Development of Algorithm for Demodulator for Processing Satellite Data Communication

K. R. Nataraj *, Dr S. Ramachandran and Dr B. S. Nagabushan *****

* M. G..R. University, Chennai, India
** National Academy of Excellence, Bangalore, 560-060, India
*** Sanlab Technologies, Bangalore, 560-038, India

Summary
Quadrature Phase Shift Keying (QPSK) is the preferred modulation scheme for satellite data communication. A two stage estimation scheme for the demodulator for processing satellite data is proposed in this paper, where carrier frequency estimation is followed by timing recovery under training. Frequency offset estimation is performed by finding the slope of the best fit line through the unwrapped phase. Since this carrier frequency estimation technique gives an approximate estimate of the frequency offset, there still remains some frequency offset to be corrected. This is done in the second stage, where least mean square algorithm is employed to track the variations in frequency and estimating the symbols. Estimation of the correct sampling instant is based upon finding the minima of the absolute of the received training pattern and then calculating the maximum energy instant from it. An overview of the whole system architecture is presented and its performance is evaluated. BER curves for different frequency and timing offsets are also presented. Some tasks in the system may serve as bottlenecks to higher data rates owing to their computationally intensive nature. These tasks are identified and based upon that, the proposed algorithm is partitioned into two parts. One part is to be implemented on FPGA and consists of high computational complexity modules and the other is to be coded on a DSP processor.

Keywords:
Algorithm, Demodulator, Quadrature Phase Shift Keying, Transponder, Linear algebra, Architecture, Field Programmable Gate Arrays.

1. Introduction

A communication satellite functions as an overhead wireless repeater station that provides a microwave communication link between two geographically remote sites. Due to its high altitude, satellite transmissions can cover a wide area over the surface of the earth. What makes a communications satellite different from other satellites is its payload, usually a radio repeater consisting of a transmitter/receiver combination called a transponder. A transponder is a communications repeater for radio signals received on board the satellite, processed and then retransmitted to an earth station. A transponder is generally defined by its bandwidth capacity, its available effective isotropic radiated power and its on-board processing capabilities.

1.1 Advanced On-Board Processing (OBP) Satellite Systems

A repeater satellite only changes frequency and power parameters on a digitally modulated signal. It makes no attempt to detect the digital data. That means that if there is distortion on the uplink signal, this distortion will be amplified and translated in frequency to the downlink, thus lowering the quality of the signal even more. This has a negative impact on the received downlink signal power and results in inferior bit error rate (BER). Neither the user nor the repeater satellite can improve that signal, although coding may help to improve the bit error rate performance. Due to these limiting factors, new types of transponders were conceived, which allow a number of different approaches to avoid these problems. None of them is currently able to solve all the problems, since the limitations are different for each application. These types of transponders are used on "smart" satellites called On-Board Processing (OBP) satellites, since they process in various ways the up-linked signals before repeating them on the downlink.
To increase the data processing rate without impairing the performance is the main focus of the present work. This problem can be solved in two ways. First, this is accomplished by making the signal processing algorithms computationally efficient, so that they need minimum hardware for their implementation, without significant loss in BER performance. Applying linear algebra techniques such as iterative, sub-optimal schemes or exploiting the structure of the algorithms achieves the algorithmic modifications. These algorithms, typically working on matrix data sets, have significant levels of parallelism. This inherent parallelism and bit-level nature of the

Manuscript received June 5, 2009
Manuscript revised June 20, 2009

computations can be exploited to achieve high data processing speeds by an efficient architectural design. Modifying the algorithm for a computationally efficient solution and exploiting its parallelism using a suitable architectural design are needed to achieve high-speed data processing. Second method to increase the data processing speed is by employing high speed and low power computational engines to implement these algorithms. Most of the signal processing is done in digital domain than in analog domain, due to the inherent limitations of analog implementations, such as high temperature sensitivity, sensitivity to DC offsets, DC voltage drifts, amplifier/mixer non-linearity, susceptible to noise, etc.
DSP algorithms can be implemented in software or hardware. In software implementation, a DSP processor such as ADSP-21020 is used. In hardware implementation, Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs) are used. FPGA devices provide nearly the throughput of custom ASICs while maintaining full flexibility of a DSP processor. This is because FPGAs can be reprogrammed to reconfigure the entire DSP implementation on-the-fly. Some functions work faster on Digital Signal Processors too. Therefore, a proper implementation strategy is required to enhance the overall processing speed of the onboard processing system. Such a strategy is proposed in this work, which will be elaborated in Section 3.

1.2 Conventional Satellite Demodulator Receiver

A conventional digital demodulator employs a DSP processor such as ADSP-21020. The received signal from the antenna is conditioned in the RF front end before the ADC. The input to the ADC is a modulated signal at a standard intermediate frequency of 455 KHz and the ADC samples it at a rate of 1.536 MHz. This sampled signal is then processed in ADSP-21020 to recover the data bits. The main drawback of this system is its low throughput (64 K bits/second). Algorithmic and architectural techniques mentioned earlier need to be applied in order to increase the throughput as has been done in the present work.
Quadrature Phase Shift Keying (QPSK) is the preferred modulation scheme for satellite data communication. A two-stage estimation scheme is presented, where carrier frequency estimation is followed by timing recovery under training. Frequency offset estimation is performed by finding the slope of the best-fit line through the unwrapped phase. Since this carrier frequency estimation technique gives an approximate estimate of the frequency offset, there still remains some frequency offset to be corrected. This is done in the second stage, where least mean square algorithm is employed to track the variations in frequency and estimating the symbols. Some tasks in the system may serve as bottlenecks to higher data rates, due to their compute intensive nature. These tasks are identified and, based on these tasks; the algorithm is partitioned into two parts. One part, that is computationally intensive, is to be implemented on FPGA and the other part is to be coded on a DSP processor. A number of DSP techniques can be applied to reduce the computational complexity of the selected tasks, thus easing their hardware implementation.
In the next section, a novel development of algorithm is presented for a demodulator used in satellite data communication. Receiver architecture and task partitioning is presented towards its end. In section 3 is presented the proposed architecture of the demodulator so that it may be implemented on an FPGA. The complete receiver was simulated in C and the BER curves are presented in the section on Results. Finally, conclusion is presented.

2. Development of the Demodulator Algorithm

Satellite receiver processes signals that bear information as well as disturbances caused by the transmitter/receiver circuits and channel impairments such as fading and additive white Gaussian noise. The receiver makes the decision on the received data using locally generated carrier oscillator and symbol clock, both of which are not referenced to the actual versions used to generate the data at the transmitter. The receiver has to estimate the offset between locally generated carrier and symbol clock to those used at the transmitter.
Carrier frequency offset recovery is the process of estimating the offset between the frequency drift/change of the local oscillator and the actual carrier frequency transmitted. Symbol timing synchronization is the process in which the receiver estimates the offset between the locally generated symbol clock at the receiver and the actual symbol clock used at the transmitter. This offset estimate is used to sample the matched filter output at the correct timing instant that maximizes the signal to noise ratio. The receiver clock, when not matched to the transmitter clock, will cause the receiver symbol decision circuitry to sample the symbols at the wrong instance, resulting in detection errors. The problem of synchronization, thus, reduces to estimating the timing and frequency offsets in the carrier, using noisy samples of the received signal, under training.

2.1 Carrier Frequency Offset Estimation

Consider a general case of a pass band signal, which has been subjected to frequency offset. It can be expressed as follows:

$$x(t) = \mathrm{Re}\left\{\tilde{x}(t)e^{j(\omega_c t+\theta(t))}\right\} = \mathrm{Re}\left\{\left[x_I(t) + jx_Q(t)\right]e^{j(\omega_c t+\theta(t))}\right\} \quad (1)$$

where $\tilde{x}(t)$ is called the complex pre-envelope of the signal, $x_I(t)$ and $x_Q(t)$ are the in-phase and quadrature-phase components, ω_c is the carrier frequency in radians/second and $\theta(t)$ models the frequency offset and/or phase jitter. If there is a constant frequency offset, then $\theta(t)$ will have a linear term, $\omega_o t$. At the receiver end, we demodulate this signal with the locally generated carrier $e^{-j(\omega_c t+\phi(t))}$, where $\phi(t)$ is the receiver's estimate of the carrier phase. If, now, we sample this demodulated signal at the symbol rate, we get

$$q_k = Re\left\{\tilde{x}(k)e^{j(\theta_k-\phi_k)}\right\} \quad (2)$$

where θ_k and ϕ_k are samples of $\theta(t)$ and $\phi(t)$ respectively. For constant value of $(\theta_k - \phi_k)$, the received constellation will be a tilted version of the transmitted constellation. If the receiver demodulates with the wrong frequency, $(\theta_k - \phi_k) = \omega_o kT$, where T is the symbol period, the received constellation rotates with an angular velocity of ω_o radian /sec. If left uncorrected, the rotating constellation will make errors every time a received symbol rotates past the boundary of a decision region. To correct this, a carrier offset estimation and a subsequent compensation is needed. Two commonly used methods of frequency-offset estimation at the receiver are data aided and non-data aided. We use data aided technique to recover the carrier offset.

2.1.1 Signal Model

The base band equivalent of the received pass band signal is given by

$$\tilde{x}(t) = \left[\sum_{n=-\infty}^{\infty}(a_n + jb_n)p(t-nT+\tau)\right]e^{j\theta(t)} + \tilde{n}_p(t) \quad (3)$$

where τ is the unknown timing offset, $\theta(t)$ models unknown frequency offset, *p(t)* is the Root Raised Cosine (RRC) pulse and $\tilde{n}_p(t)$ is the baseband equivalent of Additive White Gaussian Noise. We demodulate $\tilde{x}(t)$ with a local carrier $e^{-j\phi(t)}$, where $\phi(t)$ is the receiver's estimate of the carrier phase, resulting in

$$\tilde{h}(t) = \sum_{n=-\infty}^{\infty}(a_n + jb_n)g(t-nT+\tau)e^{j\theta'(t)} + \tilde{n}_m(t)$$

where $\theta'(t) = \theta(t) - \phi(t)$. (4)

Since the transmitter uses an RRC pulse with 40% excess bandwidth as the shaping filter, the matched filter (MF) is also an RRC pulse with 40% excess bandwidth. The output of the matched filter is given by

$$r(t) = \tilde{h}(t) \otimes p(t) = \sum_{n=-\infty}^{\infty}(a_n + jb_n)g(t-nT+\tau)e^{j\theta'(t)} + \tilde{n}(t) \quad (5)$$

where $g(t)$ is the Raised Cosine (RC) pulse.

Satellite signal impairments are mostly due to the propagation channel effects and the transmitter/receiver circuitry of both the ground stations and the satellite transponder. The timing recovery scheme that we will discuss later requires a relatively cross-talk free baseband signal. Hence we need to estimate the large frequency offset signal, compensate for it, then proceed for timing recovery and finally perform carrier phase tracking.

At the beginning of each packet, sixty-four (1, 1) symbols are used to estimate the carrier offset. In discrete time domain, the received signal for this training sequence is given by

$$r(k) = \{a_n(k) + jb_n(k)\}e^{j\theta'(k)} + \tilde{n}(k) \quad (6)$$

With $a_n(k) = b_n(k) = 1$,

$$r(k) = \{1 + j1\}e^{\theta'(k)} + \tilde{n}(k) = \sqrt{2}\,e^{j(\pi/4+\theta'(k))} + \tilde{n}(k) \quad (7)$$

$$\arg\{r(k)\} = \pi/4 + \theta'(k) + \text{some noise dependent term}$$

It may be noted that the phase of $r(k)$, i.e., $\pi/4 + \theta'(k)$ contains information about the frequency offset, but corrupted by noise. For constant phase offset, $\theta'(k) = \theta_c$ (a constant) and, for fixed frequency offset ω_o, $\theta'(k) = \omega_o k$. In both cases, $\theta'(k)$ is a linear function of k. The problem, thus, reduces to fitting a straight line through the phase trajectory and finding its slope using Recursive Least Square Algorithm (RLS) [1].

2.1.2 Unwrapping the angles

If the frequency offset $\theta'(k)$ is large, then the phase of $r(k)$ may exceed 2π. Usually, any angle greater than 2π is represented as $\theta' \bmod 2\pi$. This poses a problem during RLS implementation, since RLS works on unwrapped phase. If this sequence is input to the RLS block, then erroneous results will be generated. One approach to implement unwrapping is explained in Ref. [2]. In this method, we need to compute the phase derivative and the principal value of phase at equally

spaced frequencies. At each ω_k, one-step trapezoidal integration performs a phase estimate ω_{k-1}. If the difference between the estimate and the principal value is not within a given threshold (which can be specified), the step size is halved to get a new estimate. Another way is to use discrete time approach to solve the wrap around problem, as explained in Ref. [3]. The basic task in unwrapping is to distinguish a genuine transition of 2π radians and the spike introduced due to noise. To distinguish between the two, two absolute differences, a backward difference b(n) = $|\theta'(n)-\theta'(n-1)|$ and a forward difference f(n) = $|\theta'(n)-\theta'(n+1)|$ are maintained. While the second technique gives

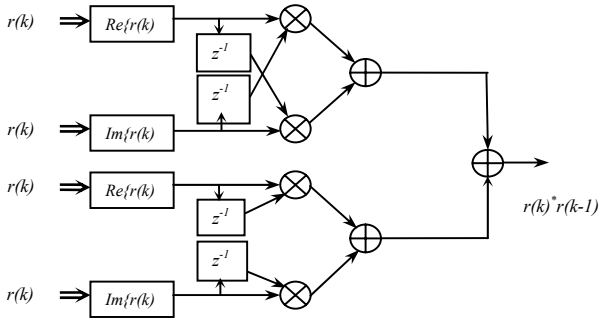


Figure 1 Structure to Calculate the Phase Angle Between Two Adjacent Samples

satisfactory results in the absence of noise and in the presence of low noise, it fails to unwrap the phase satisfactorily, when signal to noise ratio (SNR) is small. We propose a technique that works well even at lower SNR. We correlate the current sample with the conjugate of the previous one, as shown in Fig. 1. The probability of $r(k)r^*(k-1)$ exceeding 2π is very small. Therefore, accumulating these phases and then using RLS algorithm over it can solve phase unwrapping problem. It is seen that the above mentioned technique works with higher noise values as compared to the previously mentioned two unwrapping methods.

2.1.3 Timing Recovery

Once we have estimated the bulk carrier offset and compensated the incoming signal for this offset, the next step is timing recovery. The sequence is processed in the timing recovery module. We use a peak average energy criterion algorithm proposed in Ref. [3] to estimate the timing offset τ. If τ is unknown but deterministic, it has to be estimated in the receiver. We maximize the likelihood function $f(r \mid \tau)$ with respect to τ, where r is the signal space vector representation of the received noisy waveform *r(t)*, given by

$$r(t) = s(t;\tau) + n(t) \tag{8}$$

where $r(t)$ is the received signal with a unknown timing offset τ, $s(t;\tau) = \sum_n a_n g(t - nT - \tau)$ is the baseband signal, $\{a_n\}$ are binary ± 1 symbols with equal probability and $n(t)$ is the zero mean Gaussian noise. It has been shown mathematically in Ref. [3] that the value of τ that maximizes the energy is the estimate of timing offset. Finding τ that minimizes the energy is more reliable than finding τ that maximizes the energy. Thus, we will use the technique that finds the value of τ_{min}, which minimizes the energy and then correcting it to find that τ which maximizes the energy. Since we have 12 samples per symbol, the worst case of inter-symbol interference (ISI) can occur midway between two samples, resulting in maximum timing error (also called timing jitter) of

$$t_{jit} = \pm\left(\frac{1}{2}\right)\left(\frac{T}{12}\right) = \pm\frac{T}{24} \tag{9}$$

where T is the symbol period.

We maintain Modulo N sums (SUM0 to SUM11) of absolute values of the samples under a training sequence of 32 alternate symbols of (1,1) and (0,0). These sums are averaged over M (=32) symbol duration. The timing estimation algorithm is given by

$$sum(\tau) = \sum_{i=0}^{31} |x(Ni+\tau)|, \text{ where } N = 12 \text{ and } \tau = 0,1,2,...,N\text{-}1 \tag{10}$$

$$\tau_{\min} = \left\{\tau : sum(\tau) = \min_{\tau \in (0,N-1)} sum(\tau)\right\} \tag{11}$$

Once the value of τ that minimizes the energy, τ_{min}, is found, the τ corresponding to maximum energy can be calculated as follows:

If $(\tau_{min} - 6) \geq 0$, then

Timing Estimate = τmin – 6; (12)

elseif $(\tau_{min} - 6) < 0$, *then*

Timing Estimate = τmin + 6; (13)

After we have corrected the incoming signal with the estimate of the frequency offset $\hat{\theta}$, the input to the timing recovery block can be expressed as:

$$r'(k) = \sum_n (a_n + jb_n) g(kT/P - nT + \tau) e^{j(\theta' - \hat{\theta})kT/P} + n(k), \quad where\ P = 12 \tag{14}$$

Substituting $\theta' - \hat{\theta} = \varepsilon$, $r'(k)$ can be expressed as

$$r'(k) = \sum_n (a_n + jb_n) g(kT/P - nT + \tau) e^{j\varepsilon kT/P} + n(k) \quad (15)$$

where the in-phase term is given by

$$r'_I(k) = \sum_n (a_n \cos(\varepsilon kT/P) - b_n \sin(\varepsilon kT/P)) g(kT/P - nT + \tau) + n_I(k) \quad (16)$$

The quadrature-phase term is given by

$$r'_Q(k) = \sum_n (b_n \cos(\varepsilon kT/P) + a_n \sin(\varepsilon kT/P)) g(kT/P - nT + \tau) + n_Q(k) \quad (17)$$

We see that the in-phase and quadrature components have cross talk terms $b_n \sin(\varepsilon kT/P)$ and $a_n \sin(\varepsilon kT/P)$ respectively. As ε becomes large, the cross talk terms begin to affect the timing recovery estimation scheme. When $\varepsilon = 0$, there will be no cross talk and hence the estimate that we obtain is the best in the presence of noise. We get, at the output of the timing recovery scheme, a number between 0 and 11 (since we have 12 samples per symbol), which indicates the sample closest to the ideal sampling instant. Now, as we increase ε, then due to the presence of the term $e^{j\varepsilon kT/P}$, the received constellation starts rotating at a rate depending upon the magnitude of ε. The incoming symbols trace out a circle as they rotate because of uncompensated frequency offset. A least mean square algorithm based tracking loop is used to track and correct this residual frequency offset continually, which if left uncorrected, can accumulate and cause errors in the decision every time the symbols cross over the decision boundary.

2.1.4 Tracking and Symbol Detection

Frequency offset estimation and timing recovery is performed only at the start of burst and then continuous tracking and detection are done using the Least Mean Square (LMS) Algorithm [4-7].

Least Mean Square Algorithm

The LMS algorithm is a stochastic gradient algorithm. An important feature of the LMS algorithm is its computational simplicity. Consider the arrangement shown in Fig. 2. The error $e(n)$ is the difference in the filtered output $y(n)$ and the desired response $d(n)$ as shown.

$$e(n) = d(n) - y(n) \quad (18)$$

The purpose of the filter is to produce an estimate of the desired response by adaptively changing the filter coefficients. The filter coefficients are updated using the LMS algorithm with the criterion that the cost function $J = E\left[|e(n)|^2\right]$ is minimized. The desired responses in this case are the constellation points. The error signal $e(n)$ is the difference between the symbol decisions and slice input, i.e.,

$$e_k = \left[\hat{a}_k + j\hat{b}_k\right] - \left[\tilde{a}_k + j\tilde{b}_k\right] \quad (19)$$

Here, the filter has only one coefficient, which is updated using the standard LMS algorithm adaptation,

$$w_{k+1} = w_k + 2\mu e_k^* \left[a'_k + jb'_k\right] \quad (20)$$

where μ is the step size whose value needs to be chosen carefully to ensure rapid convergence of the LMS algorithm. w_{k+1} is the correction that needs to be applied to the incoming signal.

Carrier frequency offset estimation algorithm estimates the slope of the best-fit line through the unwrapped phase of the received training sequence. Timing recovery algorithm is based upon finding the instant corresponding to the minima of the absolute of the received training sequence and then correcting it to get an estimate of the ideal sampling instant.

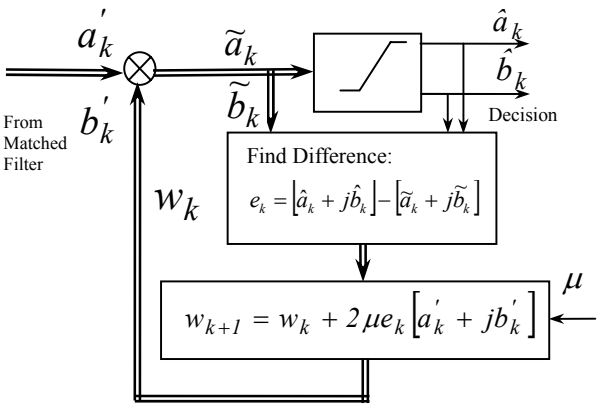


Figure 2 LMS Structure

2.1.5 Receiver Structure and Task Partitioning

In the previous section, we discussed as to how the synchronization is achieved and data is detected so as to get minimum error at the output. Carrier and Timing synchronization are the two major tasks that are performed at the receiving end. These schemes vary from receiver to receiver depending upon various factors such as tolerable bit error rate, training sequence, modulation technique in use, etc. All these techniques are employed on the signal

that is either at base band or near base band. Also, the incoming signal at the input of ADC is at the intermediate frequency (IF). Thus, IF to base band conversion must be performed before proceeding for synchronization and detection.

In the first part of this section, we will discuss IF to base band conversion scheme and the techniques involved for its simplification from the hardware implementation point of view. Then we will see an overview of the complete receiver. Finally, the entire computational load of the receiver system is partitioned between FPGA and DSP for the proposed efficient implementation.

IF to Baseband Conversion

Conversion from IF to baseband is a method to obtain a complex baseband representation of a real bandpass signal. It has a wide variety of applications in areas such as radar and sonar signal processing, digital communications and biological signal analysis. The real signal obtained from a transducer such as an antenna, a hydrophone or a biological probe, is amplified, filtered and shifted to an appropriate IF before quadrature demodulation. Once this process is done, the resulting complex signal representation contains the information present in the original signal and, its format facilitates subsequent processing such as spectral analysis or extraction of modulation information, as in the present scenario.

The input signal to a QPSK demodulator can be described by the expression:

$$\begin{aligned} x(t) &= A(t)\cos(\omega_c t + \phi(t)) \\ &= I(t)\cos(\omega_c t) - Q(t)\sin(\omega_c t) \end{aligned} \quad (21)$$

where $A(t)$ is the signal amplitude, ω_c its carrier frequency in radians/second, $\phi(t)$ its time varying unknown phase angle and $I(t)$, $Q(t)$ are the in-phase and quadrature-phase signals respectively.

The traditional analog approach to quadrature demodulation is shown in Fig. 3. The signal to be demodulated (in this case the output of the IF amplifier) is multiplied by two sinusoids with a 90 degree phase angle difference. This effectively creates quadrature versions of the signal nominally centered around zero frequency and at twice the carrier frequency. The signal component centered about $2\omega_c$ is then removed by low-pass filtering, leaving complex baseband signal [8]. If a digital representation of the in-phase and quadrature components is desired, analog-to-digital conversion (ADC) is performed. In Fig. 3, the modulating signals *cos* and *sin* are shown with amplitude of 2 in order to match the input and output power levels. In practice, however, this factor is taken care of while deciding the response of the low pass filter (LPF). The traditional analog implementation of quadrature demodulation shown in Fig. 3 suffers from many problems, especially the gain and phase mismatches between I and Q channels and the presence of DC offsets. In such an implementation, with the exception of the Analog-to-Digital Converters (ADC) used to digitize I and Q signals, all processing are carried out by analog circuits.

The first processing step after conversion of the input analog signal to a digital format is frequency shifting to baseband. It involves multiplying the input data by cosine and sine sequences at the center frequency of the input signal. This step can be quite complex, first requiring the generation of the two sinusoids, then their multiplication with the stream of input data. However, a careful selection of the sampling frequency can greatly simplify this problem [9]. If it is selected such that $f_s = 4 \times f_c$, then the two sequences are represented by $cos(\pi n/2)$ and $-sin(\pi n/2)$, which reduce to:

cos: 1, 0, -1, 0, 1, 0, -1, 0, ...
-sin: 0, -1, 0, 1, 0, -1, 0, 1, ...

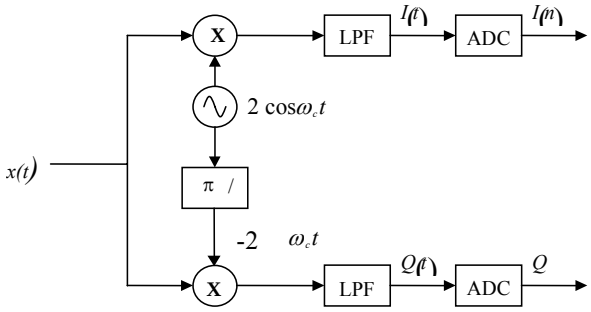


Figure 3 Analog Quadrature Demodulator Block Diagram

Multiplication by 0 and 1 are trivial. For multiplication by -1, the only processing required is sign inversion, an operation whose complexity depends on the precision of the data.

In-Phase and Quadrature Digital Filters

The second step after multiplication of the input data by quadrature sinusoids is low-pass filtering, where unwanted high-frequency mixing products are removed to obtain the I and Q channel signals. Digital filters are used for this purpose in a digital implementation of quadrature demodulation.

Phase linearity of filters used in the quadrature demodulator is an issue to be sorted out. It is essential to preserve the information contained in the original signals. A non-linear phase filter is, therefore, unsuitable for the quadrature demodulator designs considered here. While Infinite Impulse Response (IIR) filters usually have sharper transition bands than Finite Impulse Response (FIR) filters for a given filter order, they cannot have a linear phase characteristic and, thus, they are not considered further. Therefore, the quadrature demodulator

designs considered here will be restricted to linear-phase FIR filters. Fig. 4 shows a digital quadrature demodulator. The output $y(n)$ from an FIR filter with impulse response $h(n)$, filter length N, and input sequence $x(n)$ is given by the convolution of the input sequence and the filter impulse response:

$$y(n) = x(n) \otimes h(n) = h(n) \otimes x(n) = \sum_{m=0}^{N-1} h(m)x(n-m) \tag{22}$$

In the basic quadrature demodulation approach, the two low-pass filters are identical, and the filter they reproduce is called the prototype filter. The impulse response of the low pass prototype filter will hereafter be denoted by $h_{LP}(n)$. Its cutoff frequency, transition bandwidth and stop-band attenuation are selected according to the characteristics of the signal to be demodulated, especially the signal bandwidth. The passband of the filter should be at least equal to $B/2$, where B is the bandwidth of the signal prior to demodulation. From equation 22, assuming that the system sampling frequency is selected as $f_s = 4 \times f_c$, and

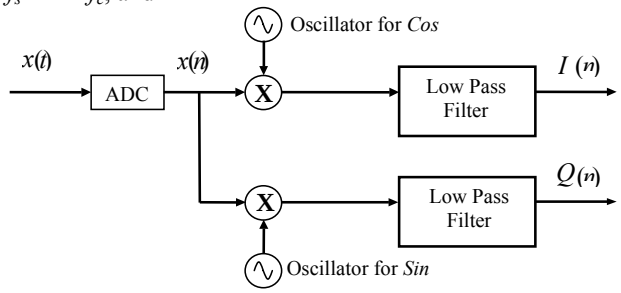


Figure 4 Digital Quadrature Demodulator

that the prototype low pass filter has an impulse response $h_{LP}(n)$, the outputs of the in-phase channel and the quadrature channel can be expressed as:

$$I(n) = h_{LP}(n) \otimes \left[x(n) \cdot cos(\frac{\pi}{2} n) \right] = \sum_{m=0}^{N-1} h_{LP}(m) \cdot x(n-m) \cdot cos(\frac{\pi}{2}(n-m)) \tag{23}$$

$$Q(n) = h_{LP}(n) \otimes \left[x(n) \cdot (-\sin(\frac{\pi}{2} n)) \right] = \sum_{m=0}^{N-1} h_{LP}(m) \cdot x(n-m) \cdot \sin(-\frac{\pi}{2}(n-m)) \tag{24}$$

Down-conversion of Frequency

If we use the criterion of sampling frequency selection for selecting the intermediate frequency and to ease the sinusoidal wave generation and filter implementation, we need an IF of $\frac{f_s}{4} = 384$ KHz for an ADC sampling frequency of 1.536 MHz. We will assume that the user specification for an IF is 455 KHz at the input of ADC. However, the intermediate frequency of 455 KHz complicates the implementation. Thus, it has to be first down-converted to 384 KHz before applying it to the input of sampling rate converter. To achieve this goal, we must generate a 71 KHz sinusoidal signal in digital domain and mix it with the 455 KHz signal so as to get signals of frequencies (455 + 71) KHz = 526 KHz and (455-71) KHz = 384 KHz at the output of the mixer. 526 KHz signal is unwanted and, therefore, it is removed by filtering. The structure of the receiver, IF to Base-band Converter, is shown in Fig. 5 and described mathematically as presented next. Signal received at the input of ADC is given by the following expression:

$$x(t) = x_i(t)cos[2\pi(455*10^3)t] - x_q(t)sin[2\pi(455*10^3)t] \tag{25}$$

Sampling at 1536 KHz, we get

$$x(n) = x_i(n)\,cos(2\pi \frac{455}{1536} n) - x_q(n)\,sin(2\pi \frac{455}{1536} n) \tag{26}$$

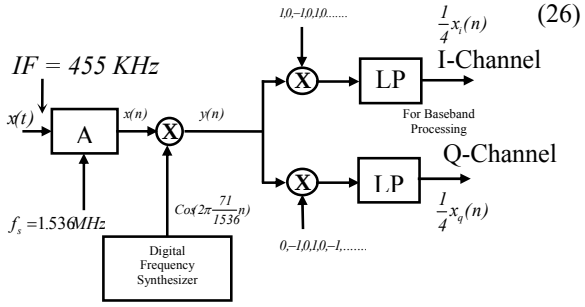


Figure 5 IF to Base-band Converter

Mixing it with $\cos\left(2\pi \frac{71}{1536} n\right)$ and, multiplying by

$cos\,(2\pi(384/1536)n)$, we get

$$\begin{aligned} y(n)*cos(2\pi\frac{384}{1536}n) = {} & \frac{1}{4}x_i(n)cos(2\pi\frac{910}{1536}n) + \frac{1}{4}x_i(2\pi\frac{142}{1536}n) \\ & + \frac{1}{4}x_i(n)cos(2\pi\frac{768}{1536}n) + \frac{1}{4}x_i(n) \\ & - \frac{1}{4}x_q(n)sin(2\pi\frac{910}{1536}n) - \frac{1}{4}x_q(n)sin(2\pi\frac{142}{1536}n) \\ & - \frac{1}{4}x_q(n)sin(2\pi\frac{768}{1536}n) \end{aligned} \tag{27}$$

Finally, passing it through low-pass filter so as to remove passband signals, we have:

$$LPF\ [\ y(n)*\ cos(\ 2\pi \frac{384}{1536} n\ )] = \frac{1}{4} x_i(n) \tag{28}$$

Similarly, the quadrature component is

$$LPF\ [y(n) * (-\sin(2\pi \frac{384}{1536} n))] = \frac{1}{4} x_q(n) \quad (29)$$

3. Proposed System Architecture for Demodulator

Combining all the parts of the system that we have discussed so far, we can present the overview of the whole system as shown in Fig. 6. The main point to note in this structure is the cascading of filters, which serves no purpose. Thus, replacing the cascade of filters by a single filter and redrawing the system structure, we get the system shown in Fig.7. In this new structure, it may be noted that the Root Raised Cosine filters serve two purposes:
1) As pulse shaping filters that have minimum ISI.
2) As low pass filters to remove unwanted out of band components.

3.1 Task Partitioning

Traditionally, DSP algorithms are implemented using general-purpose DSP chips for low throughput applications, or special-purpose DSP chips and ASICs for higher throughputs. The FPGA maintains the advantages of custom functionality like an ASIC while avoiding the high development costs and the inability to make design modifications after production. The SRAM-based FPGA is well suited for arithmetic, including multiply and accumulate (MAC) intensive DSP functions. The FPGA can also be reconfigured on-the-fly to perform one of many system level functions. When building a DSP system in an FPGA, the designer can take advantage of parallel structures and arithmetic algorithms to minimize resources and exceed the performance of single or multiple general-purpose DSP devices. Distributed Arithmetic for array multiplication in an FPGA is one way to increase data bandwidth and throughput by several orders.

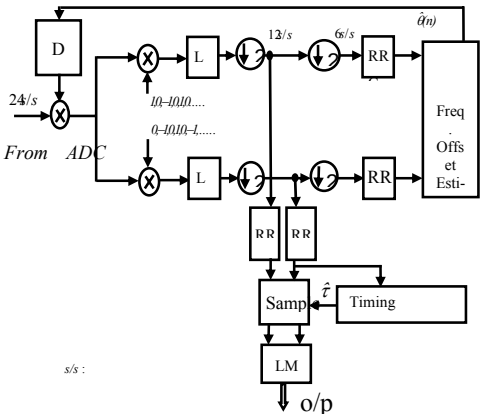


Figure 6 Overview of Architecture for Demodulator

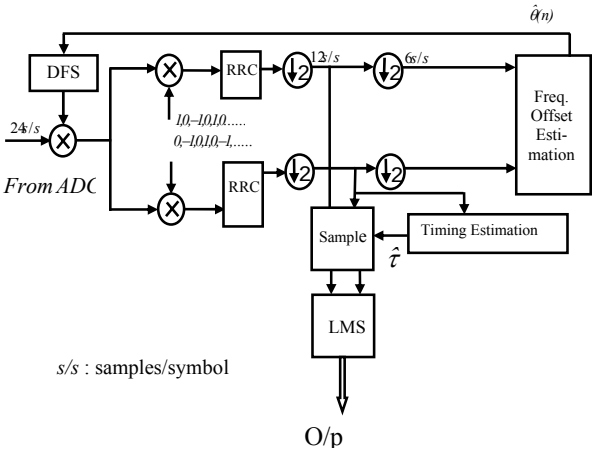


Figure 7 Modified System Structure

3.2 FPGA-Based DSP Accelerator

The engine in the DSP system is typically a specialized time multiplexed sequential computer system that performs continuous mathematical processes, attempted in real time. While the DSP processor may perform multiple instructions per clock cycle (Harvard Architecture), the overall process is performed in a three-step series of 1) Memory-Read, 2) Process and 3) Memory-Write instructions. This process is adequate for independent sequential algorithms. The processor becomes less efficient when an algorithm is dependent upon two or more of the past, present and future state conditions. This is primarily due to the feedback or parallel structure of the data flow being processed sequentially with additional wait states in a DSP.
A typical DSP algorithm may contain several feedback loops or parallel structures. The software code for DSP algorithm of this type is not efficiently implemented on general purpose DSP [10]. Typically, about 10-30 % of the DSP code utilizes 60-80 % of the processors power. Analyzing the DSP algorithm and breaking out any parallel structures or repetitive loops into multiple data-paths, one can enhance the overall performance of the algorithm. The multi-path parallel data structures can be processed either through parallel DSP devices or in a single FPGA-based DSP hardware accelerator with or without the assistance of a DSP device. The primary concepts in our approach are to load the computationally intense functions requiring multiple DSP clock cycles into the FPGA and allow the DSP processor to concentrate on the optimized single clock functions. In order to use the FPGA in a DSP design, we need to identify the parallel

data paths and the operations requiring multiple clock cycles in the DSP.

3.3 Digital Filter Design

The processing engine of most of the filter algorithms is a multiply and accumulate (MAC) functions and involving multi rate systems [11, 12]. Filter designs can vary over a wide range in the number of MACs, from one to thousands. As the number of MACs increase, the algorithm becomes increasingly more complex in terms of CPU based architecture. Hence the algorithm becomes more compute intensive for any conventional DSP. The MAC function can be implemented more efficiently with various Distributed Arithmetic Techniques than with conventional arithmetic methods. Distributed arithmetic can make extensive use of look-up tables (LUTs), which makes it ideal for implementing DSP functions in LUT-based FPGAs.

3.4 FPGA Based Filters

While designing a digital filter in an FPGA, the designer can take advantage of parallel structures and Distributed Arithmetic Algorithms to exceed the performance of multiple general-purpose DSP devices. The use of Distributed Arithmetic for array multiplication in an FPGA is one technique used to implement and increase the function's data bandwidth and throughput by several orders of magnitudes over off-the-shelf DSP solutions.
The FPGA has the capability to implement an FIR filter function using one of the several Distributed Arithmetic Techniques, depending upon the performance required. It may be noted that these techniques can be used to optimize the implementation of many other types of data processing or MAC-based algorithms. Parallel Distributed Arithmetic based techniques are used to achieve the fastest sample rates while lower rates can be sustained with a Serial Distributed Arithmetic based Techniques that use lesser resources.
The primary design concern is the performance or the sample rate of the filter. The design must work at the desired sample rate. A design which runs below the sample rate is of no value, while any additional performance, which uses more chip area, is also of no added value.

3.5 A Re-look at the Proposed Demodulator System Architecture

Observing keenly the present system structure, it can be seen that the system front end contains two high order low pass filters of size: 24 x 8 + 1 = 193 taps. Direct implementation of these structures on a DSP is computationally intensive and, therefore, time consuming. As discussed earlier, we can make use of the inherent parallelism in the FIR filter algorithm to enhance its speed by implementing it on FPGA and using Distributed Arithmetic. In addition to the FIR filter, the Digital frequency synthesizer used to generate 71 KHz in digital domain, is also one of the potential candidates for FPGA implementation owing to its inherent parallel structure. The other modules such as carrier frequency offset estimation have a built-in serial data processing scheme. Thus, they are ideally suited for DSP implementation. Based on the above facts, we can divide the whole system into two parts, one to be implemented on FPGA and the other to be implemented on DSP as shown in Fig. 8.
To summarize, the complete receiver structure has been presented. It is then partitioned into two parts, one to be implemented on FPGA and the other on DSP, based upon their best fit. The main idea is to off load the computationally intensive functions requiring multiple DSP clock cycles into the FPGA and allow the DSP to concentrate on the optimized single clock cycle functions. Work on FPGA implementation is currently under progress. Several FPGA implementations of demodulators following various other techniques have been reported in the recent years [13-15].

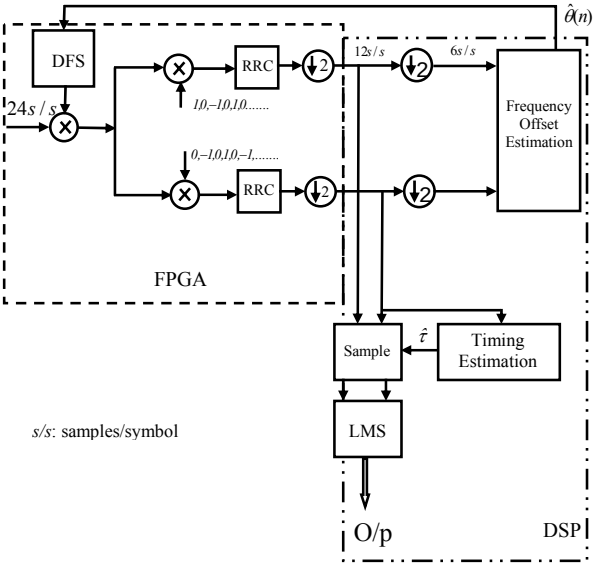


Figure 8. Task Partitioning for Implementation of the Demodulator on FPGA and DSP

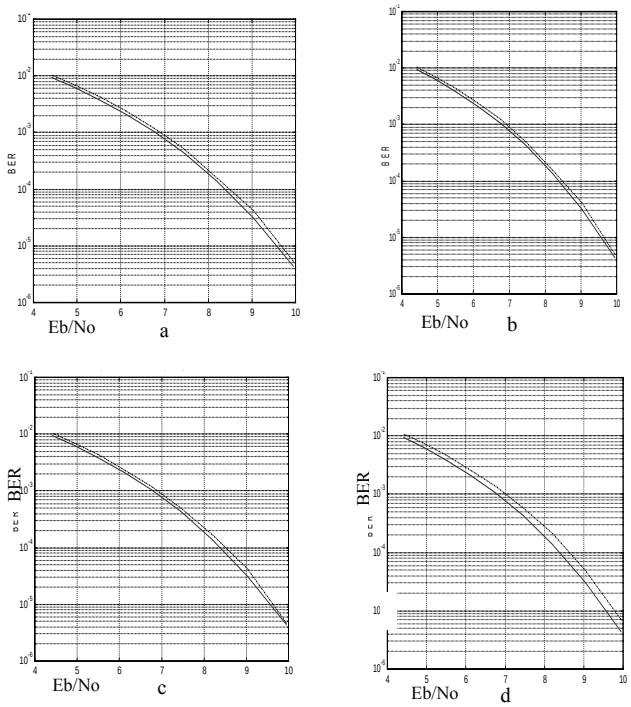
4. Results

The complete receiver presented in previous sections was simulated in C and the BER curves are presented in Fig. 9 for various values of frequency and timing offset. For simulation purposes, known frequency and timing offsets are introduced at the transmitter and estimated at the receiver end. The variance of the noise introduced after the transmitter is changed incrementally to get the range of

E_b/N_o values. The system works well up to a carrier offset of 17 KHz. It can be observed that the BER curves obtained from simulation closely follow the theoretical curves, which justifies the undertaking of FPGA-DSP implementation work proposed in this paper. Currently, the same is under development.

5. Conclusion

Complex applications such as demodulators, etc. have involved algorithms at their core, which need to be adapted or developed depending upon how we wish to implement the system. This paper dealt with the development of algorithms for the demodulator application so that it is suitable for implementation on FPGA/DSP. Before going for the architectural and hardware designs, it is of paramount importance to code



Solid line–Theoretical C Dotted line–Practical Curve

Figure 9 BER curves for

(a) Timing offset = 0, Frequency offset = 16 KHz,
(b) Timing offset = T/6, Frequency offset = 0
(c) Timing offset = T/6, Frequency offset = 16 KHz and
(d) Timing offset = T/6 + T/24, Frequency offset = 16 KHz

them in a high level language such as C or Matlab and test them to ascertain their feasibility. Accordingly, a novel algorithm of a demodulator for satellite communication was developed and presented along with its verification using C. The treatment therein was to show the power of DSP techniques to tame a complicated algorithm efficiently and thereby make it feasible for FPGA/DSP implementation. While developing algorithms for hardware implementation, we need to keep the actual hardware such as registers, counters, combination circuits, etc. in mind and, subsequently design the architecture. Only then, we will be in a position to meet stringent specifications when the algorithm is converted into an actual working product. The next in the chain of developments is the architectural design, which was also presented. It was shown that the BER results obtained from simulation using C closely follow the theoretical curves, which justifies the undertaking of time-consuming FPGA-DSP implementation work proposed in this paper. Currently, the FPGA part of the demodulator is under development using a hardware design language, Verilog.

References

[1] G. Strang: Introduction to Applied Mathematics, Wellesly Cambridge Press (1986).
[2] J. M. Tribolet: A New Unwrapping Algorithm, IEEE Trans. on Acoustic, Speech and Signal Processing, Vol. ASSP-25 No-2, pp. 170-177 (1977).
[3] Mathew P. Joseph: DSP Algorithms for On-Board Satellite Trans-multiplexer and Receiver, M.S Thesis, IIT Madras, India (2000).
[4] S. Haykin: Adaptive Filter Theory, Prentice-Hall, 2nd Edition (1991).
[5] M. E. Frerking: Digital Signal Processing in Communication Systems, Van Nostrand Reinhold, NY (1993).
[6] E. A. Lee and D. G. Messerschmitt, "Digital Communication", Second Edition, Allied Publishers Limited, 1994.
[7] J. Proakis: Digital Communications, Third Edition, International Edition, McGraw Hill (1995).
[8] J.M.P. Langlois, D. Al-Khalili, R.J. Inkol: A High Performance, Wide bandwidth, Low cost FPGA based Quadrature Demodulator, Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (1999).
[9] Henry Samueli, Bennet C. Wong: A VLSI Architecture for a High Speed All Digital Quadrature Modulator and Demodulator for Digital Radio Applications, IEEE Journal on Selected Areas in Communication , Vol. 8, No. 8 (1990).
[10] Sanjeev Dua: Algorithms and Architectural Design of an Onboard Satellite QPSK Receiver. M.S Thesis, IIT Madras, India (2003).
[11] P.P Vaidyanathan: Multirate Systems and Filter Banks. Prentice Hall Inc., Eagle-woods Cliffs, N.J (1993).
[12] Keshab K. Parhi: VLSI Digital Signal Processing Systems, John Wiley & Sons Inc., (1999).
[13] Fubing Yu: FPGA implementation of a fully digital FM demodulator, Communications Systems (ICCS), The Ninth International Conference, pp. 446-450 (2004).
[14] Charoensak, C.; Abeysekera, S. S.: FPGA implementation of efficient Kalman band-pass sigma-delta filter for application in FM demodulation, SOC Conference Proceedings, IEEE International Volume, pp. 137-138, 12-15 Sept. (2004).
[15] Zarifi, M.H.; Frounchi, J.; Asgarifar, S.; Baradaran Nia, M, FPGA implementation of a fully digital

demodulation technique for biomedical application, Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, pp. 1265-1268, 4-7 May (2008).

K. R. Nataraj obtained his ME degree from Bangalore University, India in 2000. He worked as Professor and Head of the Department during 2000-2008 and currently he is the Post Graduate Coordinator in the Department of Electronics and Communication in SJB Institute of Technology, Bangalore. Presently, he is pursuing his Ph. D. degree in Dr MGR University, Chennai. His research interests include Wireless communication, FPGA implementation, Microcontroller and Embedded systems design. He is a member of MIE, MISTE and IETE.

Dr S. Ramachandran obtained his M. Tech. and Ph. D. degrees from the Indian Institute of Technology, Kanpur and Madras respectively. He has wide academic as well as industrial experience of over 30 years, having worked as Professor in various engineering colleges as well as design engineer in industries in India and USA, designing systems and teaching/guiding students. His research interests include developing algorithms, architectures and implementations on FPGAs/ASICs for Video Processing, DSP applications, reconfigurable computing, open loop control systems, etc. He is the recipient of the Best Design Award at VLSI Design 2000, International Conference held at Calcutta, India and the Best Paper Award of the Session at WMSCI 2006, Orlando, Florida, USA. He has completed a video course on Digital VLSI System Design at the Indian Institute of Technology Madras, India for broadcast on TV by National Programme on Technology on Enhanced Learning (NPTEL) and is being broadcast in You Tube as well. He has also written a book on Digital VLSI Systems Design, published by Springer Verlag, Netherlands (www.springer.com).

Dr B. S. Nagabushana obtained his M. Tech. and Ph. D. degrees from Mysore University and Indian Institute of Science, Bangalore respectively. He has wide academic as well as software industrial experience for over 25 years. He has worked as Professor in various ngineering colleges as well as consultant to software industries like OMED (Software), Japan, BFL, CG-Smith Software Pvt. Ltd, KPIT Cummins Infosystems (Bangalore), Pvt. Limited, San Lab Technologies etc. His research interests include Wireless Communication, Neural Network, Fuzzy Logic and Embedded systems. He is the recipient of NRDC Independence Day award for the year 1992, Best Project Execution award for the year 2000 from M/s BMC Software, USA.