

FAST SYMBOL SYNCHRONIZER

Field of the Invention

[0001] The present invention generally relates communication systems. In particular, the invention relates to a system and method for implementing a fast symbol synchronizer using both feedback and feedforward structures in communications receivers.

Background of the Invention

[0002] Data recovery of transmitted communication signals at digital receivers typically require the implementation of at least three processes: 1) timing recovery for symbol synchronization; 2) carrier recovery; and 3) equalization. Timing recovery is the process by which the receiver clock is synchronized to the transmitter clock. Thus permitting the received signal to be sampled at an optimum point in time to reduce the chances of errors being associated with the retrieval of the data as well as the processing of the received symbol values. In some receivers, the received signal is sampled at a multiple of the transmitted symbol rate. For example some receivers sample the received signal at twice the transmitter symbol rate. The sampling clock of the receiver must be synchronized with that of the transmitter.

[0003] Many existing timing recovery modules or sometimes referred to as symbol synchronizers in communications receivers, are not designed for both simultaneous fast timing acquisition and low residual timing phase error performances. This is true particularly for receivers that are based on feedback loop principles.

[0004] The two requirements of fast timing acquisition and low residual timing phase error are mandatory for “bursty” communication systems which

intermittently transmit and receive short data packets, such as Time-Division-Multiple-Access(TDMA) based systems. Failure to address these requirements will degrade the receiver's Bit Error Rate (BER) performance causing the receiver to fail compliance tests during product qualification trials.

[0005] In communication systems design, having a symbol synchronizer to simultaneously achieve fast timing acquisition and low residual timing phase error performances, often requires the trading off of low residual timing error for fast timing acquisition. Although the final design may still meet minimum specifications, this however has the effect of compromising the receiver's overall BER performance and could disadvantageously result in a inferior communication system design.

[0006] Symbol synchronizers are found in all communications receivers ranging from the low-cost Bluetooth radios to high-end advanced satellite receivers. In these communication systems, data is segmented into packets each comprising three main parts: a preamble (or training) sequence, a header (system critical information), and an information (or data) sequence. The preamble sequence carries a special non-user bit sequence for the receiver to achieve fast timing acquisition. Timing acquisition is used to describe a process by which the symbol synchronizer rapidly transitions from an unsynchronized state to a near synchronized state. Once timing acquisition is established, the receiver will proceed to decode the header and then subsequently the information sequence.

[0007] The information sequence contains user data payload which may be voice, video or other data information. While decoding the payload, the symbol synchronizer will also continue to correct and minimize any residual timing errors associated with each data bit. The symbol synchronizer essentially first performs fast timing acquisition during the preamble sequence while attempting to reduce the

residual timing error for the remaining information sequence so as to achieve an overall superior Bit Error Rate (BER) performance.

[0008] Many symbol synchronization techniques have been devised by researchers over the decades and may be broadly classified into *feedback* and *feed-forward* configurations.

[0009] In feedforward structures, the symbol synchronizer first employs a timing estimator for estimating a symbol timing error during the preamble sequence and uses that information in concert with an interpolator to compensate the received data samples for timing error. Some variants of this technique may also require significant memory storage buffers during this processing stage. The main advantages of feedforward configuration are fast timing acquisition and the absence of any feedback-related problems due to “cycle slipping” and “false locked”. It however suffers from the main drawback of poorer tracking performance. An example of this class of synchronizer is described in publication “Digital filter and square timing recovery” by H. Meyr et. al. IEEE Transaction on Communications, Vol 36, May 1988. The poor tracking performance results in a larger residual timing error due to its inherent lack of tracking capability.

[0010] The inability of a feedforward based symbol synchronizer to track timing error resulted in a symbol synchronizer that cannot adaptively compensate for timing drift due to the possible frequency mismatch of transmitter and receiver oscillators. As such, feedforward based symbol synchronizers are frequently employed when preamble sequences are very short and when the header containing system critical information has to be rapidly decoded without error in a very short time frame. It should be noted that the critical information is normally sandwiched between the preamble sequence and data sequence. The Header contains important system critical parameters for receivers to know in advance of information related

to payload, such as modulation, error correction, packet length etc for proper receiver functioning. If the critical information is not decoded in time, the received data packet has to be abandoned and communications may fail to take place.

[0011] In a feedback configuration, the symbol synchroniser utilises a feedback mechanism based on phase-locked-loop (PLL) principle to reduce the timing error of the received symbols. This configuration typically comprises a timing error detector, a loop filter and an interpolator connected in a feedback manner for compensating the received symbol samples for timing offset. Such methods have been widely published an example of which is in the publication “ A BPSK/QPSK Timing Error Detector for Sampled Receivers” by F.M. Gardner, IEEE Transaction on Communications Vol. COM-34, No 5, MAY 1986. This method has the advantage of good tracking capability for minimising effects due to timing phase error or drift; low latency due to the minimal use of storage buffers during processing; and low implementation complexity.

[0012] However, one main disadvantage associated with feedback based symbol synchronizers is the inferior timing acquisition performance due to slow recursive nature of feedback loops. This has the potential effect of affecting subsequent data detection processes. Furthermore, feedback configurations have been known to suffer from “hang up” phenomena resulting in a sequence of fatal bit errors.

