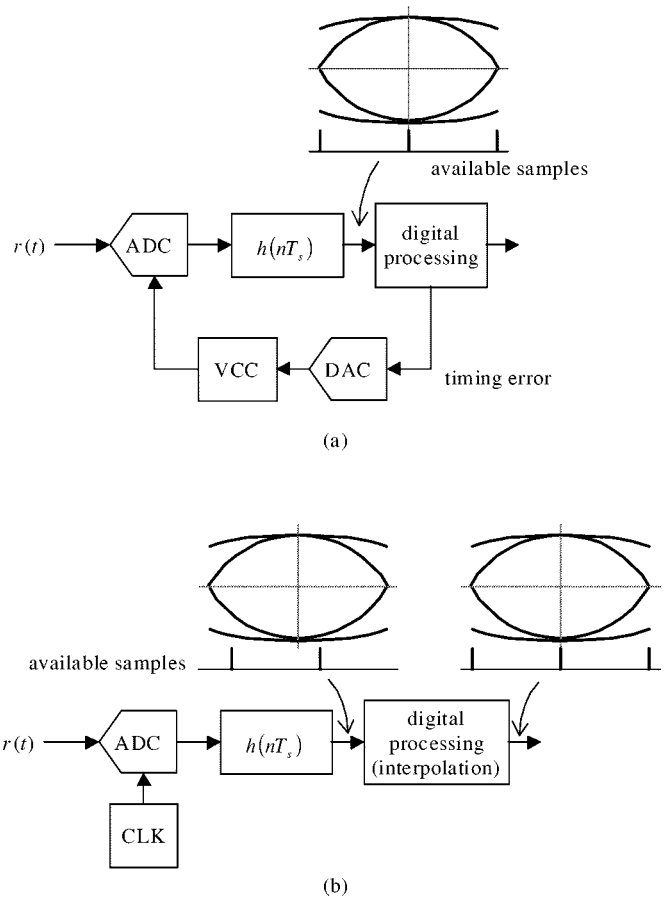


Fig. 2. The two basic discrete-time timing synchronization approaches.

controlled clock (VCC) that triggers the ADC. As a result, the samples of $r(t)$ are aligned with the symbol boundaries. The systems described in [15]–[19] are of this type. This approach has the advantage that it produces samples that are aligned in both phase and frequency with the data clock (i.e., $T$ and $T_s$ are commensurate). There are four disadvantages to this approach.

1) First, a feedback path to the continuous-time part of the system is required. The hardware overhead of transferring from the digital to analog domains via a multibit output bus, a data control line, a multibit DAC, and an analog filter to supply the control voltage to the VCC has the potential to complicate the analog front-end design [20].
2) Second, the transport delay of the matched filter now resides in the feedback path of the timing control loop. This significantly reduces the response time of the timing recovery loop.
3) Third, higher levels of phase noise (and hence, timing jitter) are contributed by the VCC relative to the phase noise contributed by fixed-frequency sampling clocks.
4) Fourth, this technique does not allow the ADC to be placed at the IF if the IF signal contains multiplexed signals whose symbol clocks are derived from independent sources. In software defined radios, the goal is to "push the ADC to the antenna." To meet this goal, demultiplexing and channel selection must be performed using digital signal processing on asynchronous samples of $r(t)$.

[1]To see this, use the approximation $\ln(\cosh(x)) \approx |x|$ for large $x$.

[2]This statement is not true for systems using a fractionally spaced equalizer. In this case, the equalizer following the matched filter operates on multiple (usually $N = 2$) matched filter outputs per symbol.

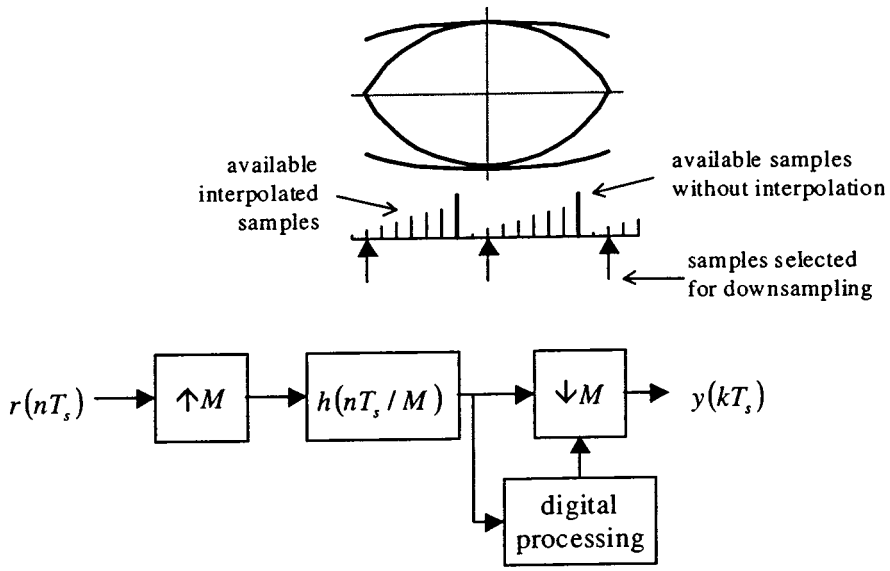


Fig. 3. Interpolation using upsampling, filtering at the high rate, and downsampling.

The second approach addresses these issues by sampling the received signal $r(t)$ at a fixed rate $1/T_s$, that is asynchronous with the data clock frequency $1/T$ (i.e., $T/T_s$ is irrational). The time delay $\tau$ is estimated solely from $y(nT_s)$, the asynchronous samples at the output of the matched filter. The problem is best understood when cast as an interpolation problem where the sample rate at the output of the interpolator is slightly different from the sample rate at the input of the interpolator [1]. This approach is illustrated by the block diagram in Fig. 2(b). Usually, interpolation is accomplished using an interpolation filter that either precedes or follows the discrete-time matched filter. Typically, the interpolation filters are polynomial filters (e.g., linear, piece-wise parabolic, or cubic) [2], [3, Chap. 7], [4, Chap. 4, 9]. While this approach avoids the need to close the loop in the continuous-time domain, it presents a new problem: Due to the asynchronous nature of the samples, $T_s$ and $T$ will rarely be commensurate thus complicating the rate change from $1/T_s$ samples/s to $N/T$ samples/s.

The equations that describe this process were derived by Gardner [1]. Matched filter outputs $y(mT_s)$ at the asynchronous rate $1/T_s$ are presented to an interpolation filter with impulse response $h_I(\cdot)$. This filter is used to produce interpolated values of $y(mT_s)$ every $T_i = T/N$ seconds. Let $\hat{y}(kT_i)$ be the optimum matched filter output for the $k$th symbol. Then, following Gardner [1], the relationship between $\hat{y}(kT_i)$ and $y(mT_s)$ is

$$\hat{y}(kT_i) = \sum_m y(mT_s)h_I(kT_i - mT_s). \tag{8}$$

Now using $n_k = \lfloor kT_i/T_s \rfloor$ (the basepoint index), $\mu_k = kT_i/T_s - n_k$ (the fractional interval), and $i = n_k - n$, (8) may be reexpressed as

$$\hat{y}(kT_i) = \sum_{i=I_1}^{I_2} y((n_k - i)T_s)h_I((i + \mu_k)T_s) \tag{9}$$

where the time span of the interpolation filter is the interval $I_1 \le i \le I_2$. The effect of the asynchronous sample clock is observed in the behavior of the fractional interval $\mu_k$ [1].

