DIGITAL RESAMPLING AND TIMING RECOVERY IN QAM SYSTEMS

A Thesis Submitted
to the College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in the Department of Electrical and Computer Engineering
University of Saskatchewan

by

Duong Xuan Quang

Saskatoon, Saskatchewan, Canada

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, it is agreed that the Libraries of this University may make it freely available for inspection. Permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professors who supervised this thesis work or, in their absence, by the Head of the Department of Electrical and Computer Engineering or the Dean of the College of Graduate Studies and Research at the University of Saskatchewan. Any copying, publication, or use of this thesis, or parts thereof, for financial gain without the written permission of the author is strictly prohibited. Proper recognition shall be given to the author and to the University of Saskatchewan in any scholarly use which may be made of any material in this thesis.

Request for permission to copy or to make any other use of material in this thesis in whole or in part should be addressed to:

Head of the Department of Electrical and Computer Engineering
57 Campus Drive
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
S7N 5A9

Acknowledgments

First and foremost I would like to express my deepest gratitude to my supervisor, Professor Ha H. Nguyen, for his invaluable guidance and tremendous support during my studies at the University of Saskatchewan. It is an honor for me to be one of his graduate students. He has spent so much effort teaching me many aspects of technical knowledge as well as research methodology. I really appreciate his guidance and advices for my every single study and research step but still allowing me the huge freedom to work in my own schedule. This thesis would not have been possible without his excellent support.

I would like to thank Professor Eric Salt for his amazing support and advices. I enjoyed every moment taking his classes with so much practical knowledge and lots of fun. I would never forget his help with a lecture note on resampling and his invaluable advices for my thesis. I also would like to thank Dr. Eric Pelet for his tremendous guidance and support during my research. He has offered me so much of his invaluable time to explain for me critical technical concepts. He has also given me advices on all important parts of my thesis.

I also wish to thank the Department of Electrical and Computer Engineering, University of Saskatchewan and Telecommunications Research Laboratories (TRLabs) for the excellent resources they offered me during my studies and research.

I would not forget to gratefully acknowledge the financial support from NSERC and the University of Saskatchewan graduate scholarship for my studies.

Abstract

Digital resampling is a process that converts a digital signal from one sampling rate to another. This process is performed by means of interpolating between the input samples to produce output samples at an output sampling rate. The digital interpolation process is accomplished with an interpolation filter.

The problem of resampling digital signals at an output sampling rate that is incommensurate with the input sampling rate is the first topic of this thesis. This problem is often encountered in practice, for example in multiplexing video signals from different sources for the purpose of distribution. There are basically two approaches to resample the signals. Both approaches are thoroughly described and practical circuits for hardware implementation are provided. A comparison of the two circuits shows that one circuit requires a division to compute the new sampling times. This time scaling operation adds complexity to the implementation with no performance advantage over the other circuit, and makes the "division free" circuit the preferred one for resampling.

The second topic of this thesis is performance analysis of interpolation filters for Quadrature Amplitude Modulation (QAM) signals in the context of timing recovery. The performance criterion of interest is Modulation Error Ratio (MER), which is considered to be a very useful indicator of the quality of modulated signals in QAM systems. The methodology of digital resampling in hardware is employed to describe timing recovery circuits and propose an approach to evaluate the performance of interpolation filters. A MER performance analysis circuit is then devised. The circuit is simulated with MATLAB/Simulink as well as implemented in Field Programmable Gate Array (FPGA). Excellent agreement between results obtained from simulation and hardware implementation proves the validity of the methodology and practical application of the research works.

Table of Contents

List of Abbreviations

M-QAM	M-ary Quadrature Amplitude Modulation
ADC	Analog to Digital Converter
AM	Amplitude Modulation
ASIC	Application Specific Integrated Circuit
ASK	Amplitude-Shift Keying
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase-Shift Keying
CATV	Cable Television
DAC	Digital to Analog Converter
DOCSIS	Data Over Cable Service Interface Specification
DVB	Digital Video Broadcasting
FIFO	First In, First Out
FIR	Finite Impulse Response
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
FSK	Frequency-Shift Keying
GTED	Gardner Timing Error Detector
HDL	Hardware Description Language

IF	Intermediate Frequency
IIR	Infinite Impulse Response
ISI	Inter-Symbol Interference
LSB	Least Significant Bit
MER	Modulation Error Ratio
MSB	Most Significant Bit
PAM	Pulse Amplitude Modulation
PLL	Phase Locked Loop
PSD	Power Spectral Density
PSK	Phase-Shift Keying
QAM	Quadrature Amplitude Modulation
RC	Raised Cosine
RF	Radio Frequency
SNR	Signal to Noise Ratio
SRRC	Square Root Raised Cosine
TED	Timing Error Detector
VLSI	Very Large Scale Integration
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access

List of Tables

List of Figures

1. Introduction

A digital communication system is built on a transmitter and a receiver. The digital information is conveyed from the transmitter to the receiver over some transmission medium. For example, the medium can be coaxial cable or a wireless channel. The transmitter generates continuous-time signals that carry the information by modulating a sinusoidal carrier. There are several ways to modulate the carrier. The modulation scheme widely used over cable is Quadrature Amplitude Modulation, as it increases the spectral efficiency of transmission by utilizing both amplitude and phase components of the carrier. Many popular communication standards adopted QAM as their modulation scheme. Examples include Cable Television (CATV), Digital Video Broadcasting (DVB), Data Over Cable Service Interface Specification (DOCSIS), Wireless Fidelity (Wi-Fi), Worldwide Interoperability for Microwave Access (WiMAX).

Most of the research works on communication systems target the Application Specific Integrated Circuit (ASIC). However, there is an alternative technology today, which is the Field Programmable Gate Array (FPGA). FPGA has many advantages over ASIC. Most importantly, FPGA is preferred over ASIC in terms of designing flexibility as well as the development cost and time. In fact, FPGA has been widely used as the hardware platform for communication systems.

In a QAM system, there is a critical need to change the sampling rate of signals digitally. This process is referred to as digital resampling. The most common application of digital resampling is to recover the transmitted sampling time in a QAM receiver (timing recovery). Another example of application is to convert sampling

rate asynchronously (sampling rate conversion) in the multiplexing of input signals at different sampling rates to an output signal at a common output sampling rate. Digital resampling in QAM systems, which can be efficiently implemented in FPGA devices, is the research subject of this thesis.

1.1 Motivation

Fundamentals and implementation of digital interpolation in timing recovery are discussed in [1] and [2] where it is shown that the hardware structure of polynomial interpolation filters can be effectively implemented by the Farrow structure [3]. In [4], a practical timing recovery circuit is described. However, those research works only focused on digital resampling in the context of timing recovery. In [5], a digital synchronizer circuit designed specifically for a digital modulator introduced in [6] is described. In this thesis, operation and circuits for the general case of digital resampling are described based on the concept of time base generation and synchronization [7] and the theory of digital interpolation [1]. The objective is to thoroughly describe and verify practical resampling circuits which are suitable for FPGA implementation. The circuits can be used in timing recovery as well as general asynchronous sampling rate conversion applications.

The heart of a digital resampling circuit is an interpolation filter. An interpolation filter introduces errors, which can be evaluated with different methods. A traditional method is to analyze the performance of each filter by considering the frequency response of the interpolator [2]. The main drawback of such a method is that the performance is evaluated as functions of fractional interval, i.e., a distance from an output sample to a referenced input sample. But in practical applications, a fractional interval is not fixed, but varies. In [8], the mean square error between the ideally interpolated points and those obtained by the circuit under investigation is computed. However, the method does not investigate interpolation error for a certain type of signal/application. Another research work evaluates the effect of interpolation on the Bit Error Rate (BER) performance [9]. The drawback is that BER does not fully

show the quality of received signals. In this research, with the understanding that interpolation errors tie to a certain type of signal and application, the performance of polynomial interpolation is specifically evaluated for QAM signal in timing recovery application. The performance criterion of interest is Modulation Error Ratio (MER), which is believed to be a very useful indicator of the quality of modulated signals in QAM systems. In particular, the objective is to propose a model that can accurately measure the MER performance of an interpolation filter for timing recovery.

1.2 Thesis Outline

In Chapter 2 of this thesis, the background of a QAM system is reviewed. Some applications of digital resampling are described. Chapter 2 also discusses key operations of timing recovery in QAM receivers.

Chapter 3 reviews the theory of digital resampling. The methodology of time base generation and synchronization to perform resampling in hardware are described. From the described methodology, asynchronous sampling rate conversion circuits (resamplers) are discussed in detail. The circuits are simulated in MATLAB/Simulink software for verification.

Another application of resampling is timing recovery in a QAM receiver and it is presented in Chapter 4. Timing recovery circuits and hardware operations are discussed in this chapter. Chapter 4 also proposes a model to evaluate MER performance of interpolation filters. Based on the model, a performance analysis circuit is devised and simulated in MATLAB/Simulink software for different interpolation filters.

Hardware structures of the MER performance analysis circuit are described in Chapter 5. The circuit is also designed using Verilog Hardware Description Language and implemented in a FPGA device. The results obtained from FPGA implementation are also discussed and compared with the results obtained from software simulation.

Chapter 6 gives some remarks and conclusions. Appendix A reviews the mathematical expressions and discusses frequency responses of the interpolations filters evaluated in this thesis.

2. Background

2.1 Digital Communication Systems

Information is always a critical part of people's lives. Communications, the way information is exchanged, can be made instantaneously by means of electronic devices. A block diagram of an electronic communication system is shown in Figure 2.1. It consists of a transmitter and a receiver. The digital information is conveyed from the transmitter to the receiver over a medium (channel). The "synchronization" block is needed in digital systems [10].

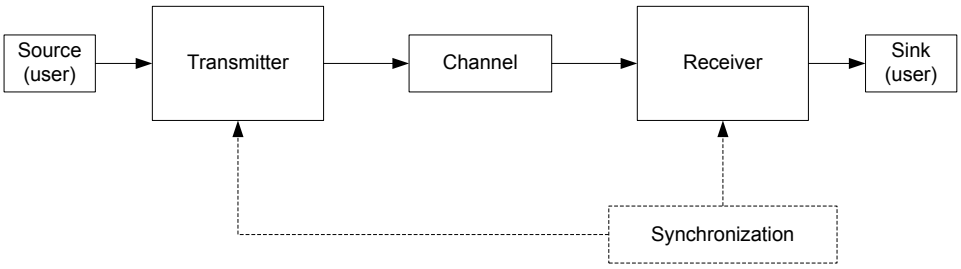

Figure 2.1 General block diagram of a communication system.

The exchange of information between a transmitter and a receiver over a medium is accomplished via modulation and demodulation processes. Modulation is a process that converts information into a signal that is suitable for transmission over the channel. For radio transmission, the modulation process uses a sinusoid signal (carrier) at radio frequency ω_0 to carry information as expressed in Equation (2.1) [11]:

$$A(t)\cos(\omega_0 t + \theta(t)) = I(t)\cos(\omega_0 t) - Q(t)\sin(\omega_0 t). \tag{2.1}$$

Here, $A(t)$ and/or $\theta(t)$ are the information to be transmitted and represented as baseband signals. In particular, $I(t) = A(t)\cos(\theta(t))$ is the in-phase component and

$Q(t) = A(t)\sin(\theta(t))$ is the quadrature component of the signal. As can be seen, $I(t)$ and $Q(t)$ are expressed in cosine and sine functions and they are 90° out of phase with each other. Information can be either in analog form or digital form. Also, modulation can be either analog type or digital type. In analog modulation, $A(t)$ and/or $\theta(t)$ are analog signals taken from a continuum of possible waveforms. Common examples are commercial AM (Amplitude Modulation) and FM (Frequency Modulation) radio systems. In digital modulation, $A(t)$ and/or $\theta(t)$ are digital signals taken from a finite set of possible waveforms. Some examples of systems using digital modulation are cellular telephone and digital video broadcast systems.

The modulated signal is transmitted through a channel. Common physical channel mediums are wireline, freespace and optical fiber. Different types of medium introduce different distortions on the modulated signal. The most significant distortion of a wireline channel is the addition of Gaussian thermal noise, while a signal transmitting through a wireless channel suffers from both fading and noise. Also, each type of modulated signal suffers differently from the channel distortions.

The receiver demodulates the received signal to detect the information that was transmitted. The receiver typically implements several techniques to overcome channel distortions. The difference between the information received and the information transmitted measures the performance of a communication system.

The signal processing and hardware realization of a communication system can also be either analog or digital type. For not to be confused with modulation and signal types, the popular alternative terms to describe the signal processing/hardware implementation method are continuous-time and discrete-time. Figure 2.2 shows possible combinations of modulation types and signal processing/hardware realization methods in a communication system.

Discrete-time signal processing/hardware realization can be applied to analog signals/modulation by *sampling* and *quantization* processes. Furthermore, the combination of continuous-time and discrete-time processing is widely used in communi-

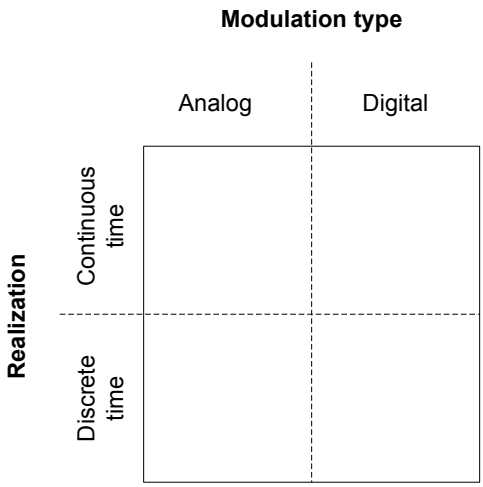

Figure 2.2 Different modulation types and hardware/signal processing methods.

cation systems. The trend is to use as much as possible the discrete-time processing/hardware realization in the form of VLSI (Very Large Scale Integration) circuits and programmable circuits.

Digital communication has been replacing analog communication for many reasons. Digital communication offers new services, security, flexibility and power saving. Moreover, digital hardware has become more reliable, powerful and affordable [11]. However, digital communication has to face the problems of synchronization and bandwidth efficiency. This thesis work focuses on digital communication systems realized with discrete-time signal processing/hardware.

The block diagram of a typical digital communication system is shown in Figure 2.3. The source is a bit stream. If the source is an analog signal, then an Analog-to-Digital Converter is needed in the transmitter and a Digital-to-Analog Converter is needed in the receiver.

In the transmitter, the source encoder removes redundancy existing in the source information so that it can be efficiently represented in a bit stream. The encrypter adds security to the source bit stream, hence protecting the information from unintended receiver. The channel encoder adds redundancy for the purpose of error detection and correction, which is performed by the channel decoder at the receiver

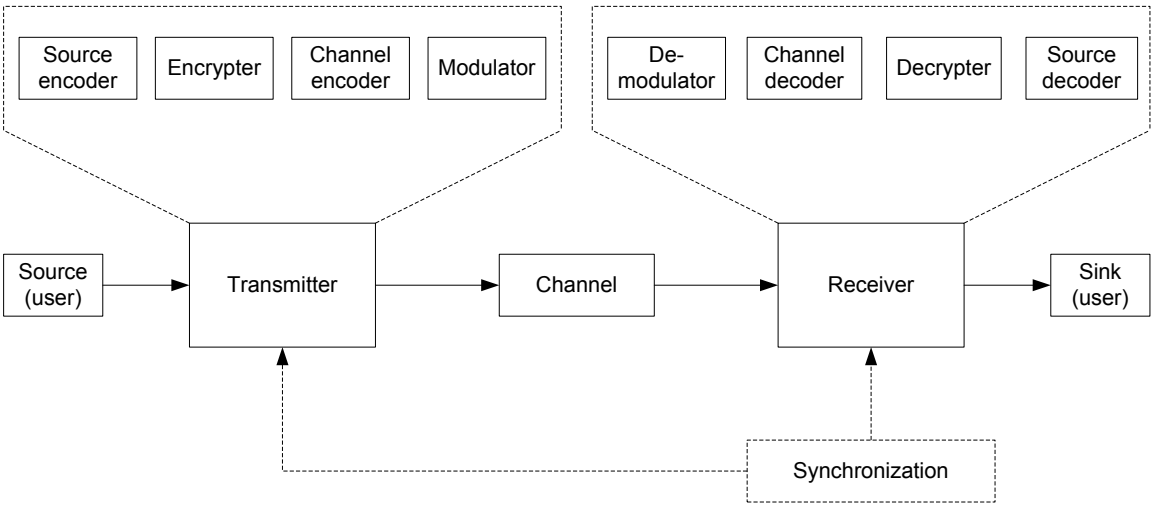

Figure 2.3 General block diagram of a digital communication system.

side. The modulator modulates the signal to generate a waveform which is suitable for transmission over the channel. The timing synchronization block synchronizes (aligns) the sampling times at the receiver and the sampling times at the transmitter.

2.1.1 Digital Modulation

In digital modulation, the signal to be transmitted is taken from a finite set of possible waveforms, which is defined as a *signal set*. Different digital modulation methods use different signal sets. In Phase-Shift Keying (PSK), the signal set consists of a finite set of waveforms that are different in phases. In Frequency-Shift Keying (FSK), the finite signal set consists of waveforms at different frequencies. The Amplitude-Shift Keying (ASK) uses a finite number of amplitudes to differentiate different signals in the set. In M-ary Quadrature Amplitude Modulation (M-QAM), the signal set consists of a finite number of waveforms that are different in amplitude and/or phase.

In general, a M-ary digital modulation scheme uses M waveforms to transmit information over the channel. The signal set of M waveforms is defined as follows:

$$\mathcal{S} = \{s_0(t), s_1(t), \ldots, s_{M-1}(t)\}\,. \tag{2.2}$$

Each waveform is a linear combination of basis signals. The set of basis signals, $\mathcal{B}$,

consists of $K \leq M$ orthogonal signals defined over the interval $T_1 \leq t \leq T_2$:

$$\mathcal{B} = \{\phi_0(t), \phi_1(t), \ldots, \phi_{K-1}(t)\}. \tag{2.3}$$

The K orthonormal basis functions satisfy the orthogonal condition for every $0 \leq i, j \leq K-1$. That is

$$\int_{T_1}^{T_2} \phi_i(t)\phi_j(t)dt = \delta(i-j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}. \tag{2.4}$$

The set of all waveforms that are linear combinations of basis signals in $\mathcal{B}$ is defined as a *signal space*, or the span of $\mathcal{B}$, Span$\{\mathcal{B}\}$. The equivalent mathematical expression is:

$$s(t) \in \text{Span}\{\mathcal{B}\} \Leftrightarrow s(t) = \sum_{k=0}^{K-1} a_k \phi_k(t), \tag{2.5}$$

where a_k, $k = 0, 1, \ldots, K$ are real constants. For *linear modulation*, $\mathcal{S}$ is a selected subset of Span$\{\mathcal{B}\}$. Each of M waveforms in $\mathcal{S}$ is a linear combination of K basis signals in $\mathcal{B}$ as follows:

$$\begin{cases} s_0(t) = a_{0,0}\phi_0(t) + a_{0,1}\phi_1(t) + \ldots + a_{0,K-1}\phi_{K-1}(t) \\ s_1(t) = a_{1,0}\phi_0(t) + a_{1,1}\phi_1(t) + \ldots + a_{1,K-1}\phi_{K-1}(t) \\ \vdots \\ s_{M-1}(t) = a_{M-1,0}\phi_0(t) + a_{M-1,1}\phi_1(t) + \ldots + a_{M-1,K-1}\phi_{K-1}(t) \end{cases} \tag{2.6}$$

Equation (2.6) is a *synthesis equation*, which defines how to construct $\mathcal{S}$ from $\mathcal{B}$. Each waveform in $\mathcal{S}$ can be equivalently represented by a $K-$ tuple of weighting coefficients, $a_{i,j}$, as follows:

$$\begin{cases} \mathbf{s}_0 = (a_{0,0}\ a_{0,1}\ \ldots\ a_{0,K-1}) \\ \mathbf{s}_1 = (a_{1,0}\ a_{1,1}\ \ldots\ a_{1,K-1}) \\ \vdots \\ \mathbf{s}_{M-1} = (a_{M-1,0}\ a_{M-1,1}\ \ldots\ a_{M-1,K-1}) \end{cases} \tag{2.7}$$

Each K-tuple $\mathbf{s}_m$ is a point in a K-dimensional space, Span$\{\mathcal{B}\}$. The set of M points $\mathbf{s}_m$ in the signal space is called the *constellation*. Each point in the constellation

represents a *symbol* which has a symbol duration $T = T_2 - T_1$. Each symbol can be mapped from a $(\log_2 M)$-bit pattern. The implication of symbol mapping is that, one symbol now can be used to transmit $\log_2 M$ bits over the channel. Linear modulation has been widely applied in communication systems since they offer good bit error rate performance and bandwidth efficiency. The modulation scheme of interest in this thesis is M-QAM, which has become a dominant modulation scheme for high-speed applications. M-QAM uses a 2-dimensional ($K = 2$) basis signal set as follows:

$$\mathcal{B} = \{\phi_I(t), \phi_Q(t)\}, \tag{2.8}$$

where

$$\begin{cases} \phi_I(t) = \sqrt{2}p(t)\cos(\omega_0 t) \\ \phi_Q(t) = -\sqrt{2}p(t)\sin(\omega_0 t) \end{cases} \tag{2.9}$$

The signal $p(t)$ is an unit-energy pulse. One common example is $p(t) = \sqrt{1/T_s}$ on the interval $T_1 \leq t \leq T_2$. The carrier frequency ω_0 is usually chosen so that there is an integer number of sinusoid cycles in the interval $T_1 \leq t \leq T_2$. The actually transmitted power can be set by scaling the basis signals with an amplitude V. The orthogonality of $\phi_I(t)$ and $\phi_Q(t)$ is easily verified. Figure 2.4 shows the geometric interpretation of 2-dimensional signal space, i.e., Span$\{\phi_I(t), \phi_Q(t)\}$.