[0013] While each of the feedback and feedforward configurations are capable of partially solving problems with the two conflicting requirement by tweaking their design.

[0014] Feedforward symbol synchronizer attain lower residual timing errors during payload by buffering larger number of signal samples (compare to preamble and header sequence) during the payload sequence before computing the timing estimate

and correcting the error. This can be done by having two sets of buffer with different size. One buffer of smaller size will be used during the preamble/header sequence for achieving fast but coarse timing acquisition, while the larger buffer is used for payload sequence it is slightly less time critical.

[0015] However, there is a limit to the size of buffers used during the payload sequence, depending on the maximum system latency that the receiver can tolerate. A larger buffer would disadvantageously mean a longer processing time, thus hindering the proper reception of subsequent packets.

[0016] In a feedback symbol synchronizer faster acquisition performance can be improved by having two sets of loop filter coefficients. During timing acquisition, one set of pre-computed coefficients is pre-loaded into the loop filter to enable the phase-locked-loop to lock rapidly closer to the steady state. When the receiver is about to receive the user payload, the second set of coefficients are loaded into the loop filter. The two sets of coefficients will thus allow the PLL to achieve a lower residual error.

[0017] However, even with the first set of loop-filter coefficients, the PLL may not be able to lock rapidly due to the occurrences of “hangup” phenomenon. There is also a limit to how fast the timing acquisition can actually be achieved via the first set of loop filter coefficient due to the inherent clumsy and unreliable feedback nature of PLL timing acquisition.

[0018] It can thus be seen that there exists a need for a simple and robust way for a system and method for implementing a fast symbol synchronizer that can overcome or at least alleviate the disadvantages of the existing prior art.

Summary of the Invention

[0019] The present invention seeks a system and method for implementing a fast symbol synchronizer .

[0020] Accordingly, in one aspect, the present invention provides, a system for fast symbol synchronization in a communications system, the system comprising:
a timing recovery module adapted for receiving a data stream comprising a plurality of symbols, the timing recovery module further adapted for implementing a feedback algorithm for producing a plurality of timing recovered symbols;
a timing estimator adapted for implementing a feedforward algorithm for estimating an estimated timing error from the data stream; and
a formatter adapted for producing an initialization value from the estimated timing error;
wherein the timing recovery module receives the initialization value for implementing the feedback algorithm for producing the plurality of timing recovered symbols.

[0021] In another aspect, the present invention provides, a method for fast symbol synchronization in a communications system using a timing recovery module adapted for implementing a feedback algorithm for producing a plurality of timing recovered symbols and a timing estimator adapted for implementing a feedforward algorithm for estimating an estimated timing error from the input data stream, the method comprising the steps:

- a) receiving input data stream;
- b) implementing a feedforward algorithm to produce an estimated timing error from the input data stream;
- c) generating an initialization value for timing recovery by the timing recovery module using the estimated timing error; and

d) implementing a feedback algorithm to produce an output data stream using the initialization value.

[0022] In another yet aspect, the present invention provides, a timing estimator of a fast symbol synchronizer in a communications system for receiving an input data stream comprising a plurality of received symbols and for producing an estimated timing error, the timing estimator comprising:

a CORDIC processor for performing an arctangent function to generate the estimated timing error;

a first processor path for receiving and processing an in-phase component of the input data stream; and

a second processor path for receiving and processing the quadrature-phase component of the input data stream.

Brief Description of the Drawings

[0023] A present embodiment of the present invention will now be more fully described, with reference to the drawings of which:

[0024] FIG.1 illustrates a block diagram of the system in accordance with the present embodiment;

[0025] FIG.2 illustrates a block diagram of a timing estimator of the system of FIG.1;

[0026] FIG.3 illustrates a block diagram of a timing recovery module of the system of FIG.1;

[0027] FIG.4 illustrates a block diagram of a timing error detector of the timing recovery module of FIG.3;

[0028] FIG.5 illustrates a block diagram of a loop filter of the timing recovery module of FIG.3;

[0029] FIG.6 illustrates a block diagram of a Numerically Control Oscillator of the timing recovery module of FIG.3;

[0030] FIG.7 illustrates a graph showing the relationship between a fictitious time-continuous $nco(t)$ and time;

[0031] FIG.8 illustrates a graph showing the relationship between fractional interval and the estimated timing error;

[0032] FIG.9 illustrates a flowchart of a method in accordance with the present embodiment;

[0033] FIG.10 illustrates a flowchart for a generating initialization value step of FIG.9; and

[0034] FIG.11 illustrates a flowchart for an implementing feedback algorithm step of FIG.9.

Detailed description of the Drawings

[0035] A system and method for implementing a fast symbol synchronizer with a preferred embodiment is described. In the following description, details are

provided to describe the preferred and the alternate embodiment. It shall be apparent to one skilled in the art, however that the invention may be practiced without such details. Some of the details may not be described at length so as not to obscure the invention.

[0036] The present embodiment proposes a system for fast symbol synchronization capable of rapidly performing “phase lock” to decode system critical information in the header sequence but also continue to “track out” the residual timing error during payload.