- $T_i$ is incommensurate with $T_s$: In this case, $\mu_k$ is irrational and changes for each $k$ for infinite precision or progresses through an infinite set of values, never repeating exactly for finite precision.
- $T_i \approx T_s$: In this case $\mu_k$ changes very slowly for infinite precision or remains constant for many $k$, for finite precision.
- $T_s$ is commensurate with $T_i$, but not equal: In this case, $\mu_k$ cyclically progresses through a finite set of values.

## III. Polyphase Filterbanks for Timing Synchronization

The polyphase filterbank is applied to symbol timing recovery by using a polyphase decomposition of the matched filter to realize interpolation instead of a separate polynomial interpolation filter. In this way, interpolation and matched filtering are rolled into a single operation. The polyphase filter bank also allows easy phase control using ML techniques and accommodates awkward rate change (from $1/T$ to $1/T_s$) in a natural way.

The polyphase filter structure is illustrated by the following simple example illustrated in Fig. 3. Suppose that the required timing resolution is $MN$ parts per symbol and that samples of the phase-corrected complex baseband signal $r(nT_s)$ provide approximately $N$ samples/symbol. The sequence $r(nT_s)$ is upsampled by a factor $M$ by inserting $M-1$ zeros between each sample of $r(nT_s)$ to produce a new sequence $r(nT_s/M)$ that provides $MN$ samples/symbol. The sequence forms the input to a matched filter whose impulse response is $h(nT_s/M)$ and is also sampled at $MN$ samples/symbol and spans $2L$ symbols. The timing control selects approximately $N$ samples during each symbol period such that one of them is as close to the op-

timum sampling instant as possible. The output $y(nT_s/M)$ is given by

$$y\left(n\frac{T_s}{M}\right) = \sum_{l=-MNL}^{MNL} r\left((n-l)\frac{T_s}{M}\right) h\left(l\frac{T_s}{M}\right). \quad (10)$$

The output is downsampled to produce $N$ samples per symbol where one of the samples is as close to $y(nT_s+\tau)$ as the resolution allows. The polyphase decomposition is due to the fact that not all of the multiplies defined by (10) are required. Since

$$r\left(n\frac{T_s}{M}\right) = \begin{cases} r(nT_s), & n = 0, \pm M, \pm 2M, \ldots \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

only every $M$th value of $r(nT_s/M)$ in the FIR filter is nonzero. At a time instant at the high sample rate, these nonzero values coincide with the filter coefficients

$$\ldots, h(-2MT_s), h(-MT_s), h(0), h(MT_s), h(2MT_s), \ldots$$

and the filter output may be expressed as

$$\sum_{i=-NL}^{NL} r((n-i)T_s)h(iT_s) = y(nT_s). \quad (12)$$

At the next time instant the nonzero values of $r(nT_s/M)$ coincide with the filter coefficients

$$\ldots, h(-2MT_s+1), h(-MT_s+1), h(1), h(MT_s+1), h(2MT_s+1), \ldots$$

so that the filter output may be expressed as

$$\sum_{i=-NL}^{NL} r((n-i)T_s)h\left(\left(i+\frac{1}{M}\right)T_s\right) = y\left(\left(n-\frac{1}{M}\right)T_s\right). \quad (13)$$

At the $m$th time instant, the nonzero values of $r(nT_s/M)$ coincide with the filter coefficients

$$\ldots, h(-2MT_s+m), h(-MT_s+m), h(m), h(MT_s+m), h(2MT_s+m), \ldots$$

so that the filter output may be expressed as

$$\sum_{i=-NL}^{NL} r((n-i)T_s)h\left(\left(i+\frac{m}{M}\right)T_s\right) = y\left(\left(n-\frac{m}{M}\right)T_s\right). \quad (14)$$

This characteristic is illustrated in Fig. 4 where a parallel bank of $M$ filters, operating at the low sample rate $1/T_s$, is shown. Each filter in the filterbank is a downsampled version of the matched filter, except with a different index offset. The impulse response for $h_m(nT_s)$ is

$$h_m(nT_s) = h\left(nT_s + \frac{m}{M}T_s\right). \quad (15)$$

The data samples $r(nT_s)$ form the input to all the filters in the filterbank simultaneously. The desired phase shift of the output is selected by connecting the output to the appropriate filter in the filterbank.

To see that the output of the $m$th filter in the polyphase filter bank given by (14) does indeed produce the desired result given by (9), assume for the moment that $T_i/T_s$ in (9) is sufficiently close to one so that $n_k = k$. Then (9) becomes

$$\hat{y}(kT_i) = \sum_{i=I_1}^{I_2} y((k-i)T_s)h_I((i+\mu_k)T_s). \quad (16)$$

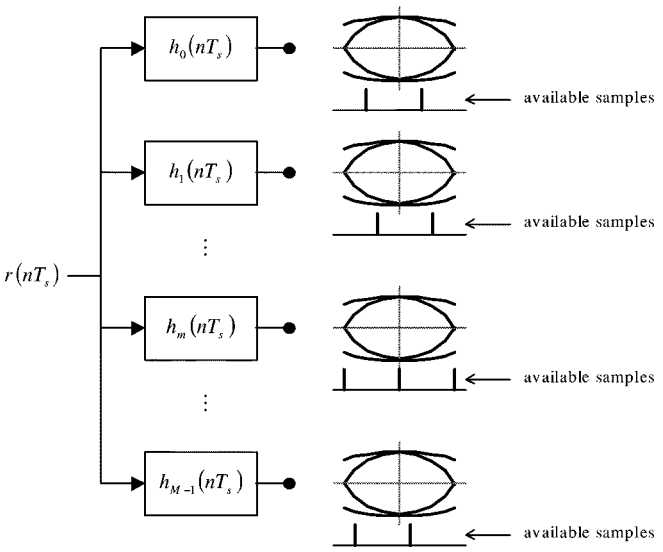


Fig. 4. Polyphase filterbank interpolator equivalent to the interpolator shown in Fig. 3.

Since the polyphase filterbank implementation uses the matched filter as the interpolation filter, the input data sequence $r(nT_s)$ in (14) plays the role of the matched filter output $y(nT_s)$ in (16) and the matched filter $h(nT_s)$ in (14) plays the role of the interpolation filter in (16). The comparison shows that the ratio of the polyphase filter stage index $m$ to the number of filterbank stages $M$ plays the same role as the fractional interval $\mu_k$ in the interpolation filter. In this way, the polyphase filterbank implements the interpolation defined by (9) with a quantized fractional interval. The degree of quantization is controlled by the number of polyphase filter stages in the filterbank. The observations regarding the behavior of $\mu_k$ at the end of Section II apply to the filter stage index $m$ for the cases where $T$ and $T_s$ are not commensurate.

The desired output is selected by a discrete-time phase locked loop (DPLL). When used in conjunction with a timing phase error detector, the DPLL outputs the index associated with the proper phase of the matched filter output corresponding to the maximum eye opening.