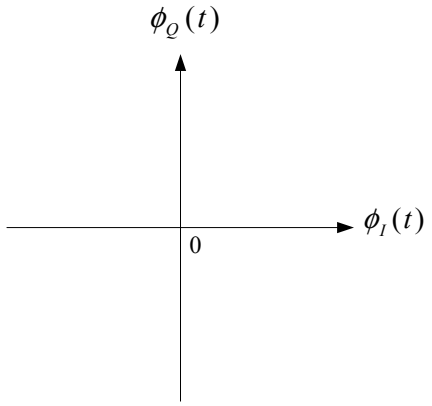

Figure 2.4 The 2-dimensional signal space.

The signal set $\mathcal{S}$ has M waveforms. Each of the waveforms is defined by a $2-$tuple

of weighting coefficients as follows:

$$\begin{cases} \mathbf{s}_0 = (a_{I,0}\ a_{Q,0}) \\ \mathbf{s}_1 = (a_{I,1}\ a_{Q,1}) \\ \vdots \\ \mathbf{s}_{M-1} = (a_{I,M-1}\ a_{Q,M-1}) \end{cases} \tag{2.10}$$

The constellation $\mathcal{C} = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{M-1}\}$ in a 2-dimensional signal space can have different shapes. Figure 2.5 shows square constellations for $M = 4$ and $M = 16$. Also shown are Gray code bit mappings of QAM symbols. In Gray code, $(\log_2 M)$-bit patterns mapped from one symbol and its closest symbols differ in only 1 bit. Note that the amplitude levels of both the inphase and quadrature signal components are equally spaced.

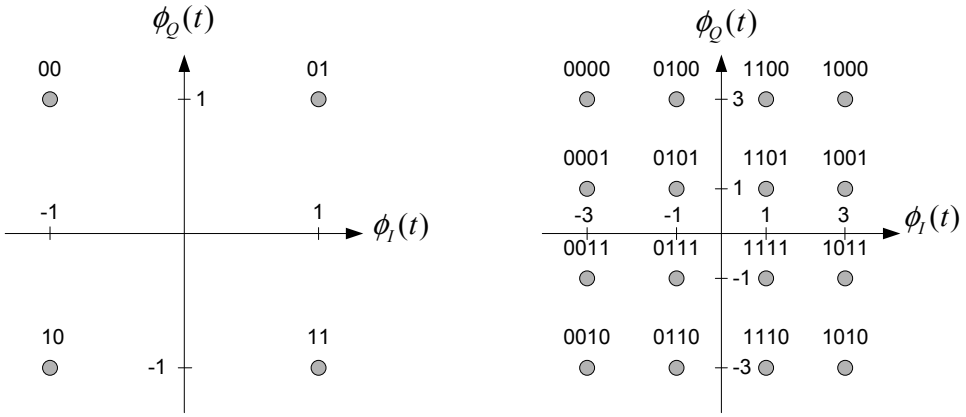

Figure 2.5 Square M-QAM constellations and Gray symbol mappings for $M = 4$ and $M = 16$.

The hardware description for a QAM modulator will be discussed in Section 2.2. As we will see, the orthogonality property of $\mathcal{B}$ simplifies the hardware structures of the modulator and demodulator.

2.1.2 Digital Demodulation

At the receiver side, suppose that the received signal is

$$r(t) = s(t) + w(t),\ T_1 \leq t \leq T_2 \tag{2.11}$$

where $w(t)$ represents additive white Gaussian noise. There are two major steps to be performed in the receiver to decide the signal in $\mathcal{S}$ that was transmitted. The first step is to convert $r(t)$ to a set of numbers that can retain all the information in the transmitted signal (also known as a set of sufficient statistics) [10]. This process is referred to as the *projection* of $r(t)$ onto the K-dimensional signal space. The projection process is to obtain a K-tuple $\mathbf{r} = (r_0\ r_1\ \ldots\ r_{K-1})$. From the orthogonality of the basis signals in $\mathcal{B}$, $r(t)$ can be projected independently onto each basis signal. The mathematical operation is to compute each coefficient of $\mathbf{r}$ as follows [11]:

$$r_k = \int_{T_1}^{T_2} r(t)\phi_k(t)dt,\ \ k = 0, 1, ..., K-1. \tag{2.12}$$

Equation (2.12) is called the *analysis equation*, which defines the projection of $r(t)$ onto the signal space. Figure 2.6 demonstrates the geometric interpretation of the projection of $r(t)$ onto a 2-dimensional signal space to obtain a 2-tuple $\mathbf{r} = (r_I\ r_Q)$ for M-QAM modulation.

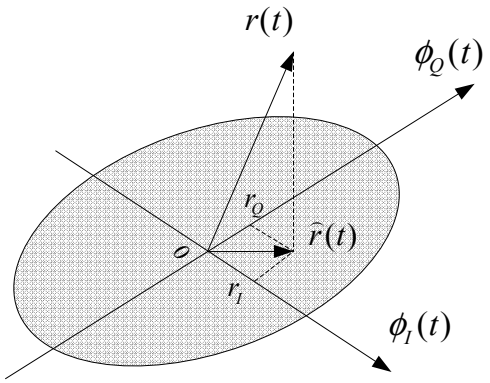

Figure 2.6 A projection of the received signal to a 2-dimensional signal space.

The second step is to approximate the K-tuple $\mathbf{r}$ to a point in the constellation. The natural approximation criterion is to minimize the *bit error probability*. The minimization problem is discussed in [10] where the signal space is divided into *decision regions* to decide which signal in the signal set $\mathcal{S}$ was transmitted given the K-tuple $\mathbf{r}$. If all the symbols are equally likely (i.e., the probability of transmitting all symbols is the same), the minimum bit error probability receiver becomes a *minimum-distance receiver*. The symbol in the constellation that is closest to $\mathbf{r}$ in terms of the Euclidean

distance is then decided as the transmitted symbol. In this way, the signal space is divided into decision regions based on the geometry of the constellation.

Figure 2.7 shows an example of square 4-QAM constellation where the 2-dimensional signal space is divided into 4 shaded quadrants, each quadrant is one decision region for one symbol. If the projection of a received symbol falls within a specific quadrant, the corresponding constellation symbol of the quadrant is decided as the transmitted symbol. In this example, $r_I \geq 0$ and $r_Q \geq 0$ so $r(t)$ is decided as $s_1(t)$. In terms of bit mapping, symbol $s_1(t)$ carries the 2-bit pattern (01).

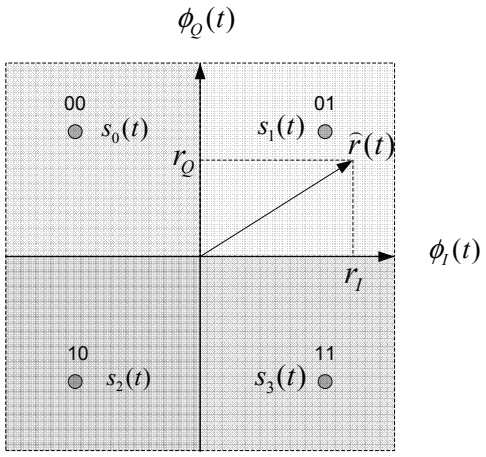

Figure 2.7 Decision regions in a 2-dimensional signal space for square 4-QAM constellation.

There is always a chance of making errors in the receiver. Given a certain model of the channel, performance of the distance minimization solution is evaluated in terms of error probability as a function of SNR (Signal to Noise Ratio) and the probability of transmitting each signal in the signal set $\mathcal{S}$ [10]. Performance of a digital communication system can be measured by using different criteria. The most common performance measures are the Bit Error Rate (BER) and Modulation Error Ratio (MER) . These two criteria are discussed in the following section.

2.1.3 Performance Criteria

Bit Error Rate (BER)

In a digital communication system, BER is the average ratio of the number of bits that are wrongly detected in the receiver to the total number of transmitted bits.

$$\text{BER} = \text{Number of bits wrongly detected/Number of bits sent.} \tag{2.13}$$

Thermal noise is the main factor that affects BER performance of information transmission over an Additive White Gaussian Noise (AWGN) channel (see Equation (2.1) for the channel model). In the presence of noise, BER is often expressed as a function of the normalized carrier-to-noise ratio E_b/N_0 where E_b is the energy per information bit and N_0 is the one-sided noise power spectral density. Figure 2.8 shows the BER curves for square 4-QAM and 16-QAM constellations with Gray code over an AWGN channel. The BERs were computed by simulating the minimum distance receiver in MATLAB software. The plots show that, given a certain noise power spectral density of the channel, the more power transmitted is, the better the BER performance becomes. Also, with the same average power transmitted, 4-QAM offers better BER performance than 16-QAM. The better BER performance of 4-QAM in comparison with 16-QAM comes with the cost of a lower bit rate.

When expressed as a function of the normalized carrier-to-noise ratio, the BER curve provides the system designer with an useful information about the performance of the system and the required energy per bit to transmit given a certain noise power spectral density of the channel.

Modulation Error Ratio (MER)

MER is used to measure how far projected received signals are from the nominal values. Figure 2.9 shows the geometric interpretation of the error between $\hat{r}(t)$ and its ideal signal $s_1(t)$. Technically, MER is affected by almost every distortion/imperfection in a communication system. MER is a good indicator of the condition of the received signals and extremely helpful for troubleshooting. It can ex-

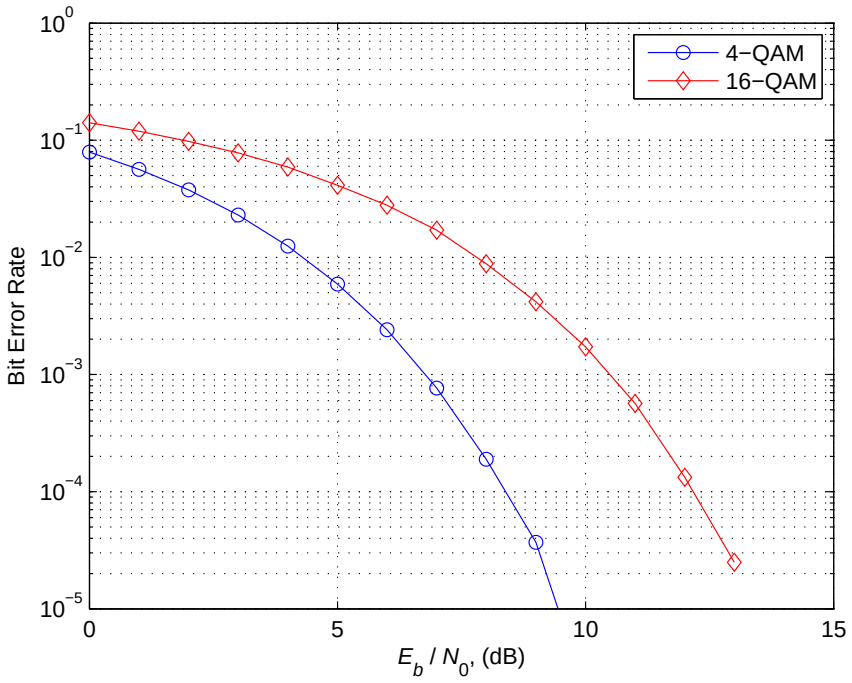

Figure 2.8 BER curves for square 4-QAM and 16-QAM.

tract useful information from noise that affects QAM signals. Figure 2.9 also demonstrates the effects of amplitude noise and phase noise in the received constellation. While BER can be zero, MER can never be infinite. For example, all the received signals which fall within the decision boundary of respective nominal symbols have different MERs but introduce no errors in terms of BER. The more errors caused by the system, the fuzzier the received constellation becomes. And the fuzzier the received constellation becomes, the lower the MER is.

Mathematically, MER is the ratio of average symbol power to average error power. Expressed in decibel:

$$\text{MER} = 10 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{error}}} \right). \tag{2.14}$$

In a M-QAM system, MER over N symbols is calculated as follows:

$$\text{MER} = 10 \log_{10} \left(\frac{\sum_{i=1}^{N} (\tilde{a}_I[i]^2 + \tilde{a}_Q[i]^2)}{\sum_{i=1}^{N} [(a_I[i] - \tilde{a}_I[i])^2 + (a_Q[i] - \tilde{a}_Q[i])^2]} \right), \tag{2.15}$$

where $a_I[i]$ and $a_Q[i]$ are the I and Q components, respectively, of the i^{th} symbol

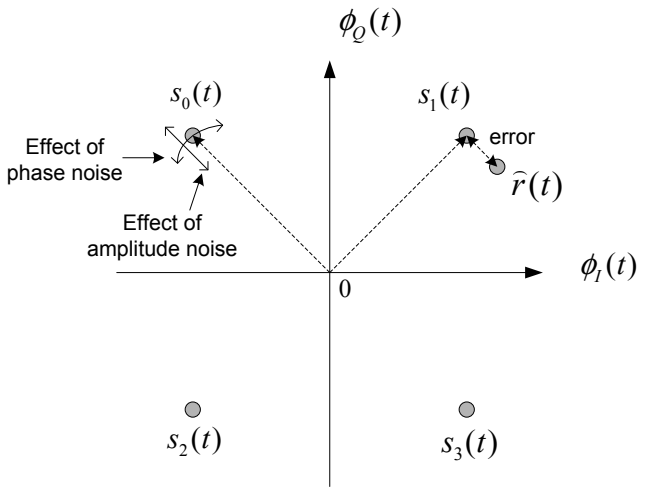

Figure 2.9 Errors measured by MER.

received, and $\tilde{a}_I[i]$ and $\tilde{a}_Q[i]$ are the ideal I and Q components of the i^{th} symbol in the constellation $\mathcal{S}$.

2.2 Discrete-Time QAM Communication Systems

This section discusses the discrete-time realization of a QAM communication system. The overall block diagram of such a system is depicted in Figure 2.10, which does not show the source encoder/decoder, encrypter/decrypter, channel encoder/decoder blocks and focuses on the modulation/demodulation and timing recovery parts.

QAM Transmitter

Let first consider the QAM transmitter. The information bits are converted into QAM symbols by *symbol mapping* (Gray mapping). A QAM symbol is a 2-tuple and is represented as $(a_I[i], a_Q[i])$, which is one of the M 2-tuples in the constellation $\mathcal{C}$ (see Equation (2.10)). Here $a_I[i]$ represents the amplitude level of inphase carrier while $a_Q[i]$ represents the amplitude level of quadrature carrier. For example, in 4-QAM, $a_I[i]$ and $a_Q[i]$ can take on the values $+1$ or -1, so $(a_I[i], a_Q[i])$ can be one of the following four possible tuples: $(+1,+1), (+1,-1), (-1,+1), (-1,-1)$. For 2^λ-QAM, λ bits are mapped to a symbol at a time. This mapping can be implemented by means of a lookup table. The sequences $\{a_I[i]\}$, $\{a_Q[i]\}$ are up converted by U

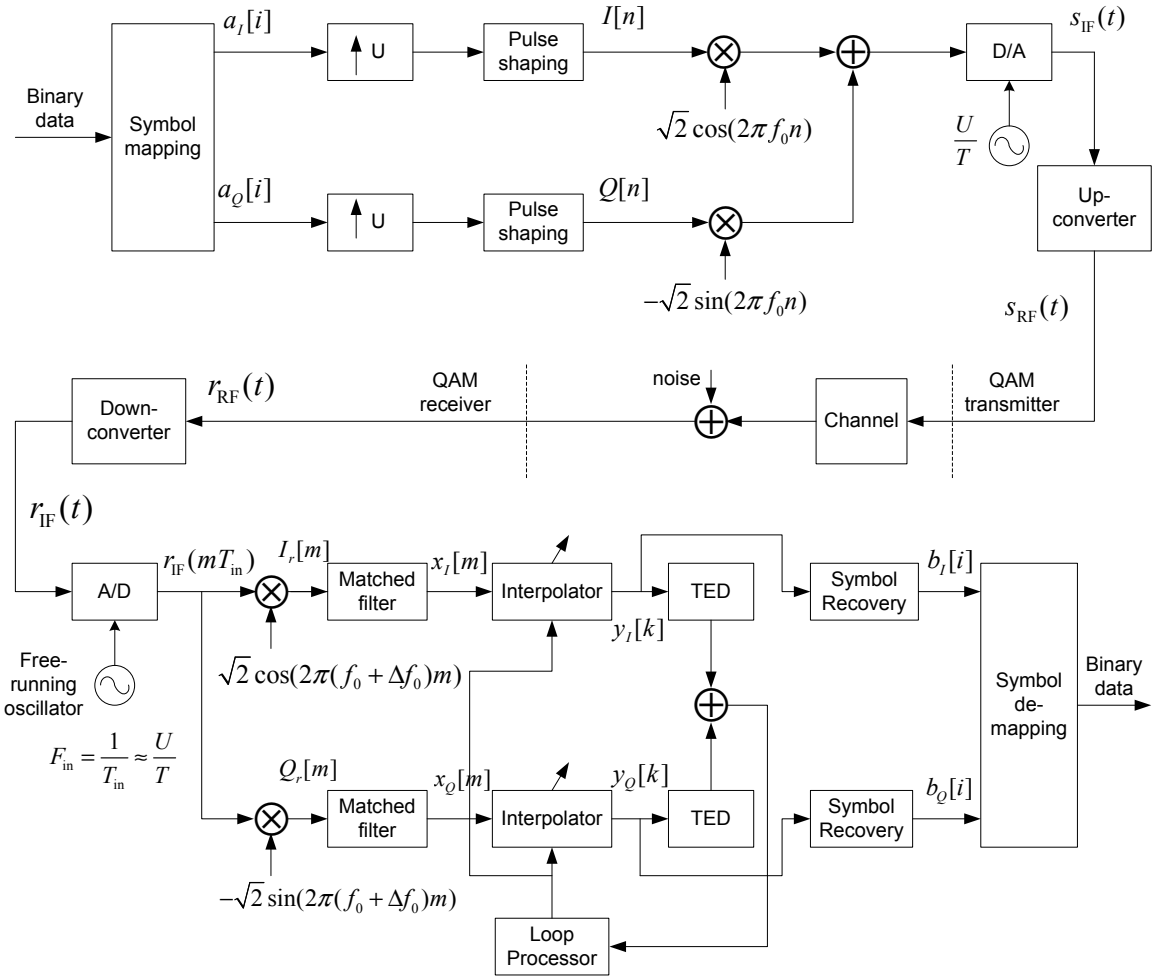

Figure 2.10 QAM system block diagram.

with zero stuffing and fed to the pulse-shaping filter. Each $(a_I[i], a_Q[i])$ is represented with 2 pulses whose peaks are $a_I[i]$ and $a_Q[i]$.

The interval of time between pulses when expressed in seconds is referred to as the *symbol interval*, which is T seconds and equal to U samples long. The pulse-shaping filter's outputs, $I[n]$ and $Q[n]$, are referred to as the inphase and quadrature signals. Those signals are modulated with digital quadrature carriers of frequency f_0. The power scaling V for actually transmission of the modulated signals is not shown here. The digital carrier frequency f_0 has an unit of cycles/sample as it is equal to the analog frequency (cycles/second) normalized by sampling frequency (samples/second). The D/A clock U/T has an unit of samples/second, and then the analog signal $s_{\mathrm{IF}}(t)$ is at *intermediate frequency (IF)* of $f_0 \times U/T$ (Hz).

Cable Channel

The medium of interest in this thesis is *coaxial cable*. The intermediate frequency signal $s_{\text{IF}}(t)$ is upconverted to a frequency band that is suitable for the actual transmission over the cable channel[1]. The transmitted signal suffers from noise and distortions. In cable transmission signal distortions are relatively small and the noise is often modeled as AWGN. This means that the channel can be modeled as a delay.

QAM Receiver

The front-end of a QAM receiver is also described in Figure 2.10. The RF (Radio Frequency) received signal is first downconverted to an IF signal. The IF signal is then sampled with a free-running oscillator at rate of $1/T_{\text{in}}$ samples/second. The signal obtained, denoted by $r_{\text{IF}}(mT_{\text{in}})$, is downconverted to baseband by using the same pair of quadrature carriers at frequency f_0 followed by matched filtering to produce the inphase and quadrature signals $x_I[m]$ and $x_Q[m]$. The digital signals $x_I[m]$ and $x_Q[m]$ are passed through a timing recovery loop before being downsampled by U to recover the transmitted QAM data, i.e., $b_I[i]$ for the inphase branch and $b_Q[i]$ for the quadrature branch. In the ideal case of perfect timing, no frequency/phase offset, and no noise, $b_I[i] = a_I[i]$ and $b_Q[i] = a_Q[i]$.

The pulse-shaping filter and the matched filter are implemented to overcome the effect of Inter-Symbol Interference (ISI). One common example to minimize ISI is to shape the pulses $a_I[i]$ and $a_Q[i]$ by a RC (Raised Cosine) filter. To further minimize the effect of AWGN, the RC filter is split evenly between the transmitter and the receiver so the pulse-shaping filter and the matched filter are identical SRRC (Square Root Raised Cosine) filter [10]. Upsampling is the key operation in several digital processing techniques, such as interpolation and timing recovery.

Figure 2.11 shows one example for the inphase component of 4-QAM modulation. In this example, $U = 4$ and the pulse-shaping filter in the transmitter and the matched

[1]For example, in the DOCSIS 3.0 Standard, the downstream frequency range is from 50MHz to 1002MHz and the upstream frequency range is from 5MHz to 85MHz.

filter in the receiver are the same SRRC filter.

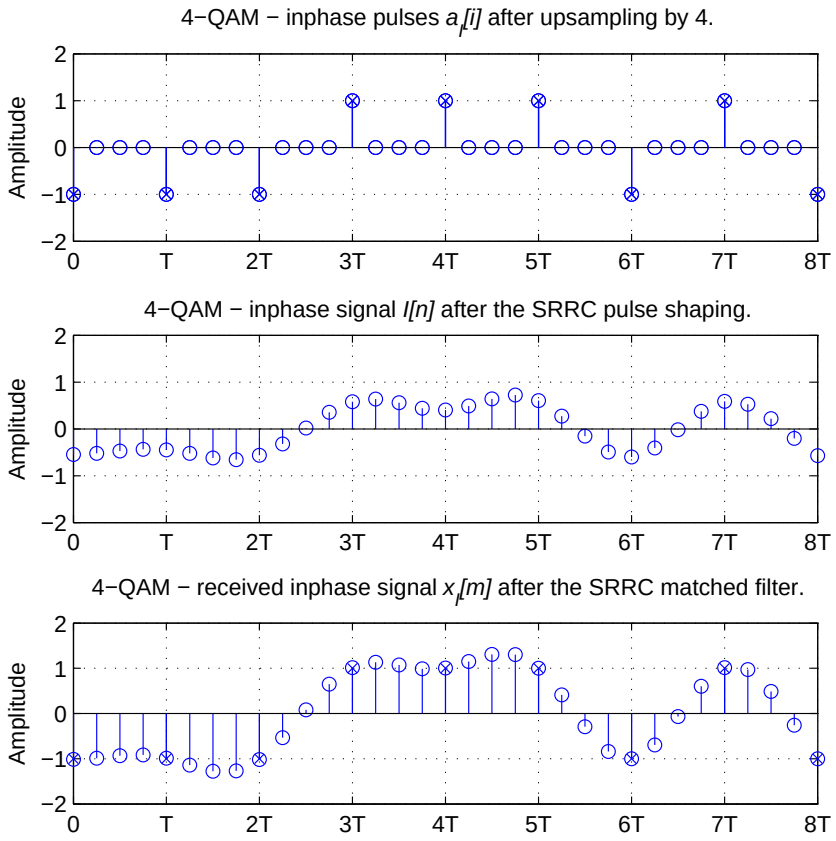

Figure 2.11 Examples of inphase components in a 4-QAM system at different points.

The SRRC filter is an industry-standard pulse shaping filter whose impulse response is given by [12]:

$$h(t) = \begin{cases} 1-\beta+4\beta/\pi, & t=0 \\ \frac{\beta}{\sqrt{2}}\left[(1+\frac{2}{\pi})\sin(\frac{\pi}{4\beta})+(1-\frac{2}{\pi})\cos(\frac{\pi}{4\beta})\right], & t=\pm\frac{T}{4\beta} \\ \frac{\sin\left[\frac{\pi t}{T}(1-\beta)\right]+4\beta\frac{t}{T}\cos\left[\frac{\pi t}{T}(1+\beta)\right]}{\frac{\pi t}{T}\left[1-(4\pi\frac{t}{T})^2\right]}, & \text{otherwise,} \end{cases} \tag{2.16}$$

where T is the input symbol rate and β is the roll-off factor. Figure 2.12 shows samples of the impulse response of a SRRC filter with roll-off factor $\beta = 0.25$. In the figure, the filter impulse response is sampled at $t = K\frac{T}{U}$, which is U times the input symbol rate. U is called the upsamping factor and K is an integer number. In this

example $U = 4$. The filter coefficients are normalized such that $\|h\|^2 = 1$, where h is a vector that holds the filter coefficients.

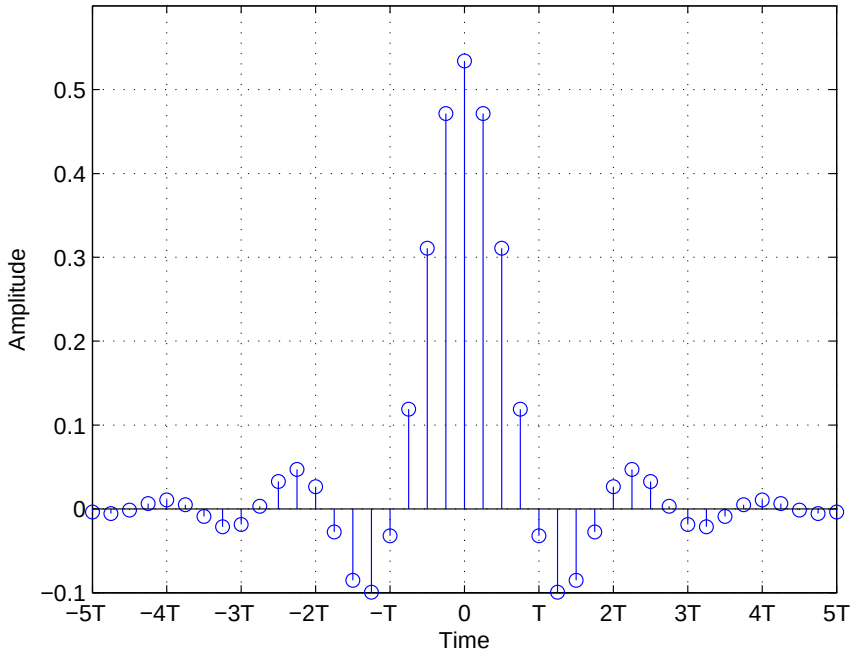

Figure 2.12 Inpulse response of a SRRC filter with roll-off factor $\beta = 0.25$.

Returning to Figure 2.11, for the purpose of illustration, the inphase sample sequences shown in the figure after pulse shaping and matched filter do not take into account the delays caused by the filters. The example demonstrates a perfect case, so the received inphase sample sequence $x_I[m]$ is exactly the same as $a_I[i]$ at the symbol times. Note that in the perfect case, the combined response of the pulse-shaping filter and the matched filter is equivalent to the response of a RC filter. This also implies that the two SRRC filter lengths are not truncated.

The bandwidth of the QAM signal after the matched filter in the perfect case, denoted by B, is given as follows:

$$B = B_{\text{nyquist}} + \Delta B, \tag{2.17}$$

where $B_{\text{nyquist}} = 1/(2T)$ is the Nyquist bandwidth and $\Delta B = \beta/(2T)$ is the excess bandwidth of the filter. The signal bandwidth B has units of cycles per second. Let

F_s denote the sampling frequency and $T_s = T/U$ as the sampling period at both the transmitter and the receiver, then

$$B = \frac{1+\beta}{2T} = \frac{1+\beta}{2UT_s} = \frac{F_s(1+\beta)}{2U}. \tag{2.18}$$

If the sampling frequency F_s is normalized to 1 sample per second, then B becomes a digital bandwidth which has an unit of cycles per sample. For example, with $U = 4$, $\beta = 0.25$, the digital bandwidth B equals to 1.25/8 cycles per sample. Equation (2.18) implies that the higher the sampling frequency, the smaller the digital bandwidth of the signal. A small digital bandwidth has certain advantages when the signal is passed through an interpolation filter (discussed in Appendix A).

In a practical receiver, the timing recovery circuit is needed to remove both the sampling frequency offset and timing offset. The timing recovery process is performed in a feedback loop as shown separately in Figure 2.13.

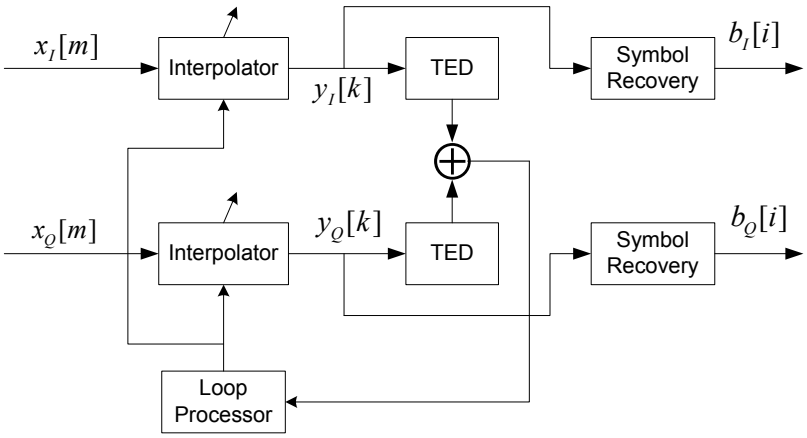

Figure 2.13 Timing recovery loop in a QAM receiver.

The interpolator block is controlled by the *Timing Error Detector (TED)*, which is the block that estimates the timing offset. The *Loop Processor* block recovers the rate M/T and removes the timing offset. Timing recovery is discussed in the next section.

2.3 Digital Timing Recovery

In an all-digital QAM receiver, the oscillator is free-running. The receiver's oscillator is said to be free-running when it is not controllable so that it is synchronized to the oscillator in the transmitter. The sampling rate F_{in} is not an exact multiple of the symbol rate in the transmitter, $1/T$. Thus, there is a difference in frequencies between the oscillators in the transmitter and the receiver. The difference is referred to as *sampling frequency offset*. Also, after the frequency offset has been corrected, the received samples may not be taken at the right time. The difference between correct and current sampling times is defined as *timing offset*. More explicitly, timing offset equals to correct sampling times minus current sampling times. There is another frequency difference which is the difference in frequency between the up and down converter oscillators in the transmitter and receiver (not to be confused with sampling clock oscillators). Even very small, this difference will cause a *carrier frequency offset*, Δf_0. This frequency offset is modeled in the receiver by denoting the frequency of the mixers by $f_0 + \Delta f_0$

Timing offset is illustrated in Figure 2.14, which shows the inphase component of a 4-QAM signal produced by upconverting by 4 the random sequence $a_I[i] = \pm 1$ before being passed through a pulse shape filter in the transmitter and a matched filter in the receiver. Again, both filters are SRRC filters with a roll-off factor $\beta = 0.25$.

Two sets of samples are shown, the samples taken at the right time are marked with a circle, and the samples taken with a fixed timing offset of $-T/16$ are marked with an asterisk. A fixed timing offset is equivalent to saying that the sampling rates for the two sets of samples are the same. In this case, the sampling rate is $4/T$, and the symbol rate is $1/T$, i.e., there are 4 samples per symbol interval. The presence of timing offset degrades system performance, even with the absence of noise.

The carrier frequency offset, sampling frequency offset, and timing offset degrade the performance of the system. Therefore they need to be removed in the receiver by a synchronization circuit. In low-speed telephone-line modems, timing problem is

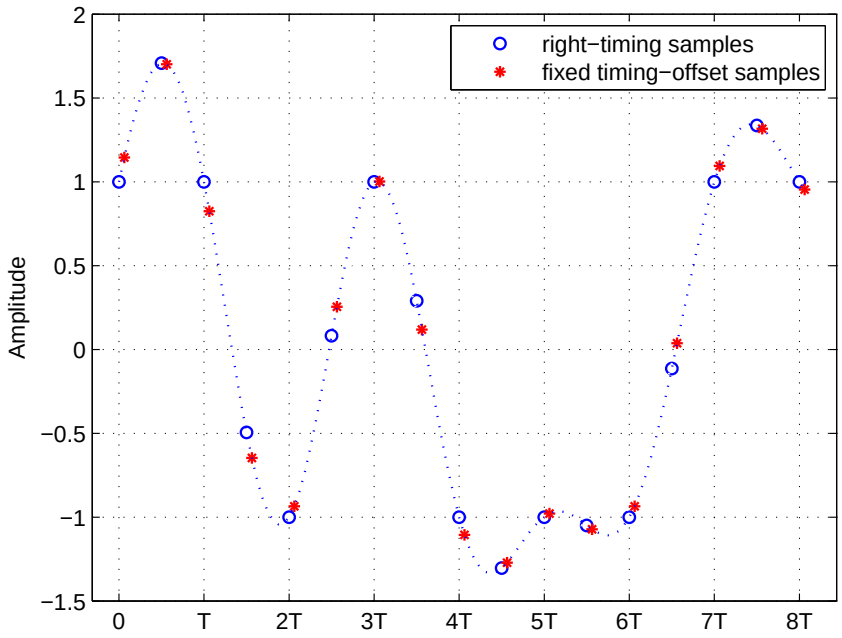

Figure 2.14 Illustration of timing offset for a 4-QAM inphase signal.

solved by an adaptive equalizer which almost incidentally corrects the timing in the progress of correcting for transmission dispersion [13]. But in high-speed communications applications, digital timing adjustment is much more challenging and needs to be solved separately from the equalization process. The synchronization process is discussed in [4] where large carrier frequency offsets are removed before the received signal is passed to the timing recovery circuit. Synchronization to carrier phase at the receiver occurs after timing recovery. At this point the receiver is synchronized and the fine tuning of timing and carrier phase often continues during the symbol detection stage. Timing recovery is one of the topics of interest in this thesis. The basic operation of digital timing recovery circuit using feedback loop is similar to classical analog phase locked loops [4]. The model of timing recovery is shown in Figure 2.15.

The timing recovery circuit implements a digital interpolator to interpolate between the received samples to produce the correct strobe (symbol) values. The strobes are what we call the symbols we want to detect. For example, in our system we will use four interpolants per one strobe (or QAM symbol). The correct strobe values should

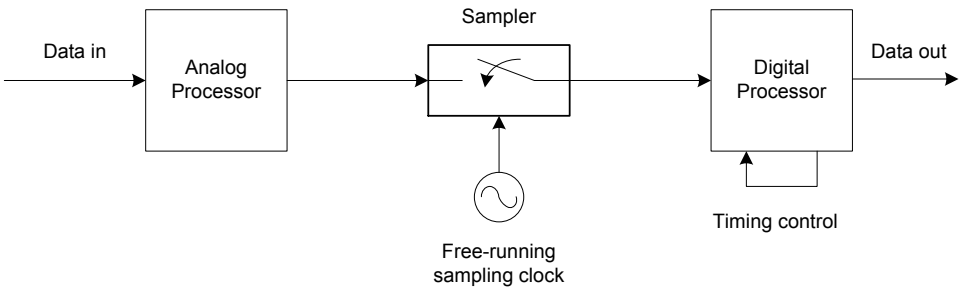

Figure 2.15 Model of digital timing recovery.

be the same as the values that would appear if the sampling had been synchronized to the input symbols.

2.3.1 Timing Recovery Loop

The critical operations in timing recovery are the estimation of the timing offset and interpolating to generate a sample at the right time (the decision time). Several timing recovery circuits have been proposed in the literature to estimate timing offsets. High accuracy in the estimation can be obtained by operating the feedback loop with a small loop bandwidth. This implies that the interpolator introduces little error in the interpolation. Several interpolators have been proposed with diverse levels of complexity and performance.

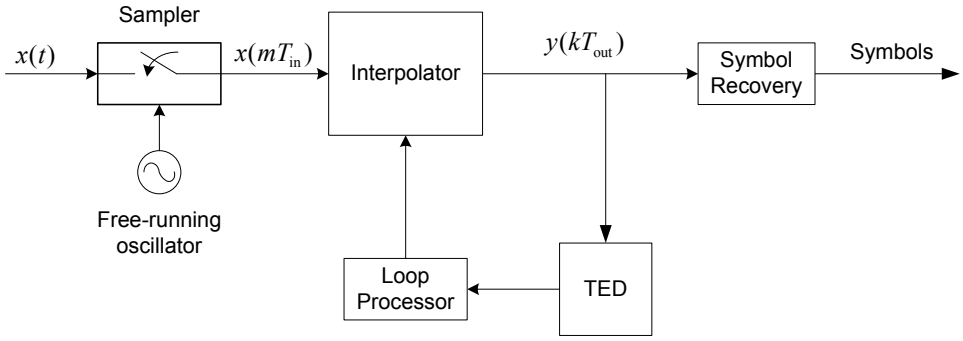

Figure 2.16 Timing recovery loop.

In an actual QAM receiver, the loop contains two interpolators and two timing error detectors. One pair to processes the inphase signal and the other pair processes the quadrature signal. For the purpose of illustration, we will consider the loop with only one interpolator and one timing detector as illustrated in Figure 2.16.

The synchronization process works as follows: the TED estimates the timing offset. Its output is then filtered by the loop processor and then fed to the interpolation controller. The controller uses timing error information to control the interpolator to generate the right interpolants from received samples, $x(mT_{\text{in}})$. The data filter uses the interpolants to generate the right strobes that are used for data and timing recovery.

The above mentioned interpolants refer to the output of the interpolator, $y(kT_{\text{out}})$, which is considered to be at rate $F_{\text{out}} = 1/T_{\text{out}}$. Basically, F_{out} is the rate that the circuit recovers from the sampling rate $F_{\text{in}} = 1/T_{\text{in}}$. In a QAM system that uses 4 samples per symbol and F_{symbol} is the symbol rate, then $F_{\text{out}} = 4F_{\text{symbol}}$. The two control signals that the controller send to the interpolator are enable signals and fractional delays. The enable signal tells the interpolator what set of input samples to use for generating a new interpolant and the fractional delay indicates the position of the new interpolant relative to the received samples.

2.3.2 Gardner Timing Error Detector

The idea behind the Gardner algorithm is to find the zero-crossing point in the output of the interpolating filter. If the current timing estimation is too early from the correct timing, then the timing offset is positive, which indicates that the timing should be advanced. And vice versa, if the current timing is too late, then the timing offset is negative, which means that the timing should be slower. The important point is that the Gardner TED does not require any extra information except the current samples to work (i.e., it is non-data-aided TED). Also, Gardner showed that for a fixed timing offset, the timing error is independent from any carrier phase rotation [11]. These properties make the Gardner TED perfectly suited for high speed QAM applications.

The algorithm was proposed by Gardner to detect timing error that uses only two samples per symbol, and one of the two samples is used for symbol detection [1]. The Gardner TED operates upon samples and generates one error sample $e[l]$ (i.e., timing

error) for each symbol. The index l refers to symbol number. The strobe values of the l^{th} symbol are denoted by $y_{\text{I}}[l]$ and $y_{\text{Q}}[l]$. The pair of samples lying midway between the $(l-1)^{\text{th}}$ and the l^{th} strobes are denoted as $y_{\text{I}}[l-1/2]$ and $y_{\text{Q}}[l-1/2]$.

The detector's algorithm is based on the following formula:

$$e[l] = y_{\text{I}}[l-1/2](y_{\text{I}}[l-1] - y_{\text{I}}[l]) + y_{\text{Q}}[l-1/2](y_{\text{Q}}[l-1] - y_{\text{Q}}[l]) \tag{2.19}$$

Timing error gives information about the current timing offset, which is denoted by $\tau_e[l]$. Let reconsider an example system that uses 4 samples per symbol (i.e., $U = 4$). The input of the Gardner TED (see Figures 2.13 and 2.14) is denoted as $y[k]$. Then one symbol (index l) is detected in every 4 consecutive samples (index k). Gardner TED uses only 2 samples per symbol so the 2 other samples are not needed to compute timing errors. Figure 2.17 shows the samples that are used by the Gardner TED. The set of samples marked with an asterisk are late from their correct values (marked with a circle). The timing error $e[l]$ is negative which indicates that the current timing offset $\tau_e[l]$ is negative and the samples marked with an asterisk should be slower.

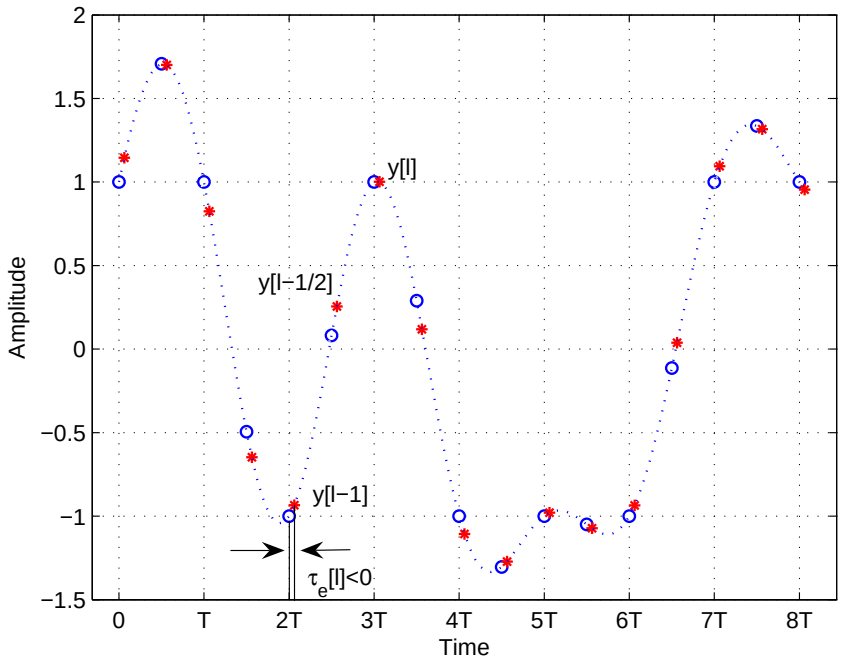

Figure 2.17 4-QAM inphase samples used by the Gardner TED.

As we can see, even the samples at correct timing also produce non-zero timing error. This is because of the randomness of the inphase 4-QAM signal. For alternative symbols ± 1 with 4 samples per symbol, timing error is exactly 0 for samples at correct timing and at every computation, as midway samples are always equal to 0. The Gardner TED still works for random symbols because the expected value of timing error at correct timing is 0. This idea is illustrated by evaluating the input-output characteristic of the detector, or S-curve. Timing error is now presented as a function of timing offset. Note that this is different from Equation (2.19) where the timing error is a function of time as it is digitally computed at every symbol time. The derivation of S-curve for the Gardner TED is discussed in [11]. Figure 2.18 shows the theoretical S-curve for PAM (Pulse Amplitude Modulation) using the SRRC pulse-shaping filter with a roll-off factor $\beta = 0.25$. The curve shows expected timing error for timing offsets varying in one symbol period range.

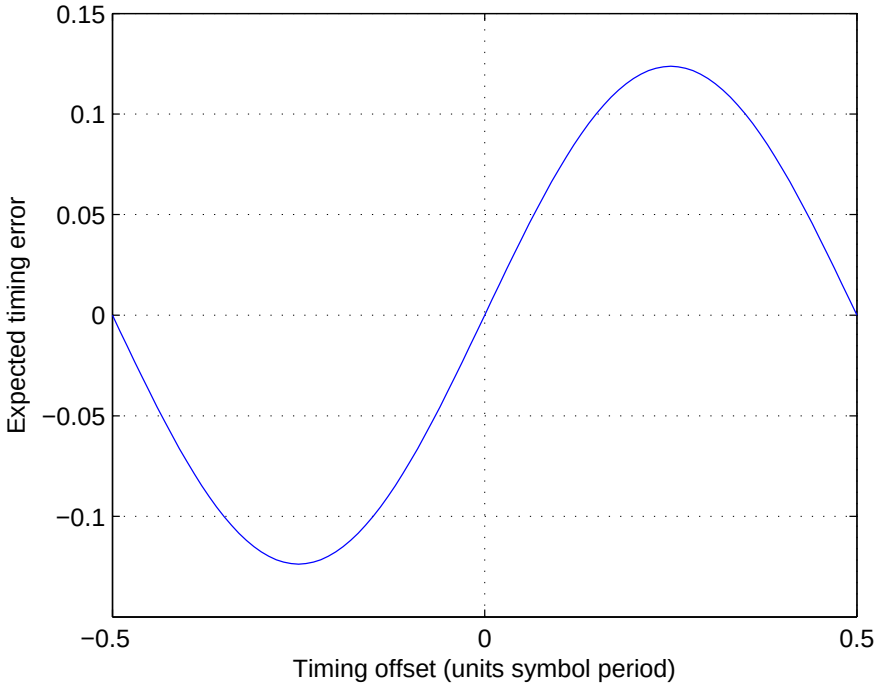

Figure 2.18 Theoretical S-curve of the Garder TED for binary PAM.

The S-curve in Figure 2.18 illustrates that the expected timing error has the same sign as timing offset and is exactly equal to 0 for correct timing (i.e., $\tau_e[l] = 0$). Note

that the Gardner TED requires the transition of signal from positive to negative or vice versa. If there is no transition (no zero-crossing), no timing information is available. The timing error is then used as the control information for the timing recovery loop to find the correct timing.

2.4 Asynchronous Sampling Rate Conversion

In some applications like CATV systems, there are facilities that receive television signals for processing and distribute over a cable television system. Such facilities are called headends. At a headend the incoming signals are received and processed for transmissions over the distribution network. The nature of the system calls for a sampling rate conversion of the received signals. The reason is that the incoming QAM signals are received at different sampling rates, and the processing which combines the QAM signals for transmissions over the network requires that all signals be at the same sampling rate.

The incoming QAM signals originate from different transmitters, and thus have different data rates. The pulse shaping filter in the receiver, which normally has a square-root raised cosine frequency response, is designed to run at an integer multiple of the rate at which the data is received, i.e., the input data rate. The sampling rate of the incoming signal, referred to in the sequel as the input sampling rate, is then an integer multiple of the input data rates. The received signals must then be resampled at a common sampling rate, i.e., the output sampling rate, so they can be combined together and transmitted over the distribution network.

One example of a setup with sampling rate conversion is illustrated in Figure 2.19, which shows N incoming signals, denoted by $x_1[m_1], x_2[m_2], \ldots, x_N[m_N]$, and their respective input sampling clocks denoted by $clk_{\text{in\_1}}, clk_{\text{in\_2}}, \ldots, clk_{\text{in\_}N}$, with the sampling rates $F_{\text{in\_1}}, F_{\text{in\_2}}, \ldots, F_{\text{in\_}N}$. The sampling rate converter resamples the input signals at the sampling rate F_{out}. The output signals $y_1[k], y_2[k], \ldots, y_N[k]$ are then combined to produce $z[k]$, which is then passed to a DAC to produce $z(t)$.

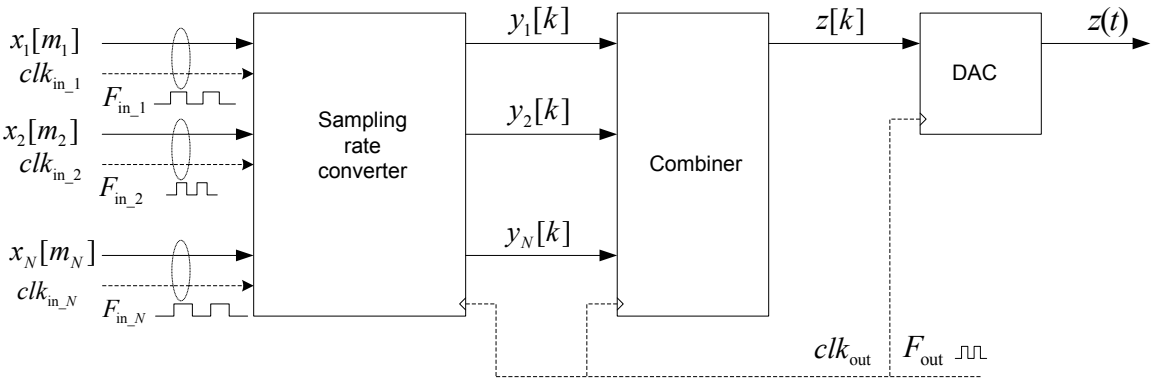

Figure 2.19 Asynchronous sampling rate conversion.

One of the blocks of interest in this thesis is the sampling rate converter (referred to as resampler). Resampling is performed by means of interpolating between the received samples at an input sampling rate to produce output interpolants at an output sampling rate. The key operation of resampling is to find the distance (referred to as fractional interval) between an output interpolant and a referenced input sample (referred to as basepoint index) at each interpolation. The method in this thesis is to generate two time bases (or accumulators) from the input sampling clock and the output sampling clock and then synchronize the two time bases by means of a phase locked loop. There are naturally two approaches to synchronize the time bases. One approach, referred to as resampling with input clock time base, is to use the output clock time base as a reference to find the input clock time base. The other method, referred to as resampling with output clock time base, is to use the input clock time base as a reference to find the output clock time base. The detailed methods and approaches will be discussed in Chapter 3.

2.5 Summary

This chapter has discussed fundamental concepts of digital communications that serves as background of the thesis. The first part reviewed digital modulation, QAM, and important performance criteria of QAM systems. The second part described the block diagram of QAM systems and discussed the architecture of QAM transmitters and receivers. The third part summarized basic ideas of timing recovery in QAM re-

ceivers and the last part considered digital sampling rate conversion that is performed in QAM headends.

3. Asynchronous Sampling Rate Conversion

An asynchronous sampling rate converter, or resampler performs a rate conversion from $F_{\text{in}} = 1/T_{\text{in}}$ to $F_{\text{out}} = 1/T_{\text{out}}$, where F_{in} and F_{out} are incommensurate with each other. In particular, resampling consists of taking a sequence of samples, $x[m] \equiv x(mT_{\text{in}})$, at sampling rate, $F_{\text{in}} = 1/T_{\text{in}}$, and generating a new sequence, $y[k] \equiv y(kT_{\text{out}})$ by means of interplation between the samples of $x[m]$. The samples $y[k]$ are also referred to as the interpolants. This is conceptually described by the block diagram in Figure 3.1.

Figure 3.1 Model of resampling.

This chapter introduces the important concepts of time base and time base synchronization in hardware that will be used to devise the circuits for asynchronous sampling rate converters. The methodology will also be used for timing recovery circuits in Chapter 4. Though the approach is mainly presented for the operation of asynchronous sampling rate converters, it can be applied to timing recovery with some modifications.

3.1 Theory of Digital Interpolation

The mathematical model for digital interpolation given in [13] is reviewed by considering a hybrid method of rate conversion in Figure 3.2.

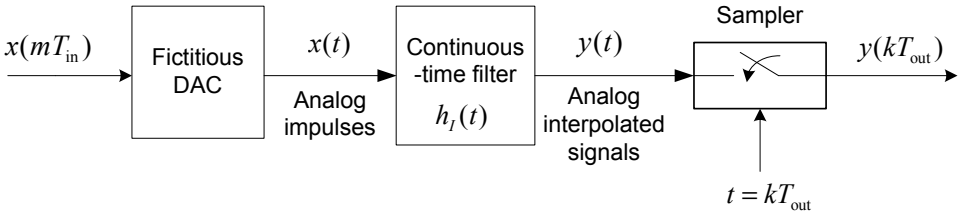

Figure 3.2 Block diagram of rate conversion with continuous-time filter.

In this thesis, we use the notations $F_{\text{in}} = 1/T_{\text{in}}$ as the input rate and $F_{\text{out}} = 1/T_{\text{out}}$ as the output rate of an interpolator. It would be useful to stress that interpolation is the operation that adjusts the sampling times of the signal and does not change a local sampling clock or timing wave. To illustrate the concept of resampling, in Figure 3.2, the input samples are converted to a sequence of analog impulses, which is then filtered by a continuous-time, analog interpolating filter. The output of the filter is

$$y(t) = \sum_{m=-\infty}^{\infty} x(mT_{\text{in}}) h_I(t - mT_{\text{in}}), \tag{3.1}$$

where m is the input sample index. Let k be the index of the output interpolants. The interpolants $y(kT_{\text{out}})$ are obtained by evaluating $y(t)$ at time instants $t = kT_{\text{out}}$, i.e.,

$$y(kT_{\text{out}}) = \sum_{m=-\infty}^{\infty} x(mT_{\text{in}}) h_I(kT_{\text{out}} - mT_{\text{in}}),\ k = 0, 1, ... \tag{3.2}$$

Equation (3.2) suggests that the interpolants can be computed *digitally* from the knowledge of:

- the time instants mT_{in} and the input sequence $x(mT_{\text{in}})$,
- the impulse response of the interpolating filter, $h_I(t)$,
- the time instants kT_{out}.

The interpolants resulting from digital calculation are identical to the values from analog calculation. To give insight into Equation (3.2), express the argument of impulse response $h_I(\cdot)$ as $h_I((kT_{\text{out}}/T_{\text{in}} - m)T_{\text{in}})$. Then let

$$i = \underbrace{\text{int}[kT_{\text{out}}/T_{\text{in}}]}_{=m_k} - m = m_k - m, \tag{3.3}$$

which can be viewed as an integer index, while m_k is known as the basepoint index [1]. Then decompose $(kT_{\text{out}} - mT_{\text{in}}) = (i + \mu_k)T_{\text{in}}$ where

$$\mu_k = kT_{\text{out}}/T_{\text{in}} - m_k. \tag{3.4}$$

is the fractional interval. This allows $h_I(kT_{\text{out}} - mT_{\text{in}})$ to be expressed as $h_I((m_k - m)T_{\text{in}} + \mu_k T_{\text{in}})$. It is clear that $(m_k - m)T_{\text{in}}$ coincides with an input sample time and μ_k is an offset from that sample time.

Assume $k > 0$ so $kT_{\text{out}}/T_{\text{in}}$ is positive and $0 \leq \mu_k \leq 1$. Suppose, the filter index i is in the range from I_1 to I_2. From the above definition, arguments in Equation (3.2) become $mT_{\text{in}} = (m_k - i)T_{\text{in}}$, $(kT_{\text{out}} - mT_{\text{in}}) = (i + \mu_k)T_{\text{in}}$. Moreover, m is now in the range from $m_k - I_2$ to $m_k - I_1$. The set of $I = I_2 - I_1 + 1$ samples in the range from $m_k - I_2$ to $m_k - I_1$ is defined as the k^{th} basepoint set. The interpolants are computed at time $kT_{\text{out}} = (m_k + \mu_k)T_{\text{in}}$ and the interpolation equation becomes

$$y(kT_{\text{out}}) = \sum_{i=I_1}^{I_2} x((m_k - i)T_{\text{in}})h_I((i + \mu_k)T_{\text{in}}). \tag{3.5}$$

The timing relations between samples and interpolants in Equation (3.5) are depicted in Figure 3.3 for the case $I_1 = -2$ and $I_2 = 1$. In this case, the computation of $y(kT_{\text{out}})$ requires 4 input samples with index m from $m_k - 1$ to $m_k + 2$.

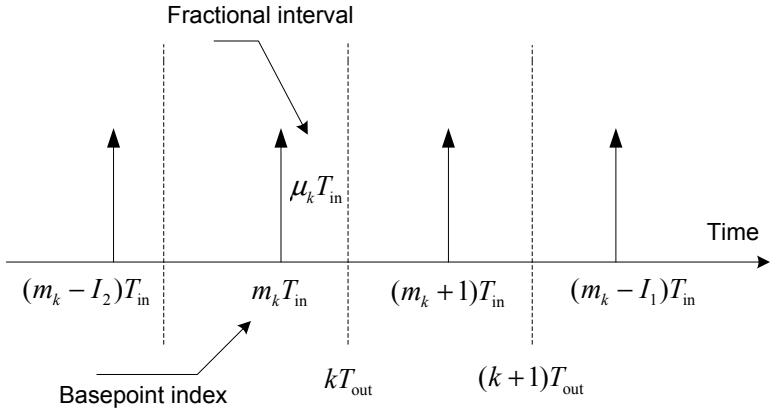

Figure 3.3 Sample time relations, $I_1 = -2$ and $I_2 = 1$.

Interpolation Filters

An interpolation filter can be designed using many types of mathematical functions, among which the most efficient one for hardware implementation is polynomial [2]. The simplest polynomial interpolation is linear interpolator, which has the degree of one and interpolates between two samples. With a linear interpolator, the filter has two coefficients. Let set $I_1 = -1$ and $I_2 = 0$. Then $I = 2$ and the interpolation filter can be described by Lagrange coefficients, which are polynomials of degree 1 in μ_k as follows:

$$\begin{cases} h_I[\mu_k] = 1 - \mu_k \\ h_I[-1 + \mu_k] = \mu_k \end{cases} \tag{3.6}$$

The interpolating equation (3.2) reduces to

$$y[k] = x[m_k](1 - \mu_k) + x[m_k + 1]\mu_k, \tag{3.7}$$

where T_{in} has been set to 1 with no loss of generality. The computation of $y[k]$ using (3.7) is illustrated in Figure 3.4 where $y[k]$ is the point on the line passing through $x[m_k]$ and $x[m_k + 1]$ at a distance of $\mu_k T_{\text{in}}$ from $x[m_k]$.

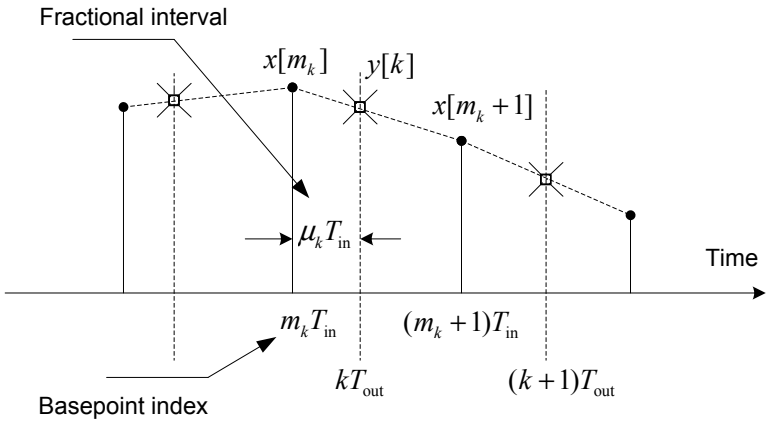

Figure 3.4 Linear interpolation.

Interpolating polynomial filters offer good filter characteristics in stopband response and passband response [2]. Also, there is a special FIR structure, namely Farrow structure, that allows simple handling of filter coefficients [3]. Farrow structure is applicable only to polynomials. The structure is discussed in Chapter 5. Good

performance and effective hardware structure make polynomial filters widely chosen in practise. Mathematical expressions and frequency responses of interpolating polynomial filters are discussed further in Appendix A.

Interpolation Error

Obviously, interpolation filter introduces errors to the signal. The error for the case of linear interpolation is depicted in Figure 3.5. As can be seen, the linear interpolator does not perform very well in this scenario. In the ideal situation, $y(kT_{\text{out}})$ should be on the $x(t)$ curve. The linear interpolator does not provide good interpolation on curved functions unless the sample rate is very high.

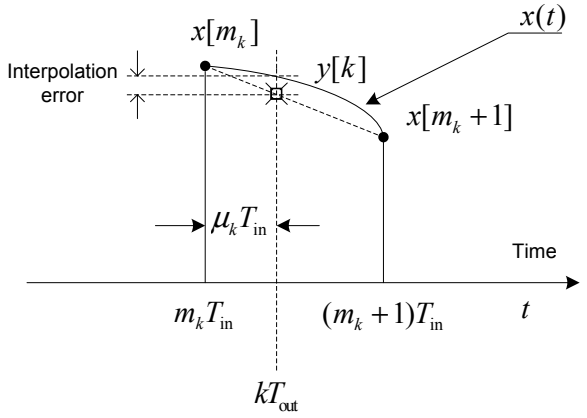

Figure 3.5 Linear interpolation error.

Interpolation error obviously ties to a certain type of signal. In our system, QAM signal is taken into consideration when analyzing the error of different types of interpolation filters. The performance of polynomial interpolation is evaluated in timing recovery application based on the MER criterion. As discussed in Chapter 2, MER is a very useful indicator of the quality of received signals in QAM systems. The circuit to evaluate MER performance of interpolating polynomial filters is presented in Chapter 4.

3.2 Resampling Operations in Hardware

With the background of digital interpolation presented in the previous section, this section discusses the operations of digital resampling in hardware and then devises practical resampling circuits that are suitable for hardware implementation. The model of resampling in Figure 3.1 is now explored further with the presence of an interpolator block and an interpolator controller block as shown in Figure 3.6.

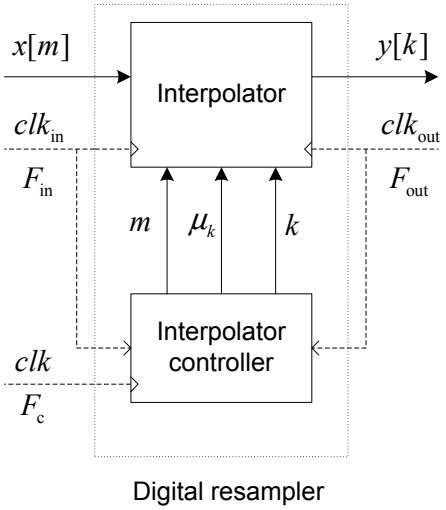

Figure 3.6 Resampler.

A new clock signal is introduced in Figure 3.6, which is the system clock clk with rate F_{c}. For simplicity, the figure does not show clock domain interfaces between clk and clk_{in}, and clk_{out}. The interpolator controller generates input sample index m, the fractional interval μ_k, and the output interpolant index k, and makes them available to the interpolator. m can be seen as an input sample enable signal and k can be seen as an output sample enable signal. The base point index m_k is determined from m and k. Note that the basepoint index m_k identifies the $I = I_2 - I_1 + 1$ signal samples while the fractional interval μ_k identifies the I filter coefficients. The I filter coefficients are convolved with the I input samples to generate the new interpolant.

Figure 3.7 depicts the sampling time with respect to the input clock and the output clock. In the figure, one input sample is loaded into the resampler at every positive edge of clk_{in} and one output interpolant is *expected* from the resampler at

every positive edge of clk_{out}. The figure ignores delays caused by the resampler as in a resampler circuit, $y[k]$ is not available immediately after $x[m_k]$. While the operations and hardware structure of a polynomial interpolator are straightforward by using the Farrow structure, the focus of this section is on hardware operations and circuit designs of the interpolator controller. As discussed in Chapter 1, the methodology used in this thesis is based on the idea of time base generation and synchronization [7], which will be presented next.

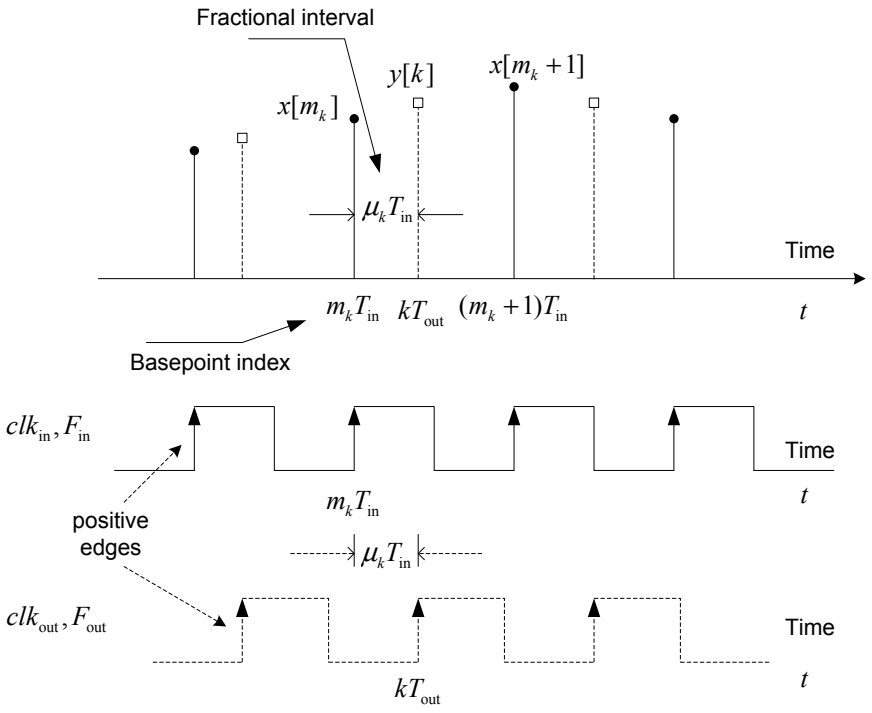

Figure 3.7 Sampling time relation.

3.2.1 Time Base Generation

Let's start by quantizing the time t with two approaches. In the first approach, the quantized value of t is called the *input clock time base* and it is denoted by Y_{in}. Y_{in} is incremented by T_{in} for every T_{in} period. In the second approach, the quantized value of t is called the *output clock time base* and it is denoted by Y_{out}. Similarly, Y_{out} is incremented by T_{out} for every T_{out} period. Obviously, both Y_{in} and Y_{out} have units of seconds.

In hardware, Y_{out} can be generated by an accumulator which is clocked by the

output clock at frequency F_{out} (units of samples/second). Figure 3.8 shows the circuit for the time base generator clocked at F_{out}, i.e., the output clock time base Y_{out}. Y_{out} is produced by incrementing the content of the register, or accumulator, by T_{out} at every clock cycle, F_{out}. As it is impossible to acquire the exact value of a real number, T_{out} is estimated by $\hat{T}_{\text{out}}$ in hardware. As a consequence, Y_{out} is the quantized value of time t with step size $\hat{T}_{\text{out}}$.

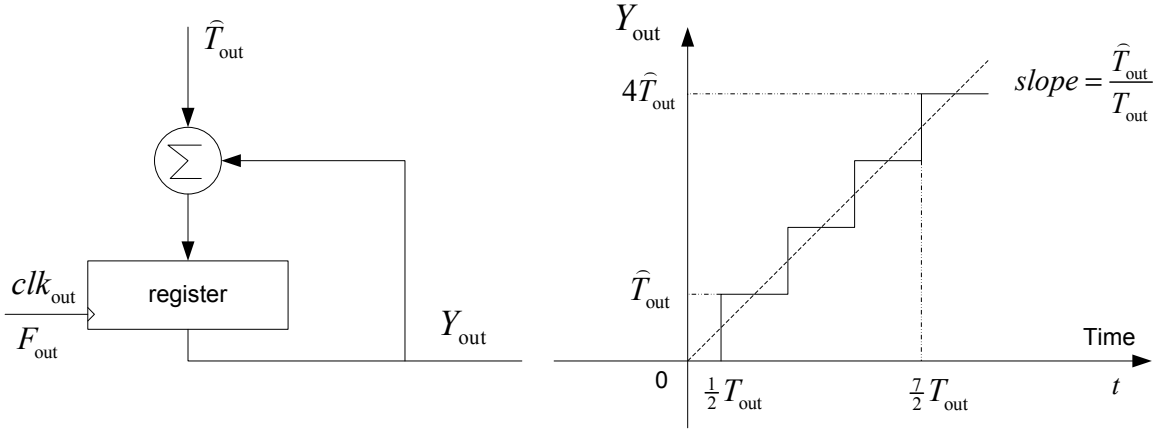

Figure 3.8 Output clock time base generator.

Mathematically Y_{out} is given by

$$Y_{\text{out}} = \hat{T}_{\text{out}} \times \text{int}\left[\frac{t}{T_{\text{out}}}\right] = \frac{\hat{T}_{\text{out}}}{T_{\text{out}}}t + n_{\text{out}}, \tag{3.8}$$

where $n_{\text{out}} = -\hat{T}_{\text{out}} \times \text{frac}[t/T_{\text{out}}]$ is the quantization noise and frac$[\cdot]$ is the fractional part of a real number. The circuit for the input clock time base Y_{in} is identical. Y_{in} is generated by F_{in} with step size $\hat{T}_{\text{in}}$. Again, $\hat{T}_{\text{in}}$ is a close estimate of T_{in} in hardware. Similar to Y_{out}, Y_{in} is the quantized value of time t with step size $\hat{T}_{\text{in}}$ and is given by

$$Y_{\text{in}} = \frac{\hat{T}_{\text{in}}}{T_{\text{in}}}(t - \Delta Y) + n_{\text{in}}, \tag{3.9}$$

where ΔY represents the timing offset between input and output clocks and $n_{\text{in}} = -\hat{T}_{\text{in}} \times \text{frac}[(t - \Delta Y)/T_{\text{in}}]$ is the quantization noise. Figure 3.9 depicts the hardware construction of Y_{in}.

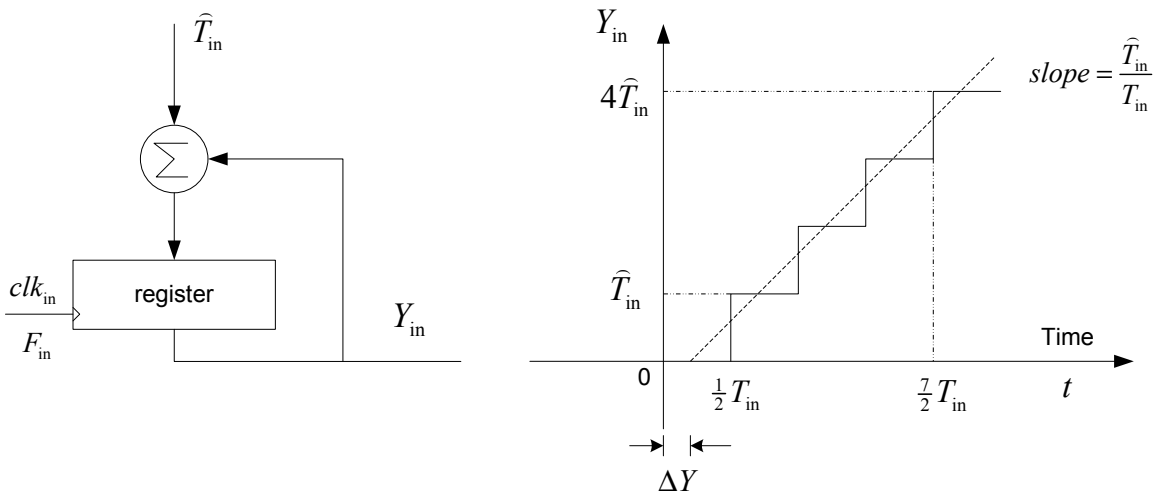

Figure 3.9 Input clock time base generator.

3.2.2 Time Base Synchronization

Since both Y_{in} and Y_{out} represent quantized values of time t with different step sizes, they are synchronized if and only if

$$Y_{\text{out}} - Y_{\text{in}} = n_{\text{out}} - n_{\text{in}}. \tag{3.10}$$

This happens when the slope of the output clock time base ramp, Y_{out}, (see Figure 3.8) is equal to the slope of the input clock time base ramp, Y_{in} (see Figure 3.9). Both slopes can be made equal by modifying the step size of one of the time base generator by an appropriate amount. The right amount is estimated by means of a phase lock loop (PLL). In other words, the PLL can either estimate $\hat{T}_{\text{in}}$ or $\hat{T}_{\text{out}}$ so that $\hat{T}_{\text{in}}/T_{\text{in}} = \hat{T}_{\text{out}}/T_{\text{out}}$ after the PLL has converged. The timing offset, ΔY, is removed in the process of finding the right step size. More precisely, the PLL is set up with reference $\hat{T}_{\text{out}}/T_{\text{out}}$ and feedback output $\hat{T}_{\text{in}}/T_{\text{in}}$ or vice versa.

Using PLL solves the problem of rate changing and step estimation, as it synchronizes and makes the circuit adapt to the change in the input sampling rate or in the output sampling rate. Either Y_{out} is taken as the reference to estimate $\hat{T}_{\text{in}}$, or Y_{in} serves as the reference to estimate $\hat{T}_{\text{out}}$. Circuits are given for both methods in the next section.

3.3 Resampling Circuits

The method to generate and synchronize input clock time base and output clock time base is applied to devise the circuits for asynchronous sampling rate conversion. The circuits, referred to as resamplers, are suitable for hardware implementation. The sampling rate converter circuit which uses the output ramp as the reference is referred to as a "resampler with input clock time base". Likewise, the circuit which uses the input ramp as the reference is referred to as a "resampler with output clock time base".

3.3.1 Resampler with Input Clock Time Base

For this case the input clock time base is constructed from the output clock time base by using the output clock time base as a reference in a PLL. This method is based on the input time domain T_{in}. In this case the rate of the system clock *clk* in Figure 3.6 must be an integer multiple of the input sampling rate, or $F_{\text{c}} = MF_{\text{in}}$, where M is an integer number which is taken sufficiently large so $F_{\text{c}} > F_{\text{out}}$. As the circuit is clocked by *clk*, the unit of time in the circuit is T_{in}/M.

The resampler needs to know the expected output sampling rate F_{out}, or equivalently, the output clock time base Y_{out}. Y_{out} is then used as the reference to construct/estimate the input clock time base Y_{in}. Y_{in} is then compared with Y_{out} to find k and compute μ_k.

The time base generators are set up as follows. Output clock time base Y_{out} is normalized by fixing the input of the accumulator (i.e., step size) in Figure 3.8 to $\hat{T}_{\text{out}} = 1$. This causes the output, Y_{out}, to be scaled by $1/T_{\text{out}}$.

$Y_{\text{out}}/T_{\text{out}}$ is now time t scaled by $1/T_{\text{out}}$ and then quantized with step size 1. Next the input clock time base is modified to also generate quantized values of t/T_{out} as shown in Figure 3.11.

The step size of the input clock time base is the estimate produced by the PLL to synchronize input and output time bases. In hardware, Y_{in} is constructed by the

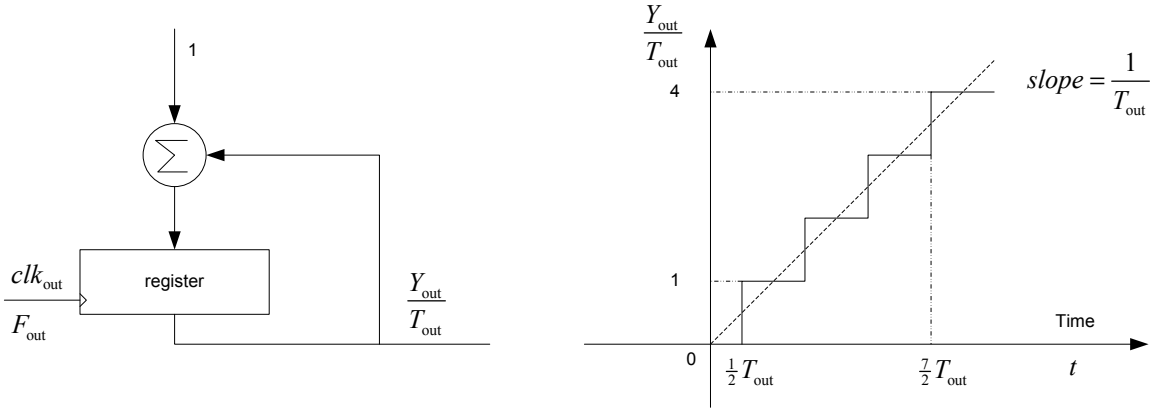

Figure 3.10 Output clock time base as a reference.

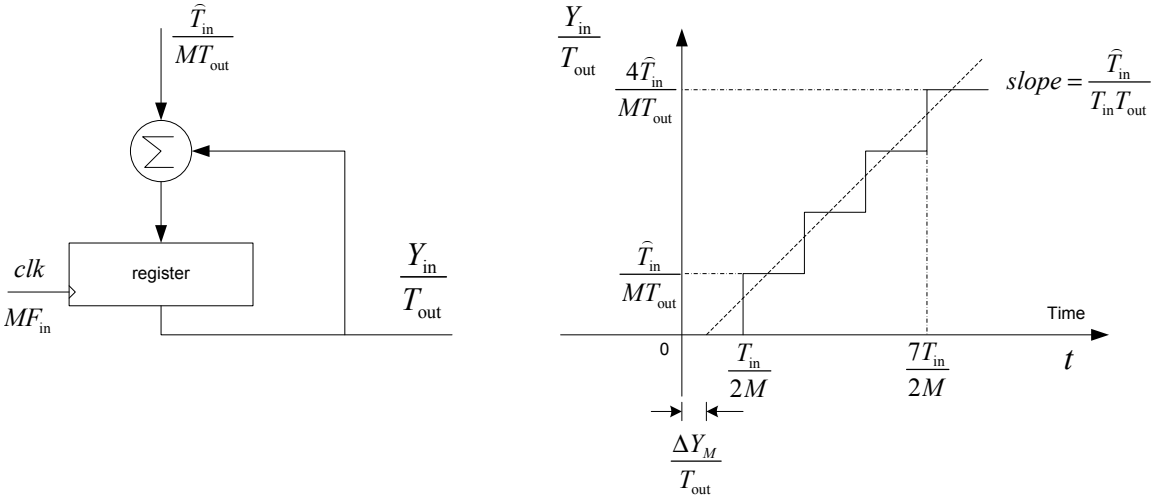

Figure 3.11 Input clock time base scaled by output rate.

clock with frequency $F_{\text{c}} = MF_{\text{in}}$. It is now $(t - \Delta Y_M)/T_{\text{out}}$ quantized with step size $\hat{T}_{\text{in}}/(MT_{\text{out}})$, where ΔY_M is the timing offset between clk and clk_{out} as indicated in Figure 3.11.

The normalization process makes the circuit much simpler, as the input clock time base can be compared with integer numbers to generate k and μ_k. A block diagram of the resampler with input clock time base is shown in Figure 3.12. It has three inputs, $x[m]$, clk_{in} and clk_{out} and one output, $y[k]$.

In Figure 3.12 a clock at frequency $F_{\text{c}} = MF_{\text{in}}$ is synthesized from clk_{in} using a built-in PLL in the FPGA. M must be chosen large enough so $F_{\text{c}} > F_{\text{out}}$. The block labeled "Output clock time base generator" is the circuit in Figure 3.10. The block

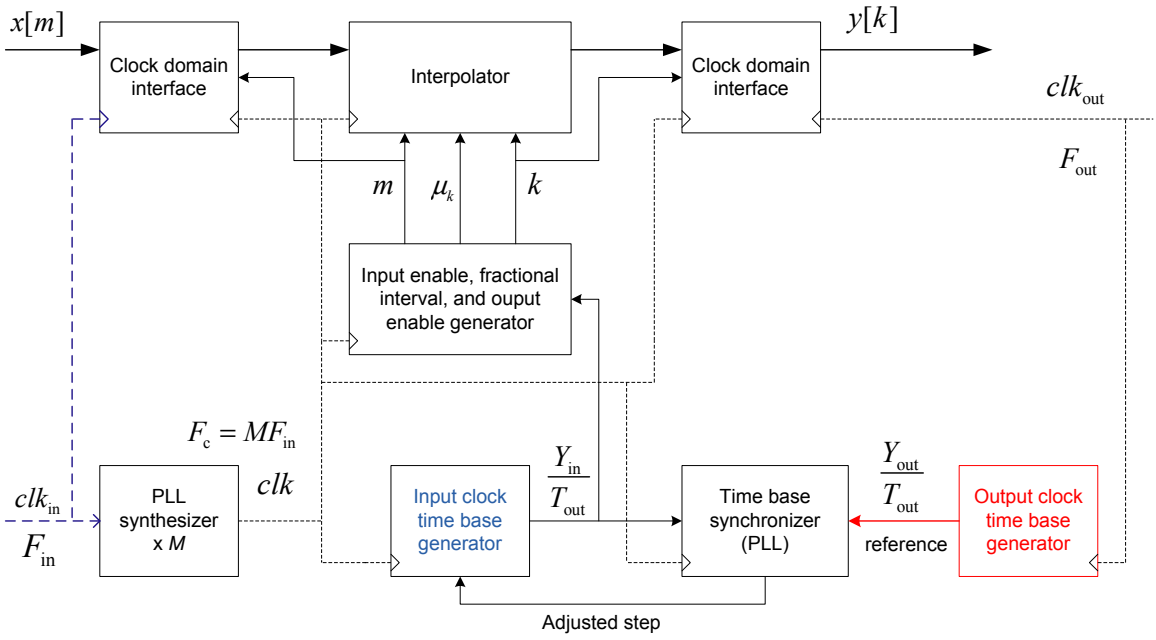

Figure 3.12 Resampler with input clock time base.

labeled "Input clock time base" is the circuit in Figure 3.11. Its input, denoted by "adjusted step" in Figure 3.12, is generated by the "Time base synchronizer" block. This block is the PLL that is set up to estimate $\hat{T}_{\text{in}}/(MT_{\text{out}})$, which is the amount by which Accumulator 1 is to be incremented on each edge of *clk*, which has frequency MF_{in}.

A model for the PLL is shown in Figure 3.13, where the phase comparator is simply the difference between input and output ramps. Accumulator 1 is the "Input clock time base" block in Figure 3.12, and Accumulator 2 is an additional accumulator that is needed to hold $\hat{T}_{\text{in}}/(MT_{\text{out}})$. Accumulator 2 builds up during the acquisition phase to settle to $\hat{T}_{\text{in}}/(MT_{\text{out}})$. The PLL operates as follows [14]. The average value of the input to Accumulator 1 must be exactly $\hat{T}_{\text{in}}/(MT_{\text{out}})$ for the loop to be locked. The gain G_1 is much larger than G_2 so that at start up the loop looks as if the leg with G_2 is not presented. After locking the error signal will fluctuate somewhat wildly but it will have a DC value of $\hat{T}_{\text{in}}/(G_1MT_{\text{out}})$ as it must for the loop to be locked. This DC value causes Accumulator 2 to ramp up or down depending on whether the error is positive or negative. The input to the Accumulator 1 must have an average

value of $\hat{T}_{\text{in}}/(MT_{\text{out}})$ for the loop to remain locked so the DC value at the output of G_1 diminishes as the output of Accumulator 2 grows. In steady state, the error has no DC value Accumulator 2 neither grows or diminishes. The output of G_1 becomes AC noise. Of course, the leg with G_1 is necessary to make the PLL stable.

In the steady state, the two time bases are synchronized. They are both quantized representations of t/T_{out}, one with a step size of 1, and the other with a step size of $\hat{T}_{\text{in}}/T_{\text{out}}$.

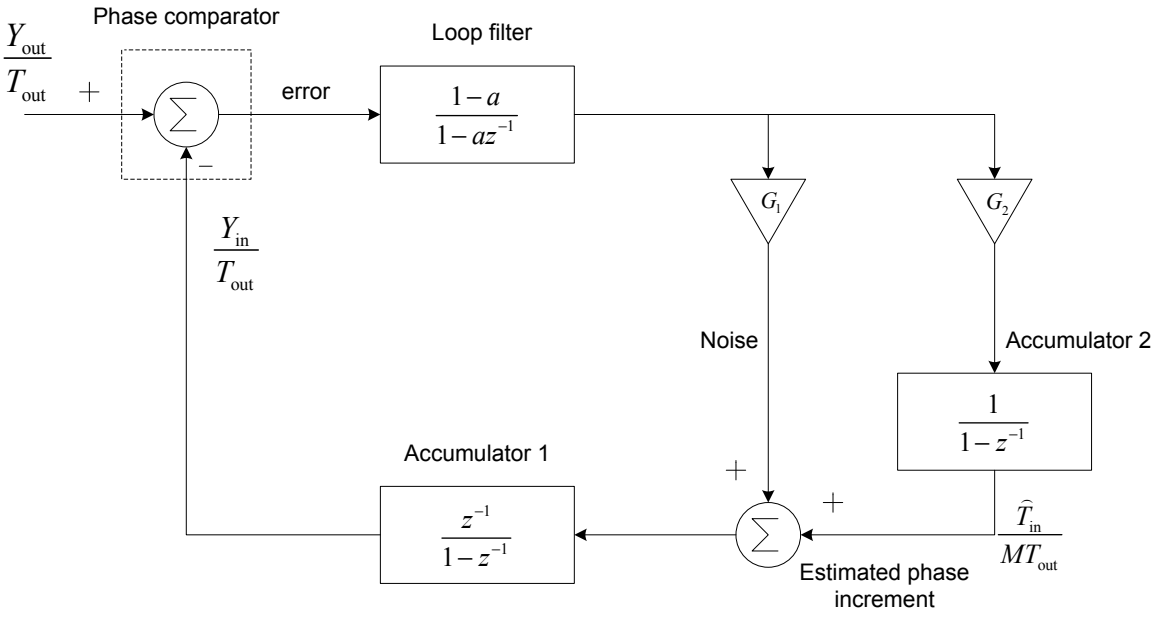

Figure 3.13 Setting up the PLL with all registers (i.e., z^{-1}) clocked at rate MF_{in}.

The presence of different clocks inside the circuit in Figure 3.12 calls for a special interface, which allows samples to be written and read using different physical clocks. The average writing and reading rates are the same but the access to the samples inside the structure, represented by the block labeled "Clock domain interface" in Figure 3.12, occurs at different instants of time. Such a structure is best implemented with a circular buffer and logics to control the access of the buffer. Two of these blocks are needed, one to interface the clocking of the input samples and the interpolator and the other one to interface the output of the interpolator and the clocking of the output samples.

The last block which remains to be described in Figure 3.12 is the "Input enable,

fractional interval, and output enable generator" block. This module uses the reconstructed input ramp, $Y_{\text{in}}/T_{\text{out}}$, to produce m, μ_k, and k. m is basically an input enable signal for registers that store the value of $x[m]$ and k is an output enable signal for registers that store the value of $y[k]$. The k^{th} output interpolant is generated from the basepoint index m_k and the fractional interval μ_k. The generations of m_k and μ_k are described as follows.

When the value of $Y_{\text{in}}/T_{\text{out}}$ crosses an integer number, the output enable signal k is set to high. It is set to low otherwise. When k is high (i.e., an integer cross-over was detected), the last received input sample index m is identified as the basepoint index m_k. The new interpolant is generated at positive edges of F_{c} when the output enable signal k is high. Most of the time, especially if M is large, this clock edge does not coincide with a positive clock edge of clk_{in}. The situation with $M = 4$ is depicted in Figure 3.14, where the top graph shows the position of $y[k]$ with respect to the input samples, the middle graph shows all three clocks and the bottom graph shows the ramps where both the integer crossing and fractional part (i.e., $\text{frac}[\frac{Y_{\text{in}}}{T_{\text{out}}}]$) which enters in the computation of μ_k have been identified.

The computation for μ_k depends on which clock edge of F_{c} the integer cross-over was detected. Since F_{c} is M times faster than F_{in}, M clock cycles of F_{c} occur between one clock cycle of F_{in}. Suppose that between input sampling times $m_k T_{\text{in}}$ and $(m_k + 1)T_{\text{in}}$, an integer cross-over is detected at the l_k^{th} positive clock edge of F_{c}. Then μ_k is given by

$$\mu_k = \frac{l_k}{M} - \frac{T_{\text{out}}}{\hat{T}_{\text{in}}} \times \text{frac}\left[\frac{Y_{\text{in}}}{T_{\text{out}}}\right]. \tag{3.11}$$

Figure 3.14 shows the case for $M = 4$, $l_k = 1$, $l_{k+1} = 3$. There is a division in (3.11) which needs explanation, as it is not required in the second circuit to be described later, and causes additional complexity in the calculation of μ_k. The fractional part, $\text{frac}[Y_{\text{in}}/T_{\text{out}}]$, which is used to compute the time that separates the new interpolant from the input sample, is expressed as a fraction of the output clock cycle. This time must be converted in units of fraction of the input clock cycle to yield μ_k. This is

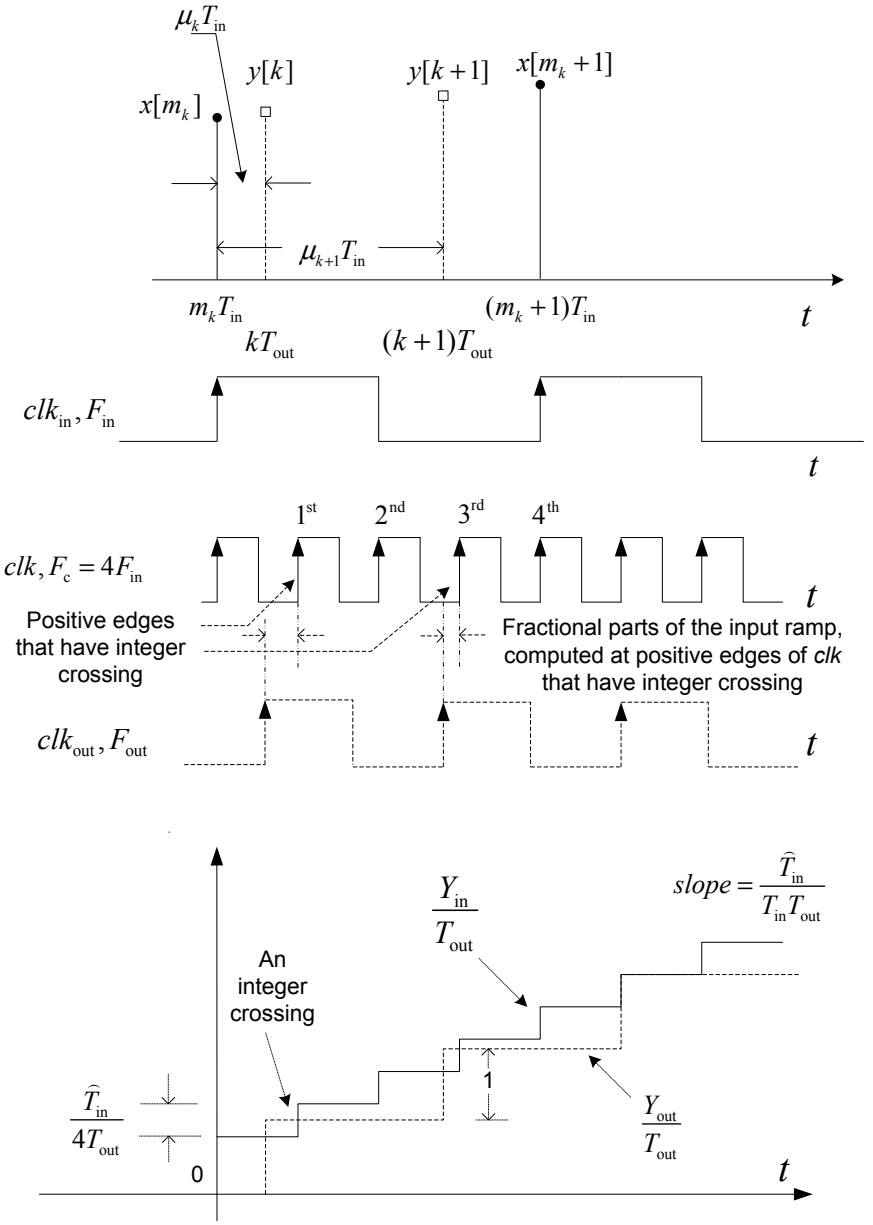

Figure 3.14 Timing relation in input clock time base resampling.

achieved in (3.11) by dividing $\text{frac}[Y_{\text{in}}/T_{\text{out}}]$ by $\hat{T}_{\text{in}}/T_{\text{out}}$.

Note that k determines the writing rate and F_{out} determines the reading rate of the "Clock domain interface" at the output in Figure 3.12, i.e., the interface for $y[k]$. When the step size has been estimated and the two time bases have been synchronized, the resampler is operated in the steady state. In the steady state, the output enable signal k is high on the average at the same rate as F_{out}, which makes the writing and

reading rates the same.

3.3.2 Resampler with Output Clock Time Base

In this method, Y_{in} is used as the reference to construct Y_{out}, which is then used to generate m_k, k and μ_k. The circuit is shown in Figure 3.15. Here the system clock, clk at frequency $F_{\text{c}} = M \times F_{\text{out}}$ is synthesized, where M is taken sufficiently large so that $F_{\text{c}} > F_{\text{in}}$. In case the output sampling rate is higher than the input rate, M equals to 1 is enough and no PLL-synthesizer is needed.

In this circuit, the step size of the "Input clock time base generator" block is set to 1 and the step size of the "Output clock time base generator" is determined with the PLL.

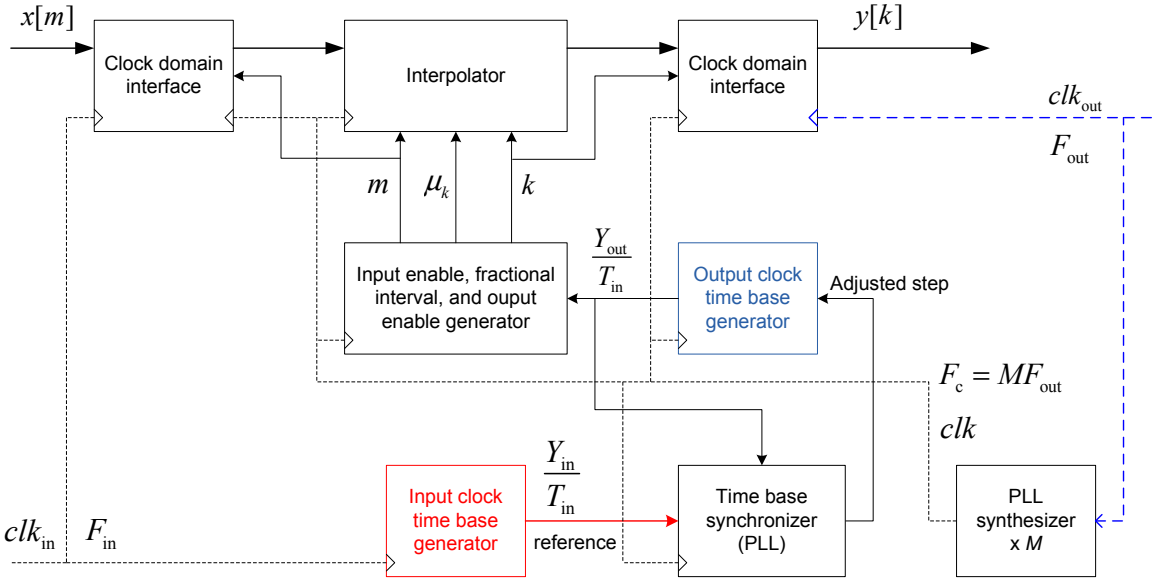

Figure 3.15 Resampler with output clock time base.

Compared to the circuit in Figure 3.12, the main difference is that a new interpolant is produced at every positive edge of the resampling clock F_{out}. This means the output enable signal k now is set to high at every positive edge of clk_{out}. It is low otherwise. In this case, whenever an integer cross-over in the reconstructed output ramp is detected, m is set to high and a new input sample is loaded into the interpolator. m is low otherwise. The basepoint index identified by m_k is now the

latest loaded input sample index.

In other words, the roles of m and k are exchanged in the two approaches. In the first approach, k is set accordingly to the time when an integer cross-over is detected. In the current approach, m is set instead. In the first approach, m is set to high at every positive edge of clk_{in} and low otherwise. In the current approach, k is set to high at every positive edge of clk_{out} and low otherwise.

The fractional interval is simply given by

$$\mu_k = \text{frac}\left[\frac{Y_{\text{out}}}{T_{\text{in}}}\right]. \tag{3.12}$$

In this case there is no division since the fractional part is already expressed as a fraction of the input clock cycle. Recall that in this method the step size of the output clock time base generator is adjusted to reproduce the ramp of the input clock, and therefore its content is directly given in the input clock domain. The timing relation for the case $F_{\text{c}} = 2F_{\text{out}}$ is depicted in Figure 3.16. The figure puts together the timing relation between sample times, clk_{out}, clk, clk_{in}, and the construction of the input clock time base and the output clock time base from the top graph to the bottom graph, respectively.

As internal PLL synthesizers are readily available in FPGA, any of the two methods can be applied independently on the relation between input and output sampling rates. Since the second circuit leads to lower hardware complexity by avoiding a division, the second method, namely resampling with output clock time base, is preferred.

3.4 Verification

Both resampler circuits were simulated in Matlab/Simulink software. The input signal $x[m]$ was generated with 4 samples per symbol using a random BPSK sequence and a raised cosine pulse shaping filter with a roll-off factor equal to 0.25.

The first simulation is set up as follows. A resampler with input clock time base is simulated using the Simulink software to resample the input signal from input rate

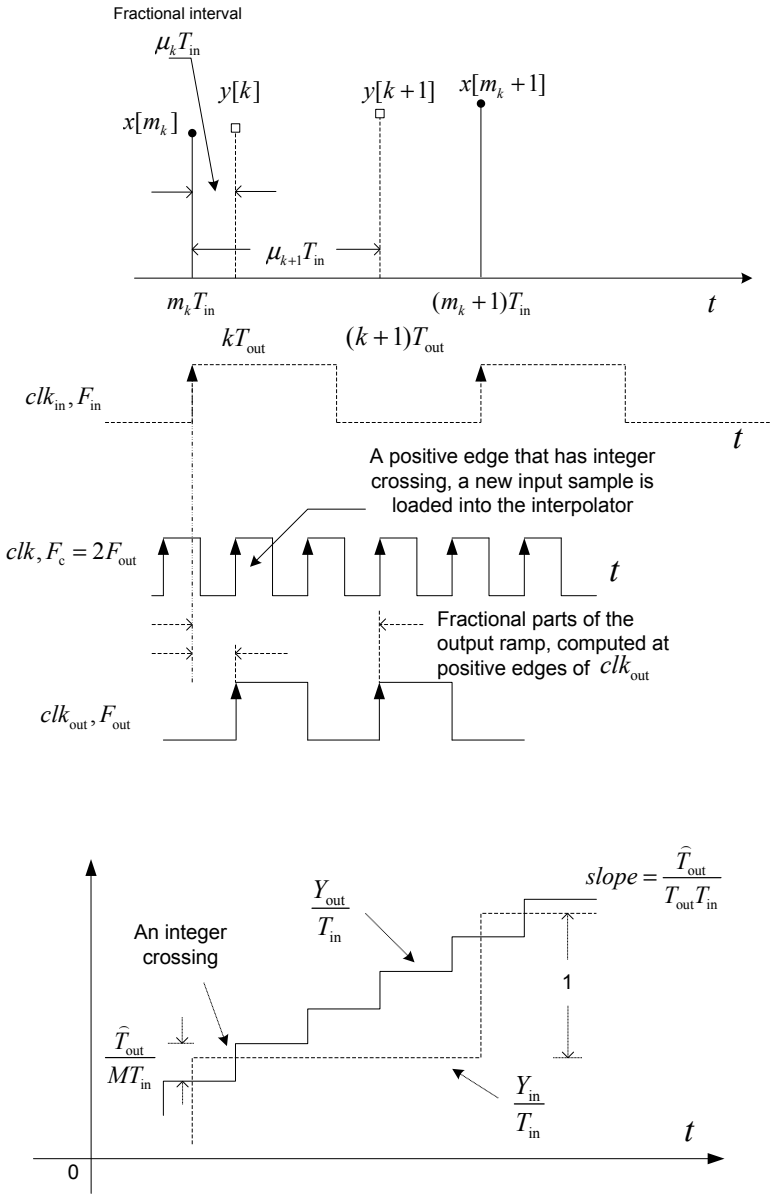

Figure 3.16 Timing relation in output clock time base resampling.

F_{in} to output rate $F_{\text{out}} = (4/3)F_{\text{in}}$. The circuit is clocked by system clock clk at rate $F_{\text{c}} = 4F_{\text{in}}$ (i.e., $M = 4$). The PLL (see Figure 3.13) was set up with small loop gains to reduce the effect of timing jitter, which is the random fluctuation in the timing [4].

Figure 3.17 shows the evolution of the step size, which converges to 1/3 and matches to the theoretical value, i.e., the input $\hat{T}_{\text{in}}/(MT_{\text{out}})$ of the accumulator in Figure 3.11. In steady state, the step size is fluctuated around 1/3 as shown in Figure 3.18.

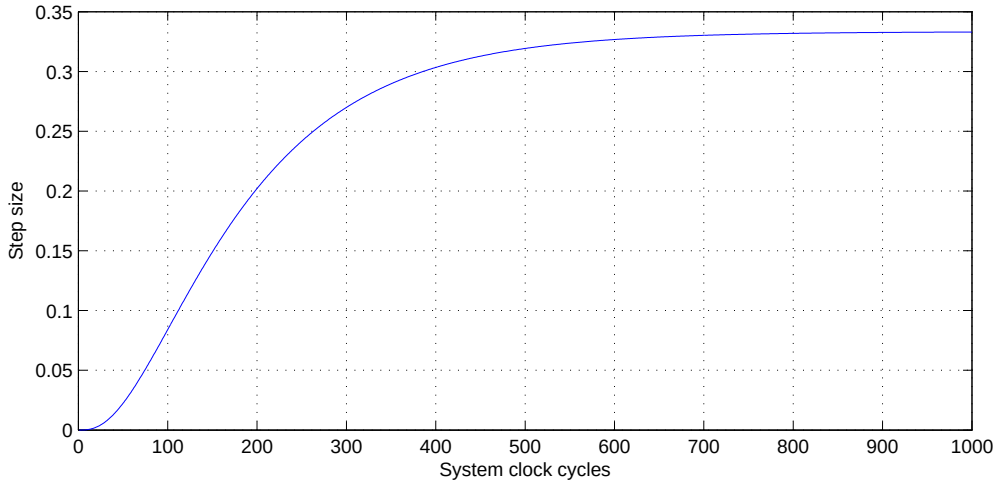

Figure 3.17 Evolution of the step size.

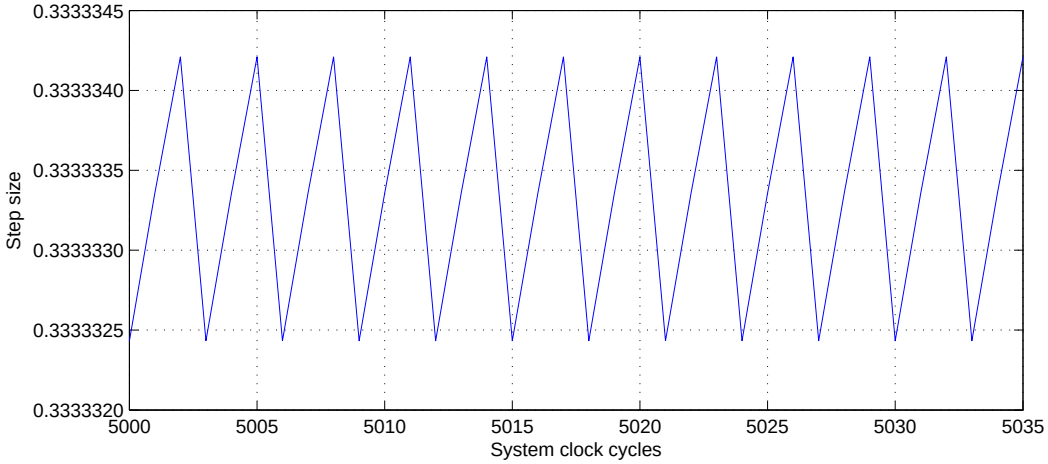

Figure 3.18 The step size in steady state.

Figure 3.19 shows the convergence of the input clock time base Y_{in} to the output clock time base Y_{out} (as the reference). Note that the output time T_{out} is normalized to 1 second. Figure 3.20 shows the two ramps which are synchronized after the step size has been converged.

Figure 3.21 shows sets of input samples (marked with a circle) and output interpolants (marked with an asterisk) after the circuit has found the correct step size and the two time bases have synchronized. The corresponding fractional intervals (marked with an asterisk) are shown in Figure 3.22. Cubic interpolator is used in this

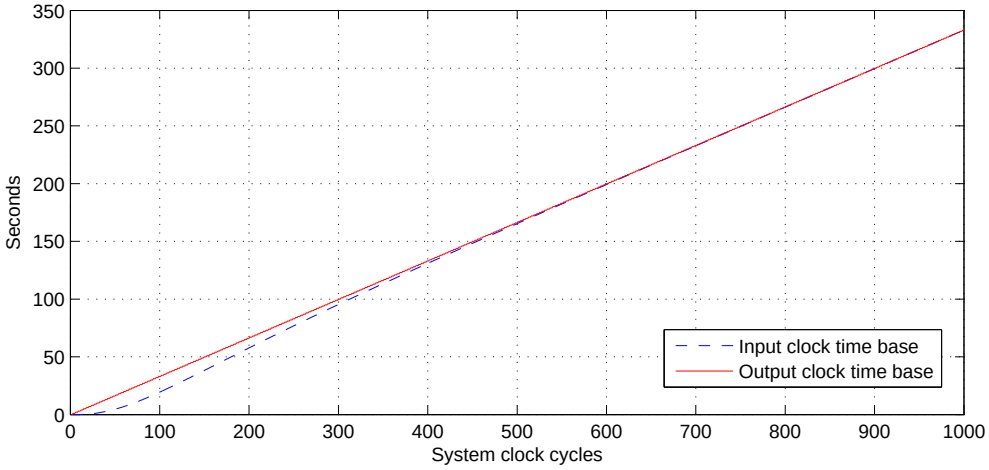

Figure 3.19 Variation of the input clock time base and the output clock time base.

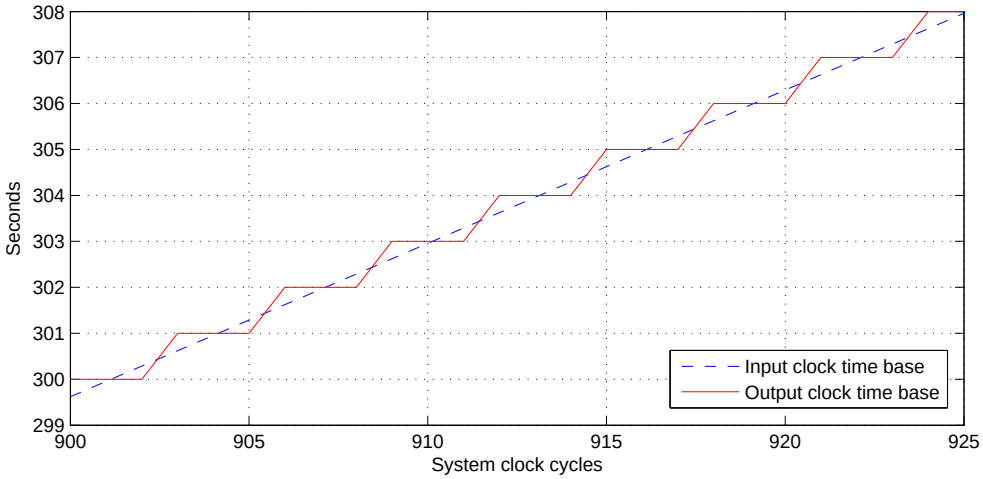

Figure 3.20 The input clock time base synchronizes with the output clock time base.

simulation.

The second simulation is set up to show the Power Spectral Densities (PSDs) of the resampled signals. The PSDs of the resampled signals were estimated using Welch's 50% overlapping method with blocks of length 2^{13} samples intervals [15]. Both resampler circuits were simulated with two different conversion rates, $10/3 \approx 3.33$ and $10/17 \approx 0.59$ of the input rate, where the input rate was $F_{\text{in}} = 4/T$ with $1/T$ being the

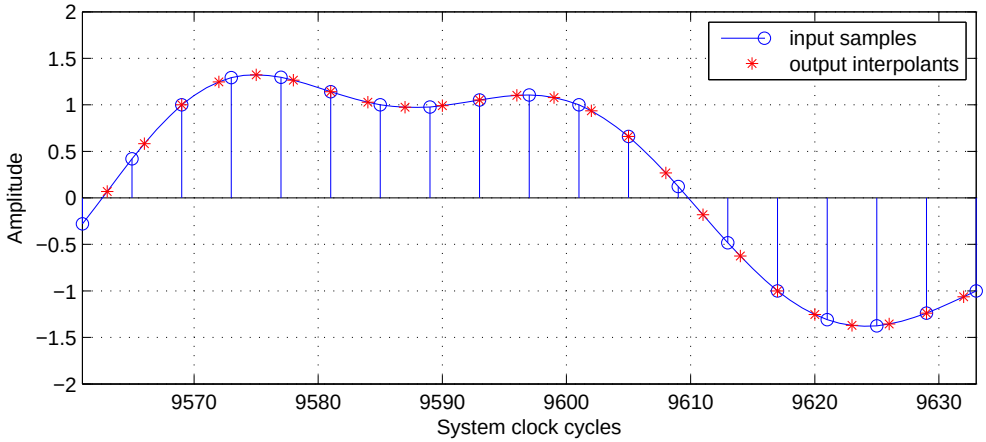

Figure 3.21 Input samples and output interpolants.

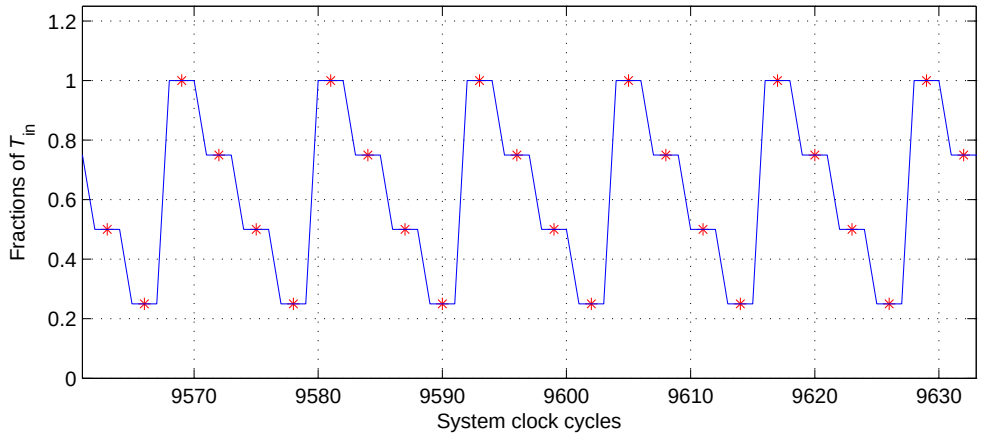

Figure 3.22 Fractional intervals.

symbol rate. The interpolator was a third order Farrow filter[1]. The PSD estimates obtained from the resampler with input clock time base are plotted in Figure 3.23 and Figure 3.24 (solid curve) and the PSD estimates obtained from the resampler with output clock time base are plotted in Figure 3.25 and Figure 3.26 (solid curve).

Theoretical curves (dashed curves) are also plotted in all the graphs of Figures 3.23, 3.24, 3.25, and 3.26. The equation for the theoretical curves is given in (3.13), which is the frequency response of the raised cosine pulse [16].

[1]Farrow filter is reviewed in Chapter 5 of this thesis.

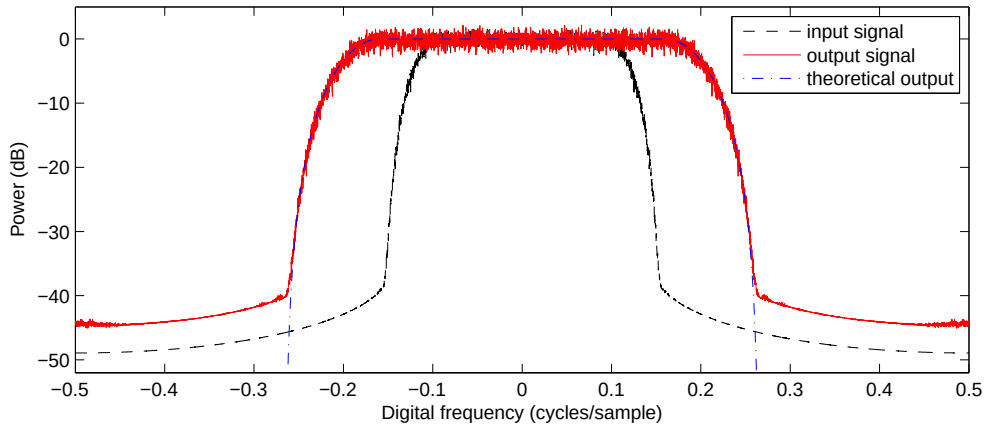

Figure 3.23 Resampler with input clock time base - PSDs for conversion rate ≈ 0.59.

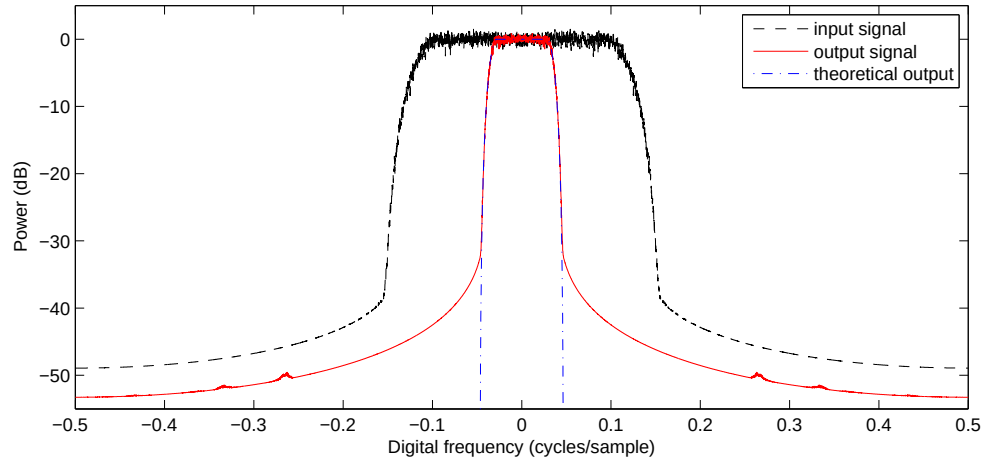

Figure 3.24 Resampler with input clock time base - PSDs for conversion rate ≈ 3.33.

$$H(f) = \begin{cases} 1, & |f| \leq \frac{1-\beta}{2T} \\ \frac{1}{2}\left[1 + \cos\left(\frac{\pi T}{\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right)\right], & \frac{1-\beta}{2T} \leq |f| \leq \frac{1+\beta}{2T} \\ 0, & \text{otherwise.} \end{cases} \tag{3.13}$$

In Equation (3.13), T is the symbol period and β is the roll-off factor, which is a real number between 0 and 1. The roll-off factor β determines the excess bandwidth of the filter. The excess bandwidth is the bandwidth occupied beyond the Nyquist bandwidth of $1/(2T)$.

The close agreement between the experimental and theoretical curves strongly suggests that the circuits work properly. From the plots, the performances of both

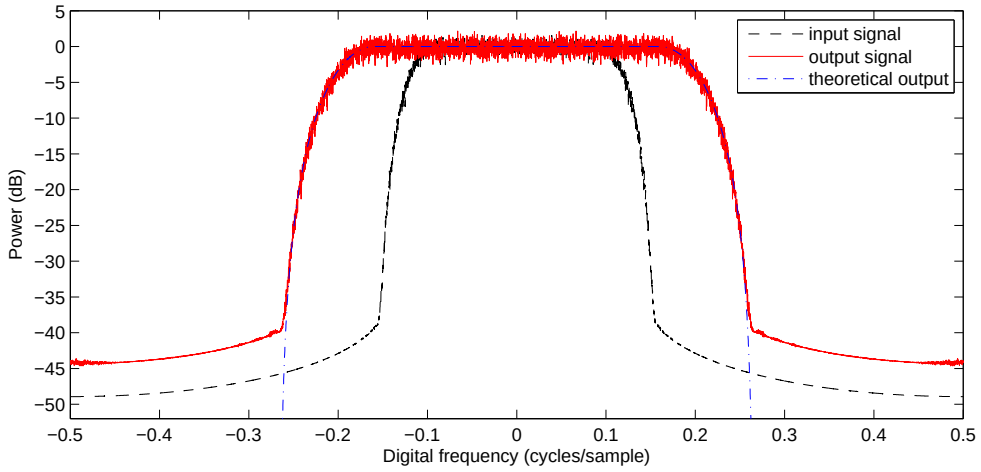

Figure 3.25 Resampler with output clock time base - PSDs for conversion rate $\approx$ 0.59.

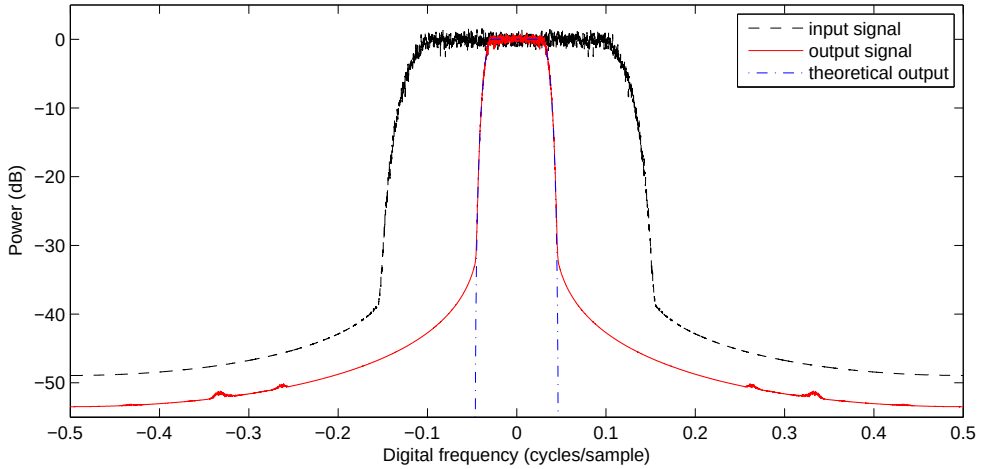

Figure 3.26 Resampler with output clock time base - PSDs for conversion rate $\approx$ 3.33.

circuits appear to be identical. However, the resampler using output clock time base requires no division for the computation of μ_k and therefore is the preferred one.

3.5 Summary

The first part of this chapter reviewed basic theory of digital interpolation. In the second part, the problem of resampling and the method to perform resampling in hardware have been discussed. The method to perform resampling is time base

generation and synchronization, which is very suitable for hardware implementation. The method is fundamental to devise practical circuits for asynchronous sampling rate conversion in the third part.

In the third part, two resampler circuits were described and simulated with MATLAB/Simulink software. Simulations show that both circuits offer similar performance but one of them, namely the resampler circuit which uses the output clock time base has a hardware implementation advantage over the other one. It does not require any division to compute the position of the new samples with respect to the incoming samples. This is a significant advantage in FPGA implementation.

The main contribution of this chapter is the detailed methodology and implementation of resampling operation in hardware. Practical circuits for digital resampling have been devised, verified and compared by simulation.

4. Digital Timing Recovery

4.1 Timing Recovery Circuits

The concepts and operations of timing recovery were reviewed in Chapter 2. In this chapter we devise timing recovery circuits with two different approaches. Both approaches share the basic ideas of time base generation and synchronization described in Chapter 3. The major difference between sampling rate conversion circuits and timing recovery circuits is that, while the sampling rate conversion circuits use a PLL to synchronize the input clock time base and output clock time base, the timing recovery circuits use a TED to estimate the timing offset between the current sampling times and the correct sampling times. More explicitly, timing recovery circuits estimate the value of current timing error to build up the step size of the reconstructed clock time base. The Gardner TED is adopted in this research since it is very efficient for QAM data [4].

4.1.1 Timing Recovery with Input Clock Time Base

The first approach, referred to as timing recovery with input clock time base, arises naturally from the timing problem described in Chapter 2. The transmitter sampling clock is clk_{tx} with the sampling rate F_{tx} and the receiver sampling clock is clk_{in} with rate F_{in}. In the receiver, the timing recovery circuit is used to recover the correct sampling rate F_{tx} (i.e., remove the sampling frequency offset) and the correct sampling times (i.e., remove the timing offset). The circuit works in the clk_{in} time domain, i.e., it is operated with system clock clk with rate $F_c = MF_{\text{in}}$, where M is an integer number. In the receiver, from the received samples $x[m]$ obtained

with rate F_{in}, the timing recovery circuit generates output interpolants $y[k]$ with the output enable signal k. The objective is to make k active with an average rate equal to F_{tx} (i.e., to recover the correct sampling frequency). The timing recovery circuit also needs to remove the timing offset (i.e., it recovers the correct sampling times).

The time base generation and synchronization are set up as follows. First, T_{tx} is normalized to 1 to generate the transmitter clock time base as a reference (see Figure 4.1). The quantity $Y_{\text{tx}}/T_{\text{tx}}$ is time t scaled by $1/T_{\text{tx}}$ and then quantized with step size 1.

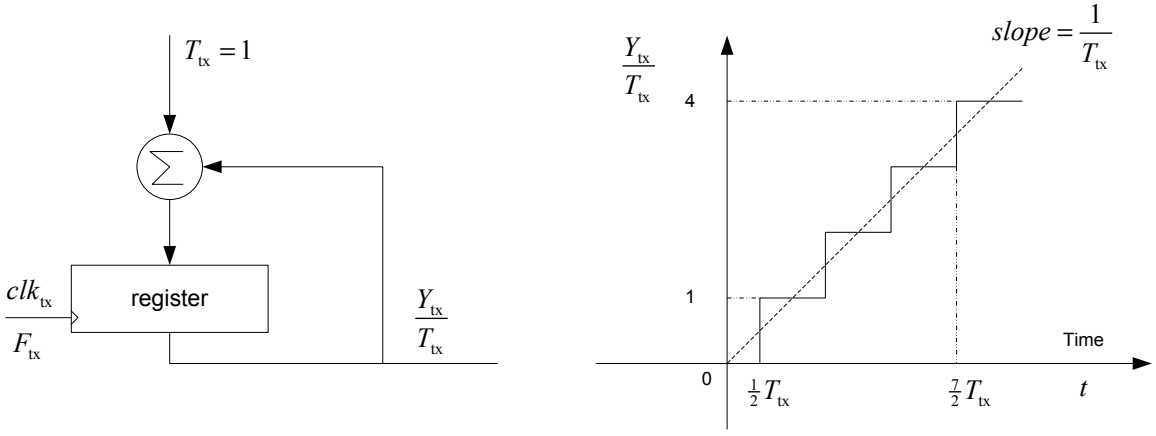

Figure 4.1 Transmitter clock time base.

The timing recovery circuit, in the process of estimating F_{tx}, will construct an input (receiver) clock time base that also generates quantized values of t/T_{tx}. In other words, the circuit constructs an input clock time base that has the same slope as the transmitter clock time base. As in Chapter 3 this is called the time base synchronization process. Note that in this scenario, receiver clock time base and input clock time base are the same. Since the circuit works in input time domain T_{in}, synchronization happens if the step size of the input clock time base is $T_{\text{in}}/T_{\text{tx}}$. In hardware, the step size is estimated by $\hat{T}_{\text{in}}/T_{\text{tx}}$.

Since the circuit is operated with clock rate $F_{\text{c}} = MF_{\text{in}}$, $Y_{\text{in}}/T_{\text{tx}}$ is $(t - \Delta Y_M)/T_{\text{tx}}$ quantized with step size $\hat{T}_{\text{in}}/(MT_{\text{tx}})$, where ΔY_M is the timing offset between clk and clk_{tx}. The construction of $Y_{\text{in}}/T_{\text{tx}}$ by system clock clk is depicted in Figure 4.2.

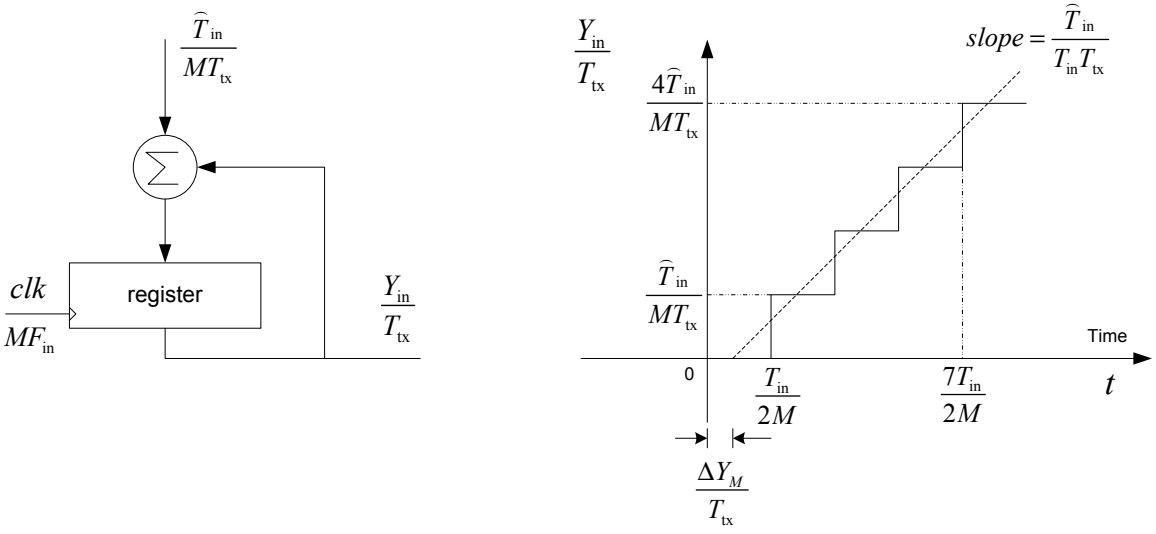

Figure 4.2 Receiver clock time base.

A complete block diagram of the timing recovery circuit with input clock time base is described in Figure 4.3. The timing recovery circuit has two inputs, $x[m]$ and clk_{in}. The output of the circuit is the symbol sequence at rate F_{symbol}. The output of the interpolator, $y[k]$, represents QAM interpolants. In the QAM systems that uses 4 samples per symbol, $F_{\text{tx}} = 4F_{\text{symbol}}$ and one QAM symbol is detected from every 4 consecutive interpolants $y[k]$. In Figure 4.3, the clock at frequency $F_{\text{c}} = MF_{\text{in}}$ is generated from a built-in PLL in the FPGA device where M is an integer number. Normally in the context of timing recovery, F_{tx} is very close to F_{in} so $M = 2$ is large enough for the circuit to operate. More specifically, F_{c} is the fastest rate the circuit can generate output enable k so M is chosen to make $F_{\text{c}} \geq F_{\text{tx}}$. The block labeled "Input clock time base" is the circuit in Figure 4.2, which is clocked by clk at frequency $F_{\text{c}} = MF_{\text{in}}$.

In a QAM receiver, the Gardner TED [1] uses two samples per QAM symbol to estimate the timing error between the current sampling time and the correct sampling time (transmitter sampling time). In a system that uses 4 samples per QAM symbol, one sample in every two consecutive interpolants $y[k]$ is used. The hardware structure to implement Gardner TED in Equation (2.19) is illustrated in Figure 4.4 for a system that use 4 samples per QAM symbol [4]. The timing error, i.e., output of the Gardner

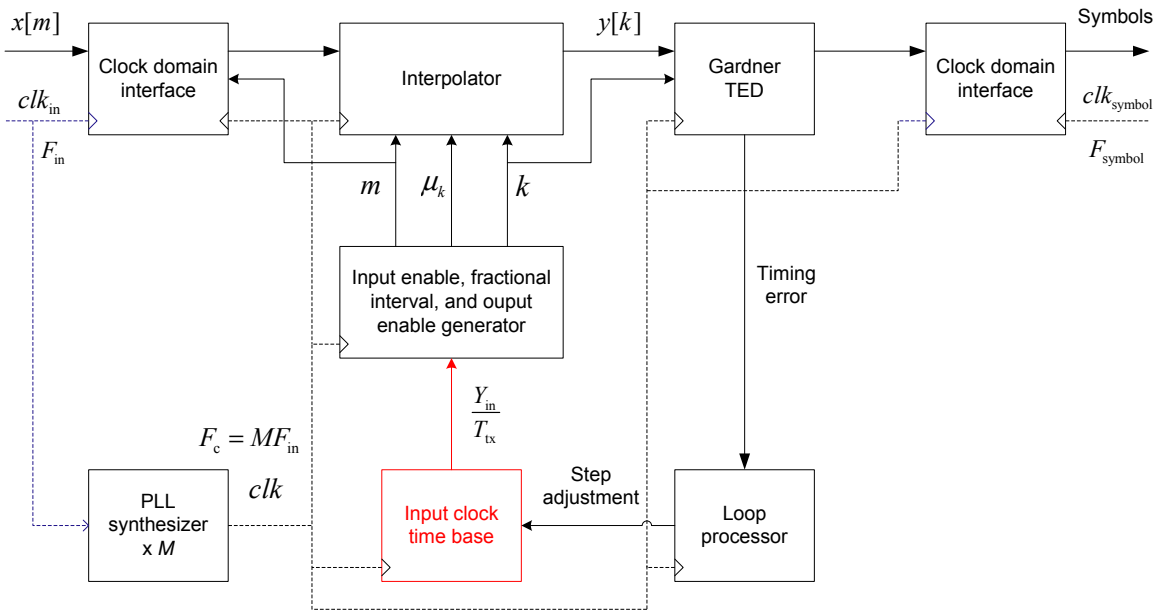

Figure 4.3 Timing recovery circuit with input clock time base.

TED, is then used to adjust the step size of the estimated transmitter time base. This is the major difference from the resampler with input clock time base, where the step size was adjusted from the difference between the input clock time base and the output clock time base.

The circuit to update the step size from timing error is illustrated in Figure 4.5. It is set up in a similar way as in Figure 3.13 for the PLL of the resampler with input clock domain. Accumulator 1 holds the value of input clock time base, which is the block "Input clock time base" in Figure 4.3. All other parts of Figure 4.5 make up the block "Loop processor" in Figure 4.3. Accumulator 2 is needed to hold the step size. The acquisition process of Accumulator 2 works as follows. If the current timing estimation is too early from the correct timing, which indicates that the timing should be advanced, the Gardner TED equation produces a positive and larger timing error. This makes the step size larger, which causes the input clock time base $Y_{\text{in}}/T_{\text{tx}}$ to become larger and keeps track with the transmitter clock time base. The error is then smaller at the next pass through the loop. Conversely, if the current timing is too late from the correct timing, then the step size becomes smaller to make the timing earlier.

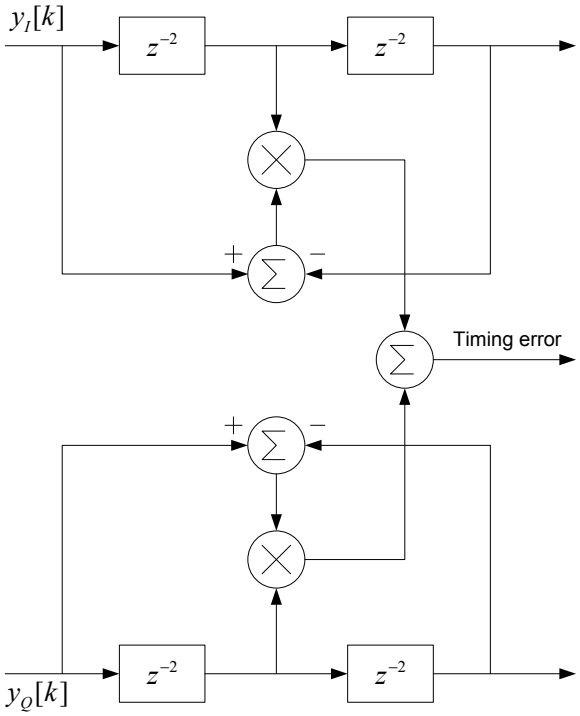

Figure 4.4 Hardware structure of Gardner TED.

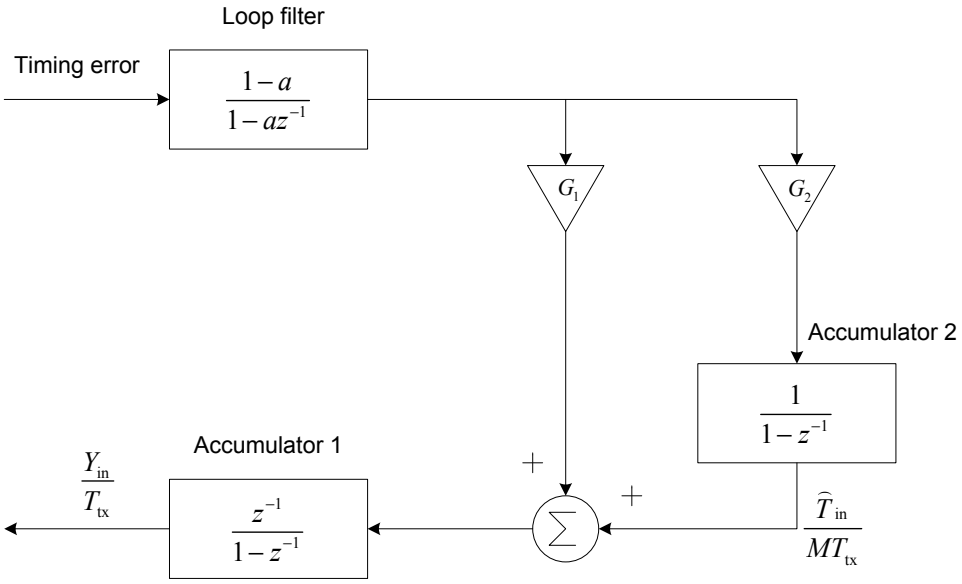

Figure 4.5 Updating the step size and constructing the input clock time base.

Since the output timing error of the Gardner TED is noisy, it is filtered by a loop filter. Only a small portion of the error is then used to update the step size $\hat{T}_{\text{in}}/(MT_{\text{tx}})$ (Accumulator 2). A bigger portion is used to update the input clock time base directly. Therefore, in the design, gain G_2 is much smaller than gain G_1. After

the loop has found the correct timing, the error is close to 0 and the loop operates in the steady state. In the steady state, Accumulator 2 contains the step size value which is approximately equal to $T_{\text{in}}/(MT_{\text{tx}})$.

The time base $Y_{\text{in}}/T_{\text{tx}}$ is then used to generate m, μ_k, and k to be used in the "Input enable, fractional interval, and output enable generator" block. The computation of μ_k and generation of m and k are similar to that in the resampler circuit with the input clock time base. The fractional interval μ_k is calculated as in Equation (3.11) and the timing relationship is depicted in Figure 3.14. The "Input enable, fractional interval, and output enable generator" block does not require the value of transmitter time base (as a reference) to detect the integer crossing, so the timing recovery circuit works without actually generating the transmitter time base.

As an example, if $M = 2$, $T_{\text{in}} = 0.8T_{\text{tx}}$, then the step size is equal to 0.4 in the steady state. In this example, F_{in} is faster than F_{tx}, the circuit is operated by *clk* at rate $F_{\text{c}} = 2F_{\text{in}}$. Since the step size equals to 0.4, the output enable k is active every $1/0.4 = 2.5$ *clk* cycles on the average. The average active rate of k is then equal to $F_{\text{c}}/2.5 = F_{\text{tx}}$. Thus the circuit recovered the correct sampling rate F_{tx}. The circuit also found the correct sampling times since the Garder TED timing error is 0 in the steady state.

As discussed, there is a division when computing μ_k, this may introduce additional errors and hardware complexities to a practical receiver.

4.1.2 Timing Recovery with Output Clock Time Base

The timing recovery circuit with output clock time base is operated by the system clock *clk* with the rate $F_{\text{c}} = MF_{\text{out}}$, where F_{out} is an integer multiple of the symbol rate. For a system that uses 4 samples per QAM symbol, $F_{\text{out}} = 4F_{\text{symbol}}$. The circuit still uses clk_{in} at rate F_{in} as the receiver sampling clock. This approach assumes that the receiver knows the symbol rate F_{symbol} and the receiver's front-end sampling rate (i.e., F_{in}) can be different from an integer multiple of F_{symbol}. In this scenario, $F_{\text{out}} = F_{\text{tx}}$ and F_{out} shall be used instead of F_{tx} in this subsection. This approach

could allow a flexibility in choosing the receiver's front-end sampling rate and make the processing of QAM symbols after timing recovery easier.

The setup of time bases is as follows. The input (receiver) clock time base is set up by normalizing the step size as illustrated in Figure 4.6. The output (transmitter) clock time base is now constructed from the output clock as in Figure 4.7. Again, the time base synchronization is obtained by processing the timing error of Gardner TED. In this approach, the TED estimates the timing error to update the step size

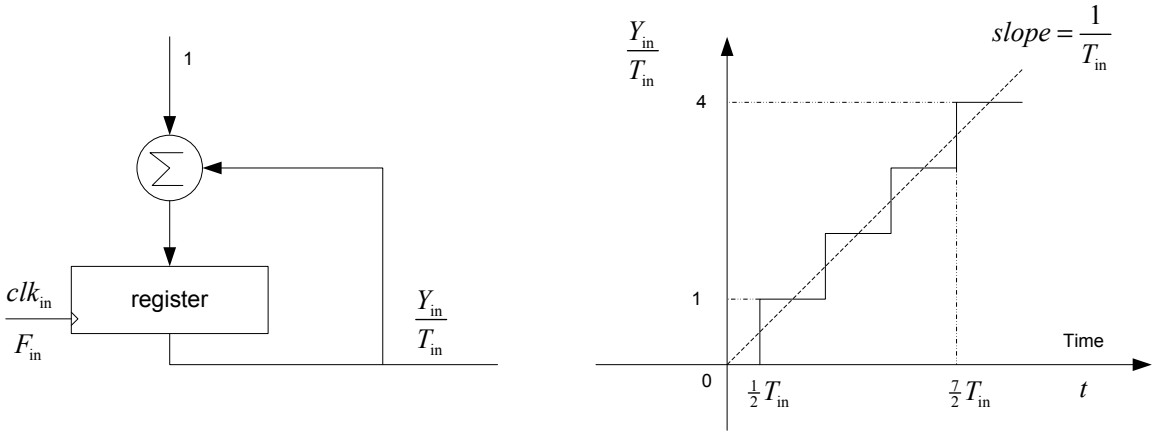

Figure 4.6 Receiver clock time base.

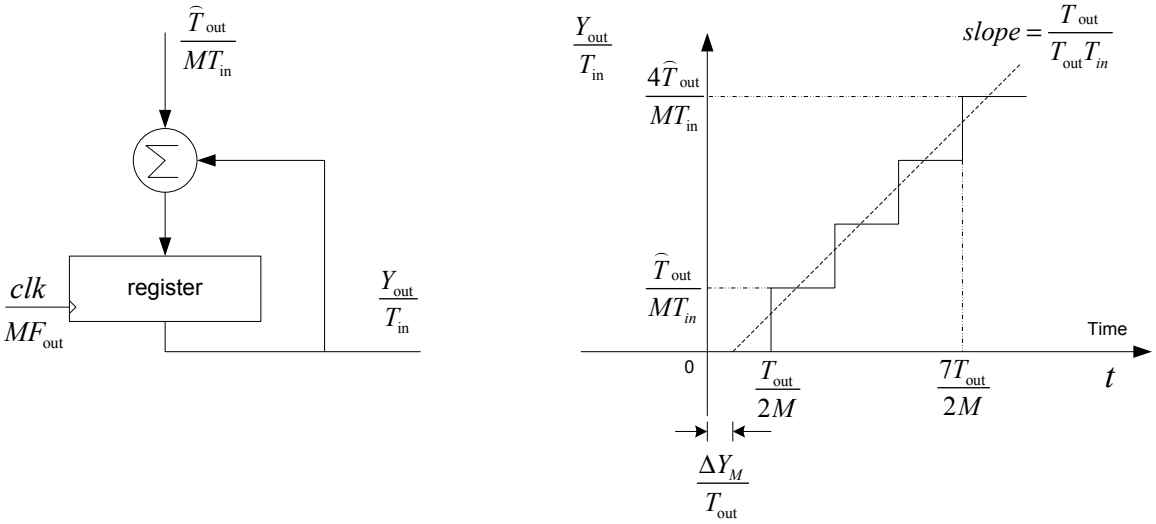

Figure 4.7 Transmitter clock time base.

$\hat{T}_{\text{out}}/(MT_{\text{in}})$ of the output clock ramp $\hat{Y}_{\text{out}}/T_{\text{in}}$. The complete timing recovery circuit with output clock time base is depicted in Figure 4.8.

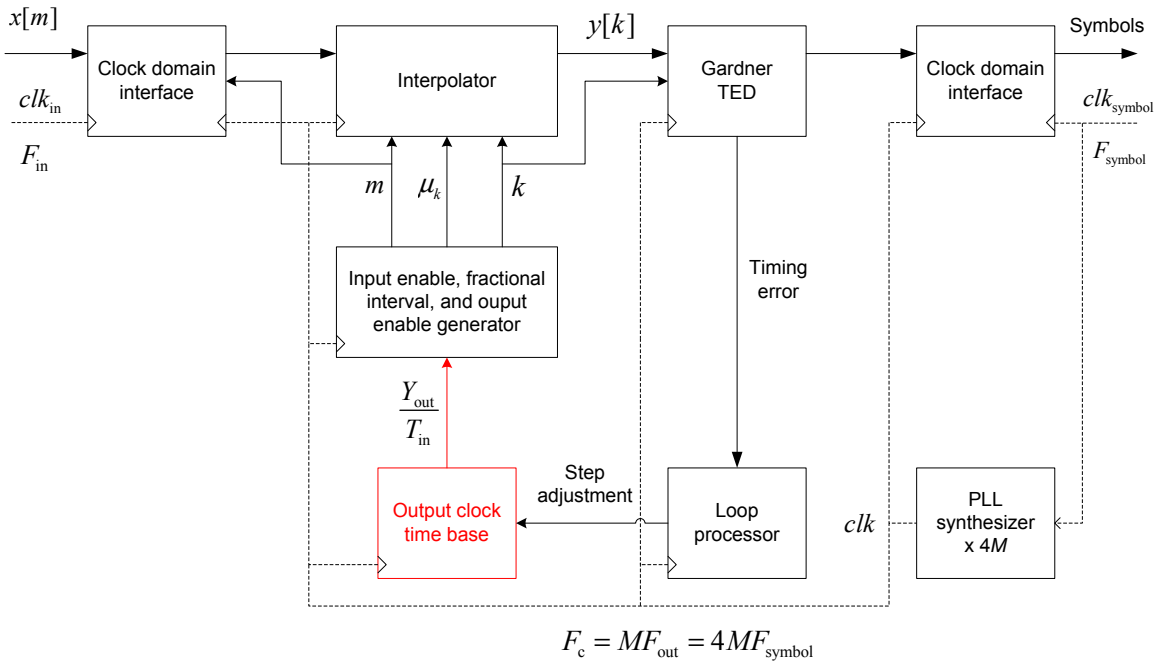

Figure 4.8 Timing recovery circuit with output clock time base.

The system clock clk with rate $F_{\rm c} = MF_{\rm out}$ (M is an integer number) is generated from the PLL synthesizer inside the FPGA device . Normally, $F_{\rm out}$ is very close to $F_{\rm in}$ so $M = 2$ is large enough for the circuit to read input samples $x[m]$ with rate $F_{\rm in}$. Technically, $F_{\rm c}$ determines the fastest rate the circuit can read the input signal, so M should be large enough to make $F_{\rm c} \geq F_{\rm in}$. In the steady state, the reading and writing rates at the clock domain interfaces are the same. The operations of this timing recovery circuit are similar to the timing recovery circuit with input clock time base. The computation of μ_k and the generation of m and k are exactly the same as in the case of resampler with output clock time base. The computation of μ_k does not require a division so this is the reason to make the circuit a preferred choice in many practical applications.

4.2 Performance Analysis

The performance of a timing recovery circuit depends on many factors. One of the most significant factors is the interpolation error. This raises an interesting question to system designers: Which type of interpolator is good enough and suitable for

a certain application? Another factor that can affect the performance of a timing recovery circuit is self noise caused by a repetition of the symbols with the same polarity. Self noise degrades the performance of Gardner TED and some techniques have been proposed to mitigate self noise. In this research, we focus on interpolation error and try to isolate other factors that might affect the performance of the circuits. The effect of self noise is minimized by choosing small loop gains (refer to Figure 4.4). Small loop gains also help to reduce the effect of timing jitter [4].

Model

To analyze the performance of an interpolator in timing recovery, the method is to use a resampler to change the sampling rate of the original QAM sequence from sampling rate F_{tx} to F_{in}. Then use a timing recovery circuit to recover QAM sequence back to sampling rate F_{tx}. The model is described in Figure 4.9 where $a[i]$, $b[j]$, and $a'[i]$ are QAM sample sequences. F_{tx} is the input sampling rate, which is normally 4 times the symbol rate. For the purpose of demonstration, when using the same symbol index i we ignore clock delays between $a[i]$ and $a'[i]$ introduced by the whole circuit. The performance of the interpolator under test in the timing recovery circuit is then obtained from input QAM symbols (the input sample sequence $a[i]$) and output QAM symbols (detected from output sample sequence $a'[i]$).

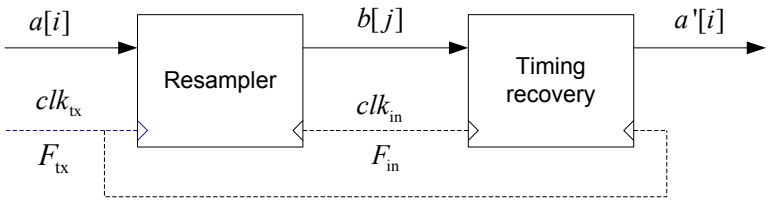

Figure 4.9 An approach to timing recovery performance analysis.

To minimize the error caused by the resampler, QAM symbols are upsampled by 16 before passing through the interpolator. To minimize the effect of timing jitter, a fixed step size is used in the resampler instead of an adjustable step size. In this way we can eliminate the need of time base synchronizer in the resampler. Recall that the step size of the input clock time base in the resampler determines the output

sampling rate of the resampler, denoted by F_{in} in this case as it is the input rate of the timing recovery part. The timing recovery part does not know F_{in} and its task is to recover the sampling rate F_{tx}. This design maintains the key requirement of the timing recovery circuit, namely recovering the symbol time. More explicitly, timing recovery circuit removes the sampling frequency offset between F_{in} and F_{tx}, and removes the timing offset in its input samples.

To minimize the effect of clock jitter and the error that PLL synthesizer in FPGA devices could introduce when generating incommensurate sampling clocks, in Figure 4.9 the resampler uses the input clock time base and the timing recovery uses the output clock time base. The design objective of the performance analysis circuit is to maximize the combination of random QAM data and fractional intervals. First, the input QAM symbols $a[i]$ is generated from a random bit sequence. Second, since the step size of the resampler determines the ratio between F_{tx} and F_{in}, the value of that step size is set to make fractional intervals vary in the range $[0, 1]$. The design of the analysis circuit is presented in the next subsection.

4.2.1 Performance Analysis Circuit

The proposed circuit to analyze the performance of different interpolators in timing recovery is illustrated in Figure 4.10.

The circuit includes the following modules. PLL synthesizer is a PLL inside the FPGA device to generate clock signal *clk* with rate F_{c}. In the design $F_{\text{c}} = F_{\text{tx}} = 4F_{\text{out}} = 16F_{\text{symbol}}$. QAM generator module generates a symbol stream pseudo-randomly using LFSR (Linear Feedback Shift Register) before passing through a RC filter and an up-sampler by 16 to generate QAM sample sequence $x_1[m_1]$ at rate F_{tx}. $x_1[m_1]$ is then the input sample sequence with 16 samples per QAM symbol, and QAM symbols are at right timing.

The resampler implements a cubic interpolator and generates an input clock time base at a fixed step size. In the design, the step size is set to 0.99 to make the output sampling rate $F_{\text{out1}} = 0.99 \times F_{\text{tx}}$. This simulates the effect of sampling frequency