[0037] Referring to FIG.1, a system 10 for fast symbol synchronization in accordance to the present embodiment comprising a timing estimator 30 and a timing recovery module 20, both adapted for receiving an input data stream 5 comprising of a plurality of received symbols. The timing estimator 30 receives the data stream and implements a feedforward algorithm for estimating an estimated timing error $\hat{\tau}$ 35 from the input data stream 5.

[0038] The system 10 further comprises of a formatter 50 which is adapted for producing an initialization value *nco_initial* 55 from the estimated timing error $\hat{\tau}$ 35 from the timing estimator 30.

[0039] The initialization value *nco_initial* 55 is then fed back into the timing recovery module 20. The timing recovery module 20 implementing a feedback based algorithm utilizes the initialization value *nco_initial* 55 together with the data stream 5 to produce an output data stream 27 comprising of a plurality of timing recovered symbols. The timing recovered symbols are then sent to subsequent stages in the communication system for demodulation and data recovery.

[0040] In the present embodiment, the timing estimator 30, the timing recovery module 20 and the formatter 50 all operate on a T/2 architecture. The system 10 would advantageously consume less power than a conventional system which uses conventional timing estimator based on T/4 architecture (i.e. 4 times the symbol rate).

[0041] The received data stream 5 may be represented by a complex signal

$$r(t) = x(t - \tau)e^{j\phi} + n(t) \quad (a)$$

where

$$r(t) = r_I(t) + jr_Q(t) \quad (b)$$

where $r_I(t)$ and $r_Q(t)$ are in-phase and quadrature components of the complex signal respectively.

[0042] A square-root-raised-cosine filter (SRRC) is a shaping filter commonly found in many Phase-Shift-Keying (PSK) based receivers. A received complex signal from the SRRC shaping filter may be represented by:

$$x(t) = \sum_{m=-\infty}^{\infty} a_m p(t - mT) \quad (c)$$

where a_m is the complex PSK symbols with $\text{abs}(a_m) = 1$; ϕ is receiver phase offset; $p(t)$ is the raised cosine filter; $\tau \in (T/2, T/2)$ is the symbol timing delay; and $n(t)$ is complex AWGN noise.

[0043] The timing estimator 30 is thus able to estimate the actual timing delay τ by using the above equations. Considering sampling of the input data stream 5 at twice the symbol rate ($t=kT_s$), where T_s is the sampling duration $T/2$, the timing estimator 30 may be represented by:

$$\frac{\hat{\tau}}{T} = \frac{-1}{2\pi} \arg \left\{ \sum_{k=1}^{2L} \left[|r(kT_s)|^2 e^{-jk\pi} + \text{Re} \left[r(kT_s) r^*((k-1)T_s) \right] e^{-j(k-0.5)\pi} \right] \right\} \quad (d)$$

where $Re(\cdot)$ is the real part of a complex number and L is the number of symbols used during the estimation process. The operator "arg" (argument or arctangent) extracts the phase angle (radian) of the accumulated complex number. An arctangent operator can be efficiently implemented by using a CORDIC processor.

[0044] Referring to Equation (d), there are two main terms involved in a summation process. This summation process may be achieved using an adder. In the first term, there is a product of an absolute-squared term and a complex exponential term. Depending on value of k , the complex exponential product $\exp(jk\pi)$ is evaluated into 1 and -1 which are interpreted as a sign inversion. The absolute-squared $|r(kTs)|^2$ term, may be given as $r_I(k)^2 + r_Q(k)^2$, can be designed with two multipliers and an adder. The resultant product for the first term is therefore a real number.

[0045] In the second term, there are two products comprising a complex exponential and a complex-conjugate multiplication with only a real (Re) part considered. The complex exponential is reduced into $j \cdot \exp(jk\pi)$, or j and $-j$. Note that at each k th instant, the signs of the two complex exponential are the same. The real part of complex-conjugate product is readily re-expressed as $r_I(k) \cdot r_I(k-1) + r_Q(k) \cdot r_Q(k-1)$ and can be designed with two multipliers and an adder.

[0046] The timing estimator 30 of the present embodiment is based on that of prior art design and reference may be made to the publication by Seung Joon Lee, "A New Non-Data-Aided Feedforward Symbol Timing Estimator using two samples per symbol", IEEE Communications Letters, Vol.6, No.5, May 2002.

[0047] Referring to FIG.2 a block diagram showing an implementation of an embodiment of the timing estimator 30 is shown. The arctangent function may be

implemented by a CORDIC processor 280. The timing estimator 30 of the present embodiment comprises two processor paths.

[0048] A first processor path for receiving and processing the in-phase component $r_I(k)$ 5a and a second processor path for receiving and processing the quadrature-phase component $r_Q(k)$ 5b of the received input data stream 5.

[0049] The first processor path comprises a first delay element 205 having an input for receiving the in-phase component $r_I(k)$ 5a. The output 205a of the first delay element 205 is coupled to a first multiplier 210. The in-phase component $r_I(k)$ 5a is further coupled to the first multiplier 210 for multiplication of the in-phase component $r_I(k)$ 5a and the output 205a which is the previous in-phase component $r_I(k)$ 5a delayed by the first delay element 205.

[0050] A second multiplier 220 is coupled to two inputs of the in-phase component $r_I(k)$ 5a, thus effectively producing an output 220a of the second multiplier 220 equal to the square the value of the in-phase component $r_I(k)$ 5a.