## IV. Implementation Issues

### A. Polyphase Filterbank Implementation

The direct brute-force implementation of the polyphase filterbank requires the operation of $M$ polyphase filters that operate in parallel. In reality, $M$ filters are not constructed, but rather a single stage filter with $M$ set of weights that are selected from memory by a pointer under control of the phase locked loop. This makes the polyphase filterbank an especially attractive alternative for implementing software defined radios on FPGA platforms [11], [21]–[24].

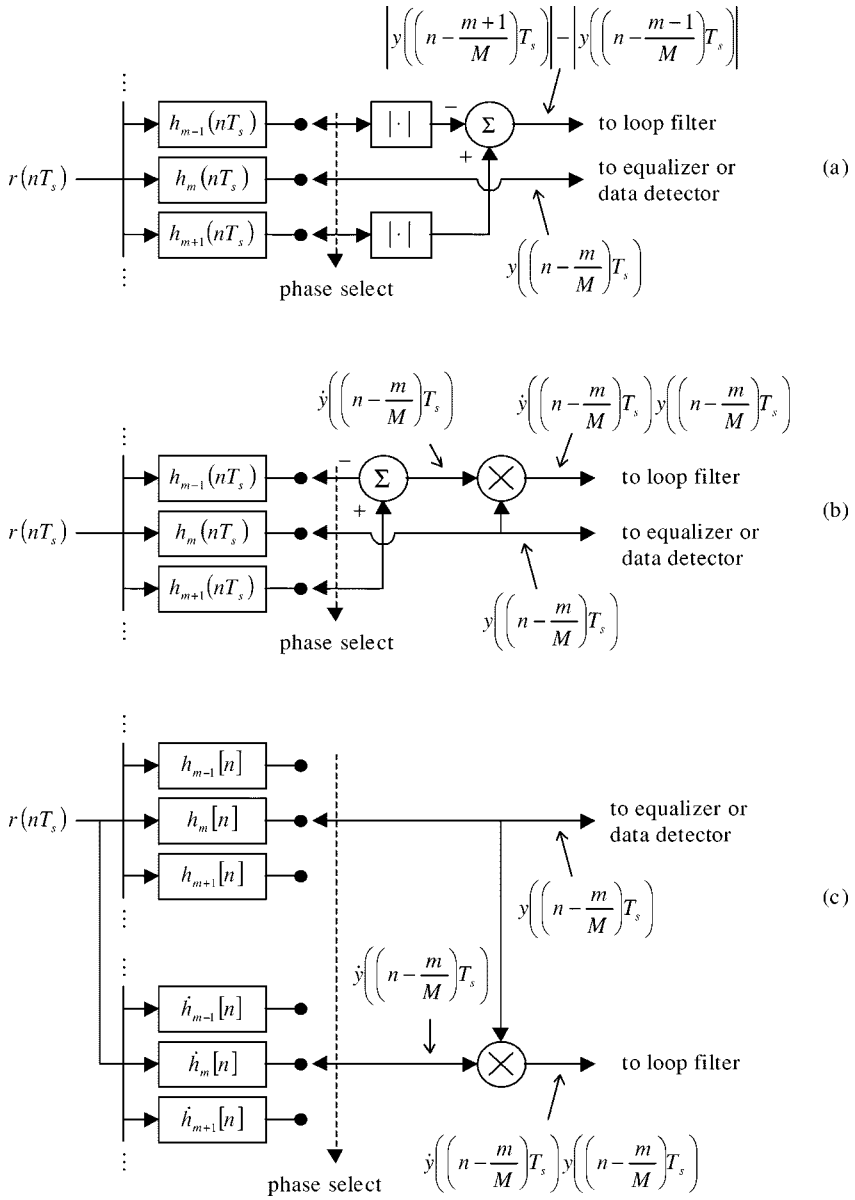


Fig. 5. Timing phase error detectors using a polyphase filter bank. (a) The early–late gate approximation. (b) ML phase error using two filters to compute the derivative (first central difference). (c) ML phase error using derivative matched filter.

### *B. Timing Error Phase Detector*

The polyphase filter bank may be incorporated into the timing phase detector in a very efficient manner. A multitude of timing phase detectors have appeared in the open literature: the ML detector [13], [14, Chap. 6]; the Mueller and Müller detector [19] and its variants [15], [25]–[27]; the wave difference method [28], [29]; and the digital data transition tracking loop [30]–[34]. Any of these phase detectors could be used with the polyphase filterbank interpolator. For the purpose of demonstration, we focus on the ML techniques and their approximations to show the ease with which these techniques can be realized. Three possibilities are illustrated in Fig. 5. The polyphase implementation of the early–late gate detector is shown in Fig. 5(a). The outputs of polyphase filter stages immediately preceding and following the current filter stage are used to form the phase error defined by (7) with $\Delta\tau/T_s = 1/MN$. This approximation requires three polyphase stage filter outputs for each matched filter output (two for the early–late-gate error and one for the actual matched filter output). The preceding and following polyphase filter outputs can also be used to form the first central difference which approximates the derivative of the matched filter output at the current filter stage as illustrated in Fig. 5(b). The matched filter output and its derivative are then combined to form the ML error signal (5). Again, this error detector requires three polyphase stage outputs for each matched filter output. Instead of using two filters to compute the derivative matched filter output $\dot{y}(\tau)$, a single filter, whose coefficients are given by

$$\dot{h}_m(nT_s) = h_{m+1}(nT_s) - h_{m-1}(nT_s) \qquad m = 1, 2, \ldots, M-2 \tag{17}$$

$$\dot{h}_0(nT_s) = h_1(nT_s) - h_{M-1}(nT_s) \tag{18}$$

$$\dot{h}_{M-1}(nT_s) = h_{M-2}(nT_s) - h_0(nT_s) \tag{19}$$

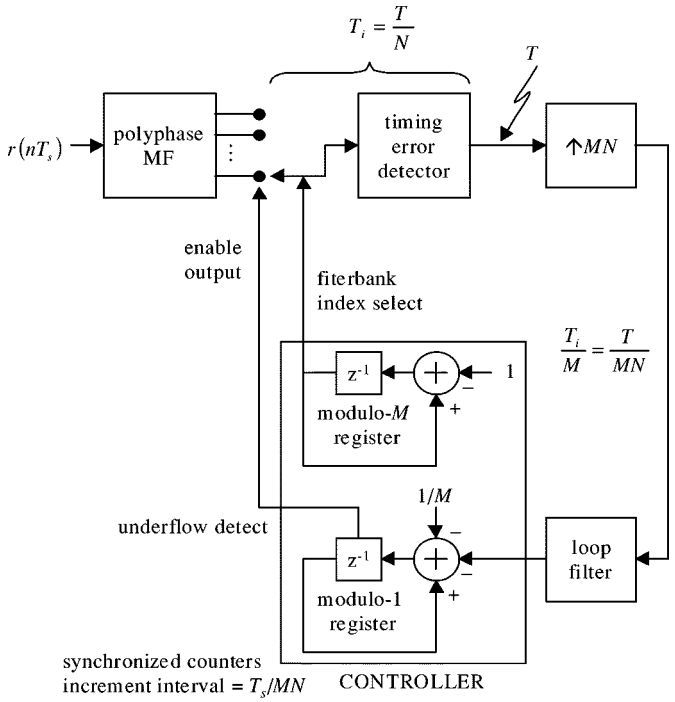


Fig. 6. Loop control architecture using a loop that operates at $MN$ samples/symbol. Interpolation control uses two counters synchronized to the data sample clock. The top counter is a modulo-$M$ counter that cycles through the polyphase filter indexes. The bottom counter is a modulo-1 counter. A new interpolant from the polyphase matched filter is taken when this register overflows.

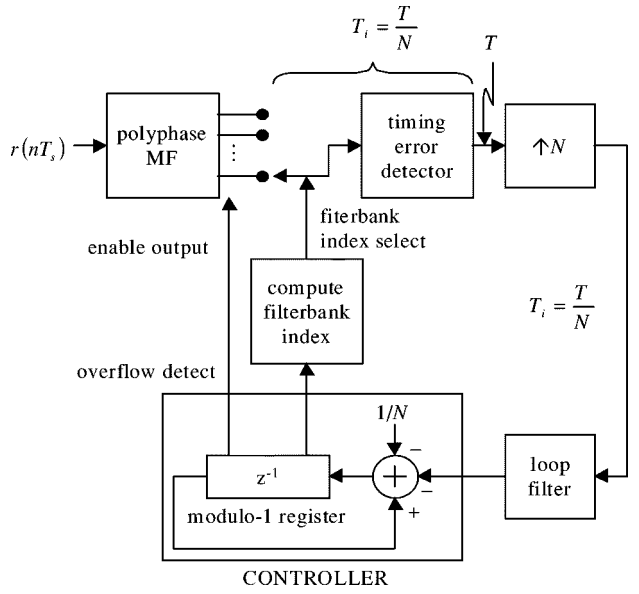


Fig. 7. Loop control architecture using a loop that operates at $N$ samples/symbol. Interpolation control is based on the modulo-1 counter that functions as the NCO as described in [1]. A new interpolant from the polyphase matched filter is taken when the register overflows. The contents of the register on overflow are used to compute the polyphase filter index.

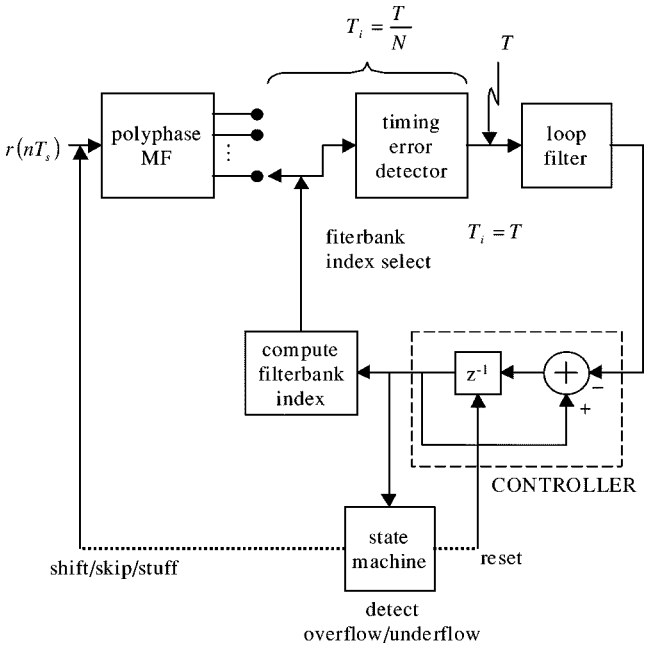


Fig. 8. Loop control architecture using a loop that operates at 1 sample/symbol. A new interpolant from the polyphase matched filter for approximately each $N$ input samples. When the register overflows or underflows, the proper interpolation interval has "rolled around" the end of the polyphase filter. This is detected by the state machine that "skips" or "stuffs" data samples at the polyphase filter input.

can be used [3]. This method is illustrated in Fig. 5(c) where $\dot{h}(nT_s)$ is the derivative matched filter. This structure requires only two polyphase filter stages for each matched filter output.

### C. Loop Control Architectures

Each loop has four essential components: the polyphase matched filter, the timing error detector, the loop filter, and the controller. In each case, the polyphase matched filter operates on samples arriving every $T_s$ seconds. The timing error detector operates on the matched filter outputs which are output from the polyphase filterbank every $T/N$ seconds and outputs a timing error every $T$ seconds or once per sample corresponding to the current estimate of the optimum sampling instant.[3] The output of the timing error detector is possibly upsampled and used to drive the loop filter and loop controller. There are three natural possibilities for the loop control when using an $M$-stage polyphase matched filter: The loop filter and controller can be run at $MN$ samples/symbol, $N$ samples/symbol, or 1 sample/symbol. Each choice results in a slightly different architecture as illustrated in Figs. 6–8.

The first architecture, illustrated in Fig. 6, is perhaps the first architecture that comes to mind. Interpolation control is performed by a pair of counters whose increments are synchronized to the input sample clock with period $MT_s$. The top counter cycles through the polyphase filter indexes $N$ times per symbol and is designed to point to the right polyphase filterbank filter at the interpolation instant. The lower counter is a decrementing modulo-1 counter with underflow period approximately $T_i = T/N$. The period of the underflow is altered by the output of the loop filter to align every $N$th underflow event with the maximum eye opening. The underflow condition indicates that an interpolant should be computed and passed on the timing error detector. When the loop is in lock, the underflow condition will be detected when the polyphase index is pointing to the filterbank filter with the sample corresponding to the optimum sampling time. The disadvantage of this approach is that the loop

[3]Note that we assume a rate change in the operation of the timing error detector.

must run at a rate $MN$ times higher than the symbol rate. In many high speed systems, this is not a possibility.

The second architecture, shown in Fig. 7, is essentially the one described in [1] where the matched filter and interpolator have been combined into a single polyphase filter. In this case, the loop operates at $N$ samples/symbol and requires only a single decrementing modulo-1 counter. The counter underflows with period nominally $T/N$ or $N$ times per symbol. Counter underflow enables the output of the polyphase matched-filter filterbank. The corresponding polyphase index $m$ is computed by quantizing the desired fractional interpolation interval $\mu$ to the nearest multiple of $1/M$. The fractional interpolation interval $\mu$ is determined from the register contents at underflow as specified in [1].

The third architecture, shown in Fig. 8, operates at 1 sample/symbol. In this architecture, the controller consists of a register whose contents are interpreted as a value in the range 0 to $M-1$. The desired polyphase filter index is a quantized version of the register contents. When the loop is in lock for the case $T_s = T/N$, the outputs of the timing error detector and loop filter are nearly zero so that the contents of the register do not change and the filterbank index remains constant. In this mode, $N$ input samples are processed by the polyphase matched filter to produce $N$ matched filter outputs during each symbol period. The timing error detector uses the $N$ matched filter outputs to update the timing error once per symbol. The timing error is filtered and used to update the control register.

When the loop is in lock for the case $T_s \neq T/N$, the outputs of the timing error detector and loop filter are not zero and the contents of the register increase or decrease at a rate governed by the frequency offset the sample clock and the symbol clock. Eventually, the register will overflow or underflow. This condition indicates that an adjustment must be made in the output sample clock as summarized in Fig. 9.

The polyphase filter index pointer shifts down (to enable successively more delay) when the time samples are too close as is the condition when the sample rate exceeds $N$ times the symbol rate. The spacing of output samples increases by slowly drifting successive samples to the right of their preassigned positions, thus increasing the sample period and hence reducing the sample rate to match the symbol rate. When the pointer reaches the end of the weights list [the last filter, filter $h_{M-1}(nT_s/M)$, in the polyphase partition], the pointer continues the drift by rolling around to the beginning of the filter weight list [the first filter, $h_0(nT_s/M)$, in the polyphase partition]. This roll over is an overflow of the pointer index that must occur periodically when the input sample rate exceeds the symbol rate. Pointer index overflow occurs when the input clock has accumulated one more input sample than output sample. We arrange to keep the input and output clocks aligned, by discarding or "skipping" the next input sample. This process is illustrated in the top plot of Fig. 9 for the case $M = 4$.

In a similar manner, the polyphase filter index pointer shifts down (to enable successively less delay) when the time samples are too far apart as is the condition when the sample rate is less than $N$ times the symbol rate. The spacing of output samples decreases by slowly drifting successive samples to the left of their preassigned positions, thus decreasing the sample period

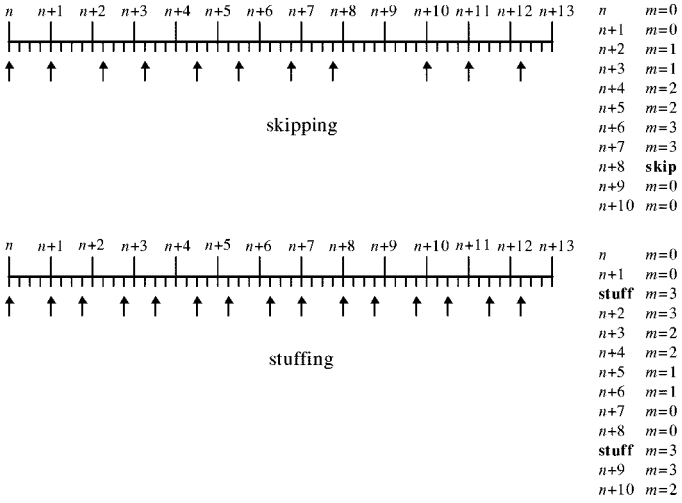


Fig. 9. Graphical illustration of the "skipping" (top) and "stuffing" (bottom) operations performed by the state machine controller in Fig. 8. The illustration is for $M = 4$ polyphase matched filterbank.

and hence increasing the sample rate to match the symbol rate. When the pointer reaches the end of the weights list [the first filter, filter $h_0(nT_s/M)$ in the polyphase partition], the pointer continues the drift by rolling around to the other end of the filter weight list [the last filter, filter $h_{M-1}(nT_s/M)$, in the polyphase partition]. This roll over is an underflow of the pointer index that must occur periodically when the input sample rate is less than $N$ times the symbol rate. Pointer index underflow occurs when the output clock has accumulated one more output sample than input sample. We arrange to keep the input and output clocks aligned, by repeating or "stuffing" the last input sample. This process is illustrated in the lower plot of Fig. 9 for the case $M = 4$.

The "skipping" and "stuffing" of input samples at overflow and underflow is controlled by the state machine as indicated in Fig. 8. The state machine detects the overflow/underflow condition and resets the register while controlling the corresponding skip/stuff operation. The skipping and stuffing of input samples at the overflow and underflow boundaries of the filter indexing scheme results in small phase jitter perturbations in the filter output time series. This jitter is easily observed in decision error of an equalizer and in the phase error of the carrier recover loop that appear downstream of the polyphase matched filter and timing recovery process. These jitter terms can be made acceptably small by increasing the number of stages in the polyphase partition. The "skipping" and "stuffing" operations are performed automatically in loops that operate at $N$ or $MN$ samples/symbol. This is because the interpolation instant is determined by the underflow condition of the controller NCO. Skipping and stuffing is achieve by occasional adjustment to the number of samples between underflow events. Thus no separate state machine is required to manage the difference in clock rates.

## V. Simulation Results

Closed-loop simulation results are presented to demonstrate the main features of polyphase filterbanks for timing synchronization. The principle of operation does not depend on the type of timing error detector. The examples presented in this section

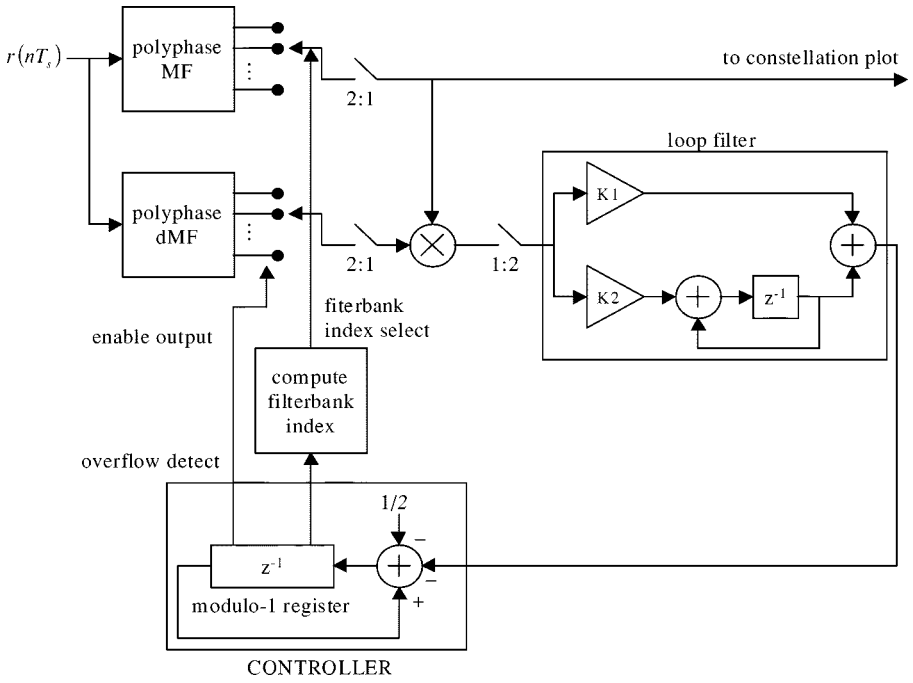


Fig. 10. Complete symbol timing synchronizing PLL operating at 2 samples/symbol using polyphase filterbanks.

use the ML timing error detector (low SNR approximation) to illustrate the ease with which the polyphase filterbank can be incorporated into the phase detector. The first simulated system is illustrated in Fig. 10. The modulation is QPSK with square-root raised cosine pulse shapes with 25% excess bandwidth. Data samples at approximately $N = 2$ samples/symbol are processed by the polyphase matched filter (MF) and polyphase derivative matched filter (dMF) filterbanks each with $M = 32$ stages. The product of the two filterbank outputs form the timing error which is updated once per symbol. The timing error signal is upsampled by 2 and filtered by a proportional-plus-integrator loop filter which is required for a second-order loop to track out the symbol clock frequency offset [35]. The loop filter output is used to control the increment in the counter (NCO) which underflows at the optimum timing instant when the loop has achieved lock.

In all simulations summarized in this paper, the gain of the timing error detector[4] was normalized to unity so that the loop characteristics were determined solely by the loop filter constants $K1$ and $K2$. The filter constants were chosen to produce a critically damped loop with a closed-loop single-sided noise bandwidth of 0.5% of the symbol rate.

Simulation results illustrating the step response for the closed-loop system are illustrated in Figs. 11 and 12. In these plots and the plots that follow, the dark smooth lines are the timing error, NCO control, and polyphase index for alternating data and the gray lines are plots for random data with equally probable symbols. The gray lines illustrate the effect of modulation noise on loop performance (no additive channel noise was included in the simulations). The phase step was $T/4$.

[4]Following [1], the gain of the timing error detector is the slope of the timing error output for small timing errors. For the ML detector, the timing error detector gain is a function of the pulse shape and the transition statistics of the data. For example, the timing error detector has a much higher gain for alternating data than for random data with equally probable signs.

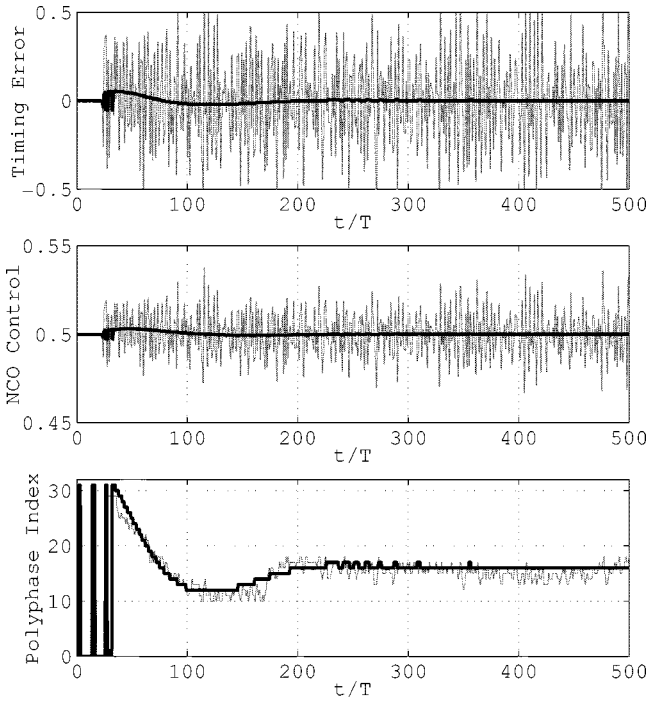


Fig. 11. Simulation results: Phase step response for a timing synchronization loop operating at 2 samples/symbol with an ML timing error detector and a 32-stage polyphase matched filter and derivative matched filter.

The three plots in Fig. 11 demonstrate how the loop chooses successively larger filterbank indexes to successively increase the filterbank delay until the filterbank delay matches the delay in the data. Once the filterbank delay matches the clock delay, the polyphase index settles to a constant steady-state value (16 in this case) and the NCO control settles to approximately 1/2 since the loop is running at 2 samples/symbol. Loop performance can also be characterized by the resulting signal

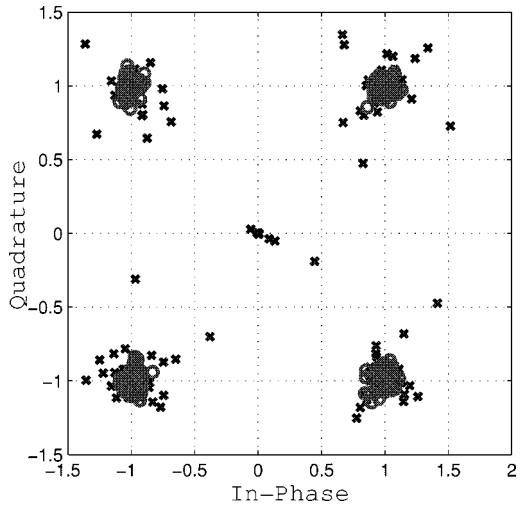


Fig. 12. Signal space projections for the same simulation summarized in Fig. 11. The Xs are the first 200 projections and the Os are the last 800 projections.

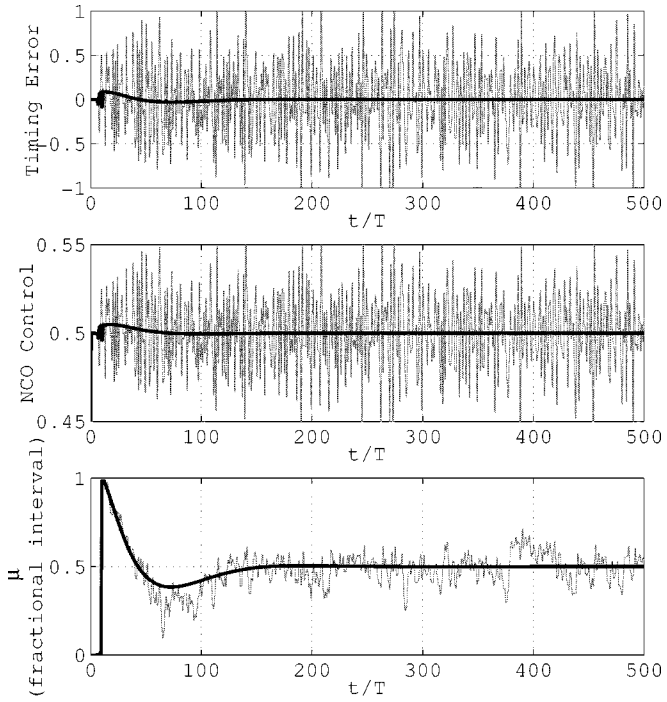


Fig. 13. Simulation results: Phase step response for a timing synchronization loop operating at 2 samples/symbol with an ML timing error detector and a Farrow interpolation filter.

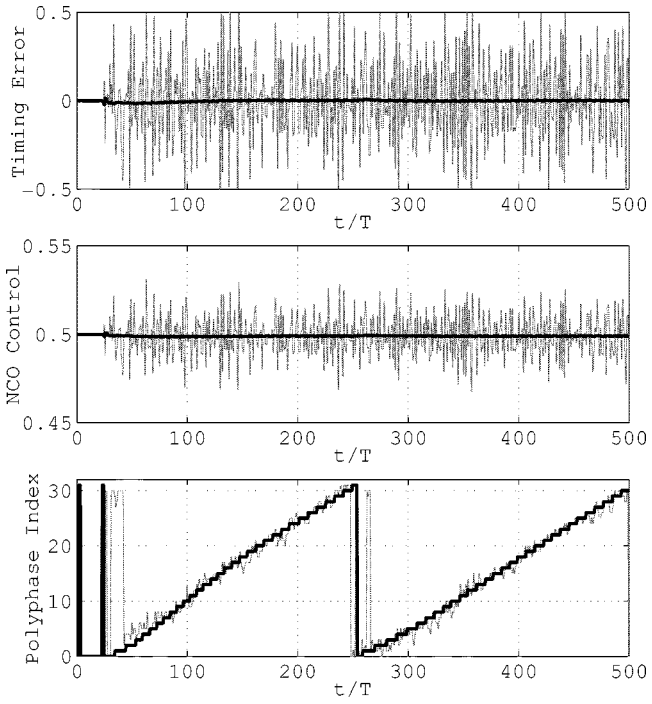


Fig. 14. Simulation results: Frequency step response for a timing synchronization loop operating at 2 samples/symbol with an ML timing error detector and a 32-stage polyphase matched filter and derivative matched filter.

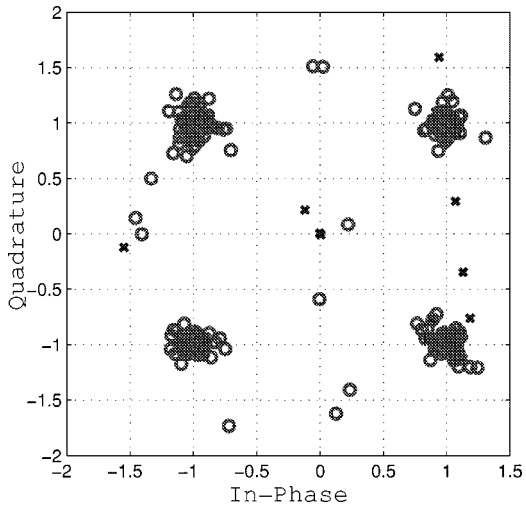


Fig. 15. Signal space projections for the same simulation summarized in Fig. 14. The Xs are the first 200 projections and the Os are the last 800 projections.

space projections shown in Fig. 12. The first 200 signal space projections are indicated by the Xs and are scattered throughout the signal space as the loop transients settle. The last 800 signal space projections are indicated by the Os and form tight clusters around the true constellation points. Variance from the true position once the loop has locked is due to timing jitter caused by the modulation noise.

It is interesting to compare this simulation to a loop using a separate polynomial-based interpolation filter such as those described in [2]. A Farrow structure based on a piecewise parabolic interpolator with $\alpha = 1/2$ was simulated for the comparison. These results are presented in Fig. 13. The behavior of the fractional interval $\mu$ agrees with that reported in [2]) closely follows the behavior of the polyphase index in the bottom plot of Fig. 11.

Simulation results illustrating the ramp response (i.e., frequency step response) are illustrated in Figs. 14 and 15. Data with a simulated sample clock offset 1/250 of the symbol clock was input to the loop. Thus, the symbols appear to slide through the data samples. As a consequence, the optimum sampling instant slides as well and does so at a rate equal to the timing offset. This behavior is observed in polyphase index illustrated in the lower plot of Fig. 14. Since the sample clock period $T_s$ is less than $2T$, the optimum sample time delays increase with

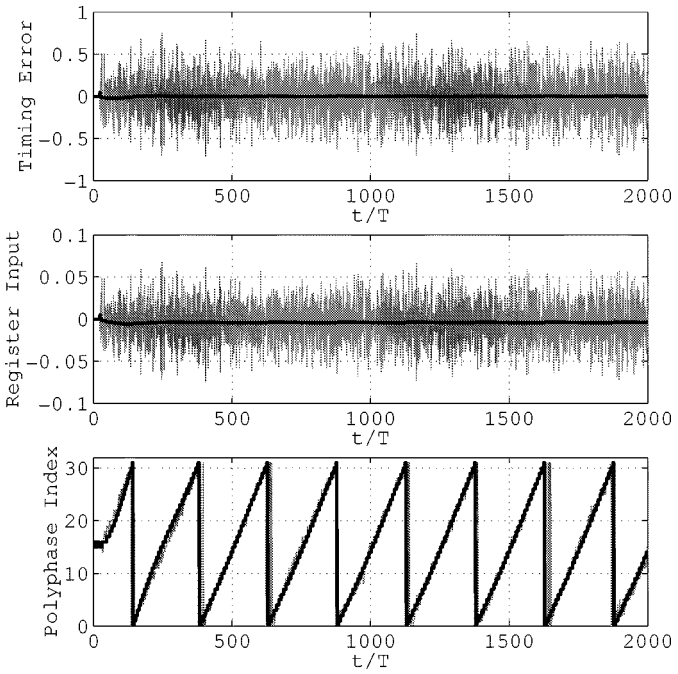


Fig. 16. Simulation results: Frequency step response for a timing synchronization loop operating at 1 sample/symbol with an ML timing error detector and a 32-stage polyphase matched filter and derivative matched filter.

each sample. The loop tracks this by increasing the polyphase filter index. The sequence of polyphase indexes repeats every 250 symbols as a result of the symbol clock offset. This behavior matches exactly the behavior of the fractional interval $\mu$ reported in [2] for the loop using a separate polynomial-based interpolation filter. The corresponding signal space projections are shown in Fig. 15 and confirm that the loop is tracking the clock offset.

As a final simulation example, we present simulation results for a loop operating at one sample/symbol to illustrate the stuff and skip operations described in the previous section. The basic loop architecture is shown in Fig. 8. For this simulation, we used the same timing error detector and loop filter as in the previous examples. (The loop filter constants were recomputed to account for the 1 sample/symbol loop operation.) For brevity, we only present the simulation results for the frequency step. These results are presented in Fig. 16. When the loop is locked, the input the control register (middle plot) is just slightly greater than zero. This causes the register to increase with each symbol clock cycle. The corresponding polyphase index output clearly illustrates this behavior (lower plot). Eventually, the register (and the corresponding polyphase index) reaches the maximum and overflow. This happens because the sample clock $T_s$ is less than twice the data period $2T$ so that the matched filter has accumulated one too many input samples. This extra sample is discarded by the "skipping" operation described in the previous section. In this example, input samples are skipped at symbols 144, 383, 630, 880, 1130, 1380, 1630, and 1880. Thus we observe that in steady state, an input sample is skipped once every 250 symbols which is the rate by which the sample clock exceeds twice the symbol clock. If the sample clock frequency had been less than $2/T$, the polyphase filter index would decrease and the control register would underflow every 250 symbols.

## VI. Concluding Remarks

In this paper, we have reviewed the development of the polyphase filterbank and have shown how to apply it to perform the interpolations required for symbol timing synchronization. The polyphase filter bank possesses two advantages over the synchronizers described in [1]–[3] that require a separate interpolation filter apart from the matched filter. First, by using the matched filter as the interpolation filter, the polyphase filter bank does not require an additional interpolation filter. Since both approaches require a matched filter operating at $N$ samples/symbol, the complexity reduction is the complexity of the interpolation filter. The second advantage lies in the direct and natural way that ML timing synchronization and its early–late gate approximation can be incorporated into the polyphase filterbank.

Two simulation examples for loops operating at 2 samples/symbol were presented to illustrate the step response and ramp response of the loop. These examples illustrate how the polyphase index is equivalent to the fractional interpolation interval described in previous work. A third simulation example demonstrated operation of the loop at one sample/symbol and how state machine control is required to "skip" and "stuff" input samples to compensate for sample clock frequency offset.

## References

[1] F. M. Gardner, "Interpolation in digital modems—Part I: Fundamentals," *IEEE Trans. Commun.*, vol. 41, pp. 501–507, Mar. 1993.

[2] L. Erup, F. M. Garnder, and R. A. Harris, "Interpolation in digital modems—Part II: Implementation and performance," *IEEE Trans. Commun.*, vol. 41, pp. 998–1008, June 1993.

[3] U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*. New York: Plenum, 1997.

[4] H. Meyr, M. Moenecleay, and S. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. New York: Wiley, 1998.

[5] R. E. Crochiere, L. R. Rabiner, and R. R. Stively, "A novel implementation of digital phase shifters," *Bell Syst. Techn. J.*, vol. 54, no. 8, pp. 1497–1502, Oct. 1975.

[6] R. Crochiere and L. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[7] P. Vaidyanathan, "Miltirate digital filters, filter banks, polyphase networks, and applications: A tutorial," *Proc. IEEE*, vol. 78, pp. 56–93, Jan. 1990.

[8] ——, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[9] F. J. Harris and T. Weschselberger, "Multirate all-digital modems for transport of universal multiplex transport layer for digital compression," in *Proc. Nat. Cable Television Association (NCTA) Conf.*, San Francisco, CA, June 1993.

[10] J. Tiernan, F. Harris, and D. Becker, "Digital receiver for variable rate symbol synchronization," Patent US5 504 785, Apr. 1996.

[11] C. Dick, M. Rice, and F. J. Harris, "Synchronization in software radios: Carrier and timing recovery using FPGA's," in *Proc. IEEE Symp. Field Programmable Custom Computing Machines*, Napa Valley, CA, Apr. 2000.

[12] F. J. Harris, "Multirate digital filters used for timing recovery in digital receivers," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, 2000, pp. 246–251.

[13] L. E. Franks, "Carrier and bit synchronization in data communication—A tutorial review," *IEEE Trans. Commun.*, vol. 28, pp. 1107–1121, Aug. 1980.

[14] J. G. Proakis, *Digital Communications*, fourth ed. New York: McGraw-Hill, 2001.

[15] A. Jennings and B. R. Clarke, "Data-sequence selective timing recovery for PAM systems," *IEEE Trans. Commun.*, vol. 33, pp. 729–731, July 1985.

[16] W. G. Cowley and L. P. Sabel, "The performance of two symbol timing recovery algorithms for PSK demodulators," *IEEE Trans. Commun.*, vol. 42, pp. 2345–2355, June 1994.
[17] A. N. D'Andrea and M. Luise, "Optimization of symbol timing recovery for QAM data demodulators," *IEEE Trans. Commun.*, vol. 44, pp. 399–406, Mar. 1996.
[18] D. W. Paranchych and N. C. Beaulieu, "Performance of a digital symbol synchronizer in cochannel interference and noise," *IEEE Trans. Commun.*, vol. 48, pp. 1945–1954, Nov. 2000.
[19] K. H. Mueller and M. Müller, "Timing recovery in digital synchronous data receivers," *IEEE Trans. Commun.*, vol. 34, pp. 423–429, May 1976.
[20] J. Stahl, G. Ascheid, M. Oerder, and H. Meyr, "An all digital receiver architecture for bandwidth efficient transmission at high data rates," *IEEE Trans. Commun.*, vol. 37, pp. 804–813, Aug. 1989.
[21] C. Dick and F. Harris, "Configurable logic for digital communication: Some signal processing perspectives," *IEEE Commun. Mag.*, pp. 107–111, Aug. 1999.
[22] F. J. Harris and C. Dick, "On structure and implementation of algorithms for carrier and symbol synchronization in software defined radios," in *Proc. EUSIPCO*, Tampere, Finland, Sept. 2000.
[23] S. Srikanteswara, J. H. Reed, P. Athanas, and R. Boyle, "A soft radio architecture for reconfigurable platforms," *IEEE Commun. Mag.*, pp. 140–147, Feb. 2000.
[24] M. Cummings and S. Haruyama, "FPGA in the software radio," *IEEE Commun. Mag.*, vol. 37, no. 2, pp. 108–112, Feb. 1999.
[25] C.-J. Tzeng, D. A. Hodges, and D. G. Messerschmitt, "Timing recovery in digital subscriber loops using baudrate sampling," in *Proc. IEEE Int. Conf. Communications*, vol. 3, 1985.
[26] M. A. Marsan and S. Benedetto, "Timing and carrier recovery for high speed data transmission over telephone channels," in *Proc. Nat. Telecommunications Conf.*, vol. 2, 1979.
[27] M. A. Marsan, G. Albertengo, and S. Benedetto, "High speed modem with microprocessors: Design and implementation of the synchronization algorithms," in *Proc. Nat.Telecommunications Conf.*, vol. 1, 1981.
[28] T. Suzuki, H. Takatori, M. Ogawa, and K. Tomooka, "Line equalizer for a digital subscriber loop employing switched capacitor technology," *IEEE Trans. Commun.*, vol. 30, pp. 2074–2082, Sept. 1982.
[29] O. Agazzi, C.-P. J. Tzeng, D. G. Messerschmitt, and D. A. Hodges, "Timing recovery in digital subscriber loops," *IEEE Trans. Commun.*, vol. 33, pp. 558–569, June 1985.
[30] W. C. Lindsey and M. K. Simon, *Telecommunication Systems Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1973.
[31] W. J. Hurd and T. O. Anderson, "Digital transition tracking symbol synchronizer for low SNR coded systems," *IEEE Trans. Commun.*, vol. 18, pp. 141–147, Apr. 1977.
[32] M. K. Simon, "Optimization of the performance of a digital-data-transition tracking loop," *IEEE Trans. Commun.*, vol. 18, pp. 686–689, Sept. 1970.
[33] A. E. Payzin, "Analysis of a digital bit synchronizer," *IEEE Trans. Commun.*, vol. 31, pp. 554–559, Apr. 1983.
[34] H. Brügel and P. F. Driessen, "Variable bandwidth DPLL bit synchronizer with rapid acquisition implemented as a finite state machine," *IEEE Trans. Commun.*, vol. 42, no. 9, pp. 2751–2759, September 1994.
[35] F. M. Gardner, *Phaselock Techniques*. New York: Wiley, 1979.

**Fredric J. Harris** (M'72–SM'72) received the B.S. degree from the Polytechnic Institute of Brooklyn, Brooklyn, NY, in 1961, the M.S. degree from San Diego State University, San Diego, CA, in 1967, and the Ph.D. degree from the University of California, San Diego, in 1973, all in electrical engineering.

He holds the CUBIC Signal Processing Chair of the Communication Systems and Signal Processing Institute at San Diego State University where he has taught since 1967 in areas related to Digital Signal Processing and Communication Systems. He holds a number of patents on digital receiver and DSP technology and lectures throughout the world on DSP applications. He consults for organizations requiring high performance DSP systems. He is well published and has contributed to a number of books on DSP.

In 1990 and 1991, Dr. Harris was the Technical and then the General Chair of the Asilomar Conference on Signals, Systems, and Computers that meets annually in Pacific Grove, CA.

**Michael Rice** (M'82–SM'98) received the B.S.E.E. degree from Louisiana Tech University, in 1987, and the Ph.D. degree from the Georgia Institute of Technology, Atlanta, in 1991.

He was with Digital Transmission Systems, Inc., in Atlanta and joined the faculty at Brigham Young University in 1991, where he is currently an Associate Professor in the Department of Electrical and Computer Engineering. He was a NASA/ASEE Summer Faculty Fellow at the Jet Propulsion Laboratory during 1994 and 1995 where he worked on land mobile satellite systems. During the 1999 to 2000 academic year, he was a Visiting Scholar at the Communication Systems and Signal Processing Institute at San Diego State University. His research interests include digital communication theory and error control coding with a special interest in applications to telemetering and software radio design. He has been a consultant to both government and industry on telemetry related issues.

Dr. Rice was Chair of the Utah Section of the IEEE from 1997 to 1999 and is currently Chair of the Signal Processing and Communications Society Chapter of the Utah Section and the IEEE Region 6 Student Activities Coordinator.