[0051] The second processor path comprises a third multiplier 250 for receiving two inputs of the quadrature-phase component $r_Q(k)$ 5b. The output 250a of the third multiplier is thus the square value of the quadrature-phase component $r_Q(k)$ 5b.

[0052] The second processor path further comprises of a second delay element 240 for receiving the input of the quadrature-phase component $r_Q(k)$ 5b. The output 240a of the second delay element 240 is a delayed signal of the input quadrature-phase component $r_Q(k)$ 5b. The output 240a of the second delay element 240 is then coupled to a fourth multiplier 245 which is further coupled to receive another input of the quadrature-phase component $r_Q(k)$ 5b. The output 245a of the fourth

multiplier 245 is the multiplication of the quadrature-phase component $r_Q(k)$ 5b with the previous quadrature-phase component $r_Q(k)$ 5b delayed by the delay element 240.

[0053] The output 210a of the first multiplier 210 is coupled to a first adder 215 for adding to the output 245a of the fourth multiplier 245.

[0054] The output 250a of the third multiplier 250 is coupled to a second adder for adding to the output of the second multiplier 220. The output 255a of the second adder 255 is coupled to a fifth multiplier 260 for multiplying with a set value 265a from a multiplexor 265. The set value is either 1 or -1 and is controlled the multiplexor 265 which is controlled by external control means.

[0055] The set value 265a is further coupled to a sixth multiplier 225 for multiplying with an output 215a of the first adder 215. The output 225a of the sixth multiplier 225 is then coupled to a third adder 230 for adding the output 225a of the sixth multiplier 225 to an output 235a of a third delay element 235. The third delay element 235 receiving input from the output of the third adder 230.

[0056] The output 260a of the fifth multiplier 260 is further coupled to a fourth adder 270 for adding the output 260a of the fifth multiplier 260 to an output 275a of a fourth delay element 275. The fourth delay element 275 receiving input from the output 270 of the fourth adder 270.

[0057] The output 235a of the third delay element 235 and the output 275a of the fourth delay element 275 are coupled to the CORDIC processor 280 for performing an arc tangent calculation. The output from the CORDIC processor 280 is thus the estimated timing error $\hat{\tau}$ 35.

[0058] Referring to FIG.3, the timing recovery module 20 further comprises an interpolator 22, a timing error detector 26, a loop filter 25, a Numerically Controlled Oscillator (NCO) 24 and a Decimator 21.

[0059] The interpolator 22 receives the data stream 5 comprising of the plurality of symbols and constructs timing-recovered samples comprising a plurality of interpolated symbols 23 $x(k)$ by interpolating a “fractional interval” at an interpolation point provided by the NCO 24. The interpolation point from the NCO 24 is based on a “fractional interval” control signal $\mu(k)$ 24a, where k represents the k_{th} interpolation point.

[0060] This technique has the advantage of moderate design complexity and acceptable “aliasing” performance. The Interpolator is also operating at twice the symbol rate ($T/2$), the plurality of interpolated symbols 23 $x(k)$ is therefore also running at twice the symbol rate.

[0061] The interpolator 22 of the present embodiment is based on that of prior art design and reference may be made to the publication by F.M. Gardner, “Interpolation in Digital Modem – Part 2: Implementation and Performance”, IEEE Transaction on Communications; Vol 41, No.6, March 1993.

[0062] The interpolator 22 may be based on a cubic polynomial function or a piecewise parabolic function. In conventional practice, the parabolic function is preferred over the cubic polynomial due to ease of implementation, while maintaining performance comparable with other interpolators.

[0063] While an interpolator utilizing a linear parabolic function is less complex than interpolators utilizing other functions, it suffers from slightly poorer performance. However, the overall BER performance of the linear interpolator is

still acceptable particularly for lower orders of modulations. The interpolator that gives the best BER performance, while offering the least complex implementation, uses the piecewise parabolic function. Hence, the piecewise parabolic function is preferred over other functions in practical implementation, especially for lower-order modulation signals.

[0064] The decimator 21 is then strobed at an optimum instant to reduce the plurality of interpolated symbols 23 $x(k)$ at $T/2$ sampling rate producing the output data stream 27 comprising of the plurality of timing recovered symbols at T symbol rate. The output data stream 27 comprising of a plurality of timing recovered symbols will then be sent to subsequent stages for demodulation and data recovery. The decimator 21 is activated by an “underflow” control signal (UDF) 24b from the NCO 24.

[0065] The timing error detector 26 receives the plurality of interpolated symbols 23 and generates an error signal $e(k)$ 26a which indicates whether sampling of the input data stream 5 is advanced or delayed relative to an optimum sampling point. The optimum sampling point is the point at which the communications receiver is synchronous to the transmitter sampling point. Since the error signal $e(k)$ 26a is typically noisy, the Loop filter 25 is used to smoothen the error signal $e(k)$ 26a producing a filtered error signal $\delta(k)$ 25a thereby controlling transient response. An example of the timing error detector 26 may be found in the publication by F.M. Gardner, “A BPSK/QPSK Timing-Error detector for sampled Receivers”, IEEE Transactions on Communications, Vol. Com-34, Bo.5, May 1986..4

[0066] The filtered error signal $\delta(k)$ 25a is then utilized by the NCO 24 together with the initialization value *nco_initial* 55 from the formatter 50 to generate the “fractional interval” control signal $\mu(k)$ 24a.

[0067] Compared to traditional non-interpolation based timing recovery technique that uses Analog Digital Converter (ADC), one key advantage of this interpolation-based timing recovery technique is that it obviates the need for an ADC where its sampling clock has to be adjusted synchronously to the transmitter's sampling clock.

[0068] Referring to FIG.4, the timing error detector 26 of the timing recovery module 20 is shown. The in-phase $x_I(k)$ 23a and quadrature-phase $x_Q(k)$ 23b components of the plurality of interpolated symbols $x(k)$ 23 are fed to the timing error detector 26. The final output from the timing error detector 26 is the error signal $e(k)$ 26a.

[0069] The timing error detector 26 comprises two stages for processing the plurality of interpolated symbols $x(k)$ 23. A first stage processes the in-phase $x_I(k)$ 23a components and a second stage processes the quadrature-phase $x_Q(k)$ 23b components.

[0070] The first stage of the timing error detector comprises a first delay element 405 in series with a second delay element 410. The in-phase component $x_I(k)$ 23a is sent to the first delay element 405 and the second delay element 410. The output of the second delay element 410 is then subtracted from the in-phase component $x_I(k)$ 23a by a first subtractor 415 resulting in a first subtracted output 415a.

[0071] A output 405a of the first delay element 405 and the first subtracted output 415a are multiplied by a first multiplier 420 to yield a result of a first stage output 420a.

[0072] The second stage of the timing error detector similarly comprises a third delay element 425 in series with a fourth delay element 430. The quadrature-phase

$x_Q(k)$ 23b is sent to the third delay element 425 and the fourth delay element 430. The output of the fourth delay element 430 is then subtracted from the quadrature-phase $x_Q(k)$ 23b by a second subtractor 435 resulting in a second subtracted output 435a.

[0073] A output 425a of the third delay element 425 and the second subtracted output 435a are multiplied by a second multiplier 440 to produce a second stage output 440a.

[0074] At a final step, the first stage output 420a and the second stage output 440a are added together by an adder 450 to produce the desired timing error signal $e(k)$ 26a.

[0075] The delay elements in the timing error detector 26 are substantially similar and delay the signals by one clock cycle which is half the sampling rate.

[0076] The Loop Filter 25 of the timing recovery module 20 receives the error signal $e(k)$ 26a from the timing error detector 26 and produces a filtered error signal $\delta(k)$ 25a.

[0077] Referring to FIG.5, the error signal $e(k)$ 26a is coupled to a first multiplier 505 and a second multiplier 510. A loop proportional constant $K1$ and the error signal $e(k)$ 26a are multiplied by the first multiplier 505 to produce a proportional modified error signal 505a.

[0078] Another loop integral constant $K2$ and the error signal $e(k)$ 26a are multiplied by the second multiplier 510 to produce a integral modified error signal 510a. The integral modified error signal 510a is further coupled to a delayed signal adder 515.

[0079] The delayed signal adder 515 adds the integral modified error signal 510a to a delayed added error signal 520a to produce an added error signal 515a. The delayed added error signal 520a is a feedback of the added error signal 515a after having been delayed by an error delay element 520. The error delay element 520 is thus coupled to an output of the delayed signal adder 515 and an input of the delayed signal adder 515.

[0080] The proportional modified error signal 505a and the added error signal 515a are then added in a loop filter adder 530 to produce the filtered error signal $\delta(k)$ 25a.

[0081] The NCO 24 is adapted to receive the inputs of the initialization value *nco_initial* 55 and the filtered error signal $\delta(k)$ 25a to generate the “fractional interval” control signal $\mu(k)$ 24a and also the “underflow” control signal (UDF) 24b.

[0082] Referring to FIG.6, the NCO 24 comprises a control adder 605 coupled to a modulo-two accumulator 650.

[0083] The filtered error signal $\delta(k)$ 25a and a normalizing constant 610 are added by the control adder 605 to produce a control word $W(k)$ 605a. The normalizing constant in the present embodiment has a value of 1.0.

[0084] The control adder 605 is coupled to the modulo-two accumulator 650. The modulo-two accumulator 650 receives the input of the control word $W(k)$ 605a and generates the “fractional interval” control signal $\mu(k)$ 24a to be used by the interpolator 22 and also the “underflow” control signal (UDF) 24b to be sent to the timing error detector 26.

[0085] The modulo-two accumulator 650 further comprises a control subtractor 615, a modulo-two module 620, a sampler 630 and a NCO register 625.

[0086] The NCO register 625 receives a first input of the initialization value *nco_initial* 55, this only occurs once for the initialization of the NCO 24. The initialization value *nco_initial* 55 is then treated as a register stored signal 625a from the NCO register 625 and sent to the control subtractor 615.

[0087] The control word $W(k)$ 605a is then subtracted from the register stored signal 625a by the control subtractor 615 to produce a subtracted control signal 615a. The subtracted control signal 615a is then sent to the modulo-two unit 620 where the modulo-two function is performed to monitor the subtracted control signal 615a for an underflow condition. The underflow condition occurs when the value of the subtracted control signal 615a goes below zero. The subtracted control signal 615a is then sent into the NCO register 625 and overwrites the previous value stored in the register as the register stored signal 625a.

[0088] The register stored signal 625a is again sent to the control subtractor 615, and the cycle starts all over again.

[0089] When the underflow condition is detected, the modulo-two unit 620 sends the “underflow” control signal (UDF) 24b to the timing error detector 26. The “underflow” control signal (UDF) 24b is also sent to the sampler 630. The sampler 630 then samples and captures the register stored signal 625a and treats it as the “fractional interval” control signal $\mu(k)$ 24a that is sent to the interpolator 22.

[0090] The “underflow” control signal (UDF) 24b is also used for strobing the decimator 21 at near optimum timing to yield the timing recovered symbols of the output data stream 27. Note that the interpolator 22 still outputs two interpolated

samples per symbol as required for proper operation by the timing error detector 26.

[0091] The formatter 50 receives the estimated timing error $\hat{\tau}$ 35 from the timing estimator 30 and generates an initialization value *nco_initial* 55 for the timing recovery module 20.

[0092] The formatter 50 detects the signs of the estimated timing error $\hat{\tau}$ 35 and computes the initialization value *nco_initial* 55 depending on the signs of estimated timing error $\hat{\tau}$ 35.

[0093] If the estimated timing error $\hat{\tau}$ 35 is positive, then the initialization value *nco_initial* 55 will be computed based on a value that is twice the estimated timing error 35.

[0094] However, if the estimated timing error $\hat{\tau}$ 35 is negative, then the initialization value *nco_initial* 55 will output a value that is a sum of twice the estimated timing error $\hat{\tau}$ 35 and a constant value of 2.

[0095] If the estimated timing error $\hat{\tau}$ 35 is equal to zero, then the initialization value *nco_initial* 55 is also equal to zero. A pseudo-code for the aforementioned implementation of Formatter 50 is given as follows:

If $\hat{\tau} > 0$ then

$$nco_initial = 2 * \hat{\tau}$$

else if $\hat{\tau} < 0$ then

$$nco_initial = 2 + 2 * \hat{\tau}$$

else

```

nco_initial = 0
end

```

[0096] In deriving the initialization value *nco_initial* 55 generated by the formatter 50, the relationship between the register stored signal 625 and the “fractional interval” control signal $\mu(k)$ 24a and the estimated timing error $\hat{\tau}$ 35.

[0097] When considering the publication by F.M. Gardner, “Interpolation in Digital Modem – Part 2: Implementation and Performance”, IEEE Transaction on Communications; Vol 41, No.6, March 1993, the extraction of the “fractional interval” control signal $\mu(k)$ 24a may be calculated from the content of the NCO register 625.

[0098] Designating the NCO register 625 contents computed at the k th clock cycle as $nco(k)$, and the NCO control word as $W(k)$, the NCO difference equation in F.M. Gardner’s publication may be given as

$$nco(k) = [nco(k-1) - W(k-1)] \bmod 1 \quad (1)$$

[0099] However, the present embodiment was modified slightly affording a simplification to the above equation. Instead of “mod 1” operation, a “mod 2” operation is used, and the modified NCO difference equation is represented by:

$$nco(k) = [nco(k-1) - W(k-1)] \bmod 2 \quad (2)$$

[00100] In the present embodiment, the NCO 24 is decrementing allowing for minor simplification in computing of the “fractional interval” control signal $\mu(k)$ 24a.

[00101] The control word $W(k)$ 605a which is likely to be a positive fraction is adjusted by the timing recovery module 20 so that the output of the interpolator 22 is sampled at near optimal timing. Under loop equilibrium conditions, the control word $W(k)$ 605a will likely be in close range of a constant value. In the present embodiment, with the earlier modification of replacing mod1 with mod2, the constant value the control word $W(k)$ 605a is in close range with is the value 1.

[00102] Referring to FIG.7, a graph illustrates a plot of fictitious time-continuous $nco(t)$ versus continuous time is extracted from F.M. Gardner's publication. Referring to FIG.7, kT_s is the time of a sample clock pulse immediately preceding the k th interpolation time $(k+u(k))T_s$. The register stored signal 625a decreases through zero at the interpolation time and the zero crossing (resulting in the "underflow" control signal (UDF) 24b being activated) becomes known at the next clock cycle at time $(k+1)T_s$. The register stored signal 625a, $nco(k)$ and $nco(k+1)$ are then available at the next clock cycle.

[00103] Applying concept of similar triangles in FIG.7, it can be seen [3] that

$$\frac{u(k)Ts}{nco(k)} = \frac{(1-u(k))Ts}{2-nco(k+1)} \quad (3)$$

which can be solved for $u(k)$ as

$$u(k) = \frac{nco(k)}{2-nco(k+1)+nco(k)} = \frac{nco(k)}{W(k)} \quad (4)$$

[00104] As estimate of $u(k)$ 24a can be obtained by performing the indicated division of the two numbers $nco(k)$ and $W(k)$ 605a that are both available from the NCO 24. The above equation (4) is an estimate of the exact $u(k)$ 24a because its constituents $W(k)$ 605a and $nco(k)$ are both estimates of the true timing frequency and phase.

[00105] The division in equation (4) can be further simplified by approximating $W(k)$ 605a with the value 1. This simplification is still able to achieve successful loop tracking performance. This value of 1 occurs only during loop equilibrium. In general $W(k)$ 605a is not equal to 1 during the loop acquisition phase. Therefore the final expression of the estimated $u(k)$ 24a which will be used by the interpolator 22 is

$$u(k) = nco(k) \quad (5)$$

[00106] It should be noted that the reason why the expression (5) can be used successfully to generate the fractional interval $u(k)$ 24a for the interpolator is because the timing estimator 30 of the present embodiment is used in conjunction with the timing recovery module 20 to estimate the initialization value *nco_initial* 55 which is close enough to “jump start” the timing recovery module 20 into equilibrium mode.

[00107] Referring to FIG.8 and FIG.1, a method for fast symbol synchronization in a communications system using a timing recovery module 20 adapted for implementing a feedback algorithm for producing a plurality of timing recovered symbols 27 and a timing estimator adapted for implementing a feedforward algorithm for estimating the estimated timing error $\hat{\tau}$ 35 from an input data stream comprising a plurality of received symbols, starts with the step of receiving 910 an input data stream 5.

[00108] Next, the step of implementing 915 a feedforward algorithm to produce an estimated timing error from the input data stream 5 is performed. Following which, an initialization value is generated 920 for timing recovery by the timing recovery module 20 using the estimated timing error $\hat{\tau}$ 35.

[00109] Next the step of implementing 925 a feedback algorithm to produce an output data stream 27 using the initialization value *nco_initial* 55 is performed by the timing recovery module 20.

[00110] Referring to FIG.9, the step of generating 920 the initialization value *nco_initial* 55 further starts with the step of receiving 1010 the estimated timing error $\hat{\tau}$ 35. Next, the initialization value *nco_initial* 55 is set 1020 zero if the estimated timing error $\hat{\tau}$ 35 is equal to zero 1015. If however, the estimated timing error $\hat{\tau}$ 35 is positive 1025, then the initialization value 55 is set 1035 to twice the estimated timing error $\hat{\tau}$ 35. If however, the timing error 35 is negative 1025, then the initialization value *nco_initial* 55 is set to sum of twice the estimated timing error $\hat{\tau}$ 35 and a constant value of two.

[00111] Referring to FIG.10 and FIG.3, the step of implementing feedback algorithm further starts with the step of the NCO 24 receiving 1110 the initialization value *nco_initial* 55. Next, the step of generating 1115 the “fractional interval” control signal $\mu(k)$ 24a using the initialization value *nco_initial* 55 and the error signal $e(k)$ 26a is performed.

[00112] The error signal $e(k)$ 26a is usually filtered by the loop filter 25 to produce the filtered error signal $\delta(k)$ 25a which is used by the NCO 24.

[00113] Next, the step of producing the plurality of interpolated symbols 23 using the “fractional interval” control signal $\mu(k)$ 24a is performed.

[00114] Following which, the plurality of interpolated symbols 23 are decimated 1125 by the decimator 21 to generate the output data stream 27 using the “underflow” control signal (UDF) 24b from the NCO 24.

[00115] It will be appreciated that various modifications and improvements can be made by a person skilled in the art without departing from the scope of the present invention.

Claims

1. A system for fast symbol synchronization in a communications system, the system comprising:

a timing recovery module adapted for receiving a data stream comprising a plurality of symbols, the timing recovery module further adapted for implementing a feedback algorithm for producing a plurality of timing recovered symbols;

a timing estimator adapted for implementing a feedforward algorithm for estimating an estimated timing error from the data stream; and

a formatter adapted for producing an initialization value from the estimated timing error;

wherein the timing recovery module receives the initialization value for implementing the feedback algorithm for producing the plurality of timing recovered symbols.

2. The system of claim 1, wherein the timing recovery module comprises:

an interpolator for receiving the data stream comprising a plurality of symbols, and for producing a plurality of interpolated symbols using a fractional interval control signal; and

a decimator for reducing the plurality of interpolated symbols to produce a output data stream comprising a plurality of timing recovered symbols;

3. The system of claim 2, wherein the timing recovery module further comprises:

a timing error detector for receiving the plurality of interpolated symbols and for generating an error signal indicating whether sampling of the input data stream is advanced or delayed relative to an optimum sampling point;

a loop filter for filtering the error signal to produce a filtered error signal;

a Numerically Controlled Oscillator (NCO) for receiving the filtered error signal and for generating the fractional interval control signal using the initialization value;

the NCO further for generating an underflow control signal for instructing the loop filter, the timing error detector and the decimator.

4. The system of claim 3, wherein the error detector further comprises:

a first stage for processing an in-phase component of the plurality of interpolated symbols from the interpolator; and

a second stage for processing an quadrature-phase of the plurality of interpolated symbols from the interpolator.

5. The system of claim 4, wherein the first stage further comprises:

a first delay element for receiving the in-phase component, the first delay element in series with a second delay element,

wherein an output of the second delay element is then subtracted from the in-phase component by a first subtractor producing a first subtracted output;

further wherein an output of the first delay element and the first subtracted output are multiplied by a first multiplier to produce a first stage output.

6. The system of claim 5, wherein the second stage further comprises:

a third delay element for receiving the quadrature component, the third delay element in series with a fourth delay element,

wherein an output of the third delay element is then subtracted from the quadrature component by a second subtractor producing a second subtracted output;

further wherein an output of the third delay element and the second subtracted output are multiplied by a second multiplier to produce a second stage output.

7. The system of claim 6, wherein the first stage output and the second stage output are added to produce the error signal.

8. The system of claim 3, wherein the loop filter comprises:

a first multiplier for multiplying the error signal with a loop proportional constant for producing a first proportional modified error signal;

a second multiplier for multiplying the error signal with a integral proportional constant for producing a integral modified error signal;

a delayed signal adder for adding the integral error signal to a delayed error signal for producing an added error signal;

wherein the delayed error signal is a feedback of the added error signal after delay by an error delay element;

further wherein the proportional modified error signal and the added error signal are then added in a loop filter adder producing the filtered error signal.

9. The system of claim 3, wherein the NCO comprises:

a control adder coupled to a modulo-two accumulator;

wherein a normalizing constant is added to the filtered error signal by the control adder producing a control word.

10. The system of claim 9, wherein the modulo two accumulator comprises:

a control subtractor for subtracting a register stored signal from the control word producing a subtracted control signal;

a modulo-two module for performing a modulo-two function on the subtracted control signal for monitoring the subtracted control signal for an underflow condition;

a NCO register for storing the subtracted control signal and for overwriting a previously stored subtracted control signal;

wherein the underflow control signal is generated by the modulo-two module to instruct a sampler for sampling and capturing the register stored signal and sending the register stored signal to the interpolator as the fractional interval control signal.

11. The system of claim 1, wherein the initialization value is twice the estimated timing error, when the estimated timing error is positive.

12. The system of claim 1, wherein the initialization value is sum of twice the estimated timing error and a constant value of two, when the estimated timing error is negative.

13. The system of claim 1, wherein the initialization value is zero, when the estimated timing error is zero.

14. A method for fast symbol synchronization in a communications system using a timing recovery module adapted for implementing a feedback algorithm for producing a plurality of timing recovered symbols and a timing estimator adapted for implementing a feedforward algorithm for estimating an estimated timing error from the input data stream, the method comprising the steps:

- a) receiving input data stream;
- b) implementing a feedforward algorithm to produce an estimated timing error from the input data stream;
- c) generating an initialization value for timing recovery by the timing recovery module using the estimated timing error; and
- d) implementing a feedback algorithm to produce an output data stream using the initialization value.

15. The method of claim 14, the step c) further comprising:
 - c1) receiving the estimated timing error;
 - c2) setting the initialization value to zero if the estimated timing error is equal to zero;
 - c3) setting the initialization value to twice the estimated timing error if the estimated timing error is positive; and
 - c4) setting the initialization value to sum of twice the estimated timing error and a constant value of two if the estimated timing error is negative.
16. The method of claim 14, wherein the step d) further comprising:
 - d1) receiving the initialization value;
 - d2) generating a fractional interval control signal using the initialization value and an error signal by a Numerically Contolled Oscillator (NCO);
 - d3) producing a plurality of interpolated symbols from the input data stream using the fractional interval control signal; and
 - d4) decimating the plurality of interpolated symbols to generate an output data stream using an underflow control signal generated by the NCO.
17. A timing estimator of a fast symbol synchronizer in a communications system for receiving an input data stream comprising a plurality of received symbols and for producing an estimated timing error , the timing estimator comprising:
 - a CORDIC processor for performing an arctangent function to generate the estimated timing error;
 - a first processor path for receiving and processing an in-phase component of the input data stream; and
 - a second processor path for receiving and processing the quadrature-phase component of the input data stream.
18. The timing estimator of claim 17, wherein the first processor path comprises:
 - a first delay element for receiving the in-phase component, the first delay element having an output coupled to a first multiplier;

the first multiplier further coupled to the in-phase component for multiplication of the in-phase component and the output of the first delay element;

a second multiplier having two inputs coupled to the in-phase component for producing an output equal to square of the in-phase component.

19. The timing estimator of claim 18, wherein the second processor path comprises:

a third multiplier having two inputs coupled to the quadrature-phase component for producing an output equal to square of the quadrature-phase component;

a second delay element for receiving the input of the quadrature-phase component, the second delay element having an output coupled to a fourth multiplier

the fourth multiplier coupled to the quadrature-phase component for multiplication multiplication of the quadrature-phase component with the quadrature-phase component from the second delay element.

20. The timing estimator of claim 19, further comprising

a first adder coupled to the output of the first multiplier and the output of the fourth multiplier;

a second adder coupled to the output of the third multiplier and the output of the second multiplier, the output of the second adder being coupled to a fifth multiplier for multiplying with a set value from a multiplexor, the set value being either 1 or -1 ;

the set value being further coupled to a sixth multiplier for multiplying with an output of the first adder;

a third adder coupled to the output of the sixth multiplier and an output of a third delay element, the third delay element receiving input from the output of the third adder;
and

a fourth adder coupled to the output of the fifth multiplier and an output of a fourth delay element, the fourth delay element receiving input from the output of the fourth adder;

21. The timing estimator of claim 20, further comprising:
the output of the third delay element and the output of the fourth delay element
being coupled to the CORDIC processor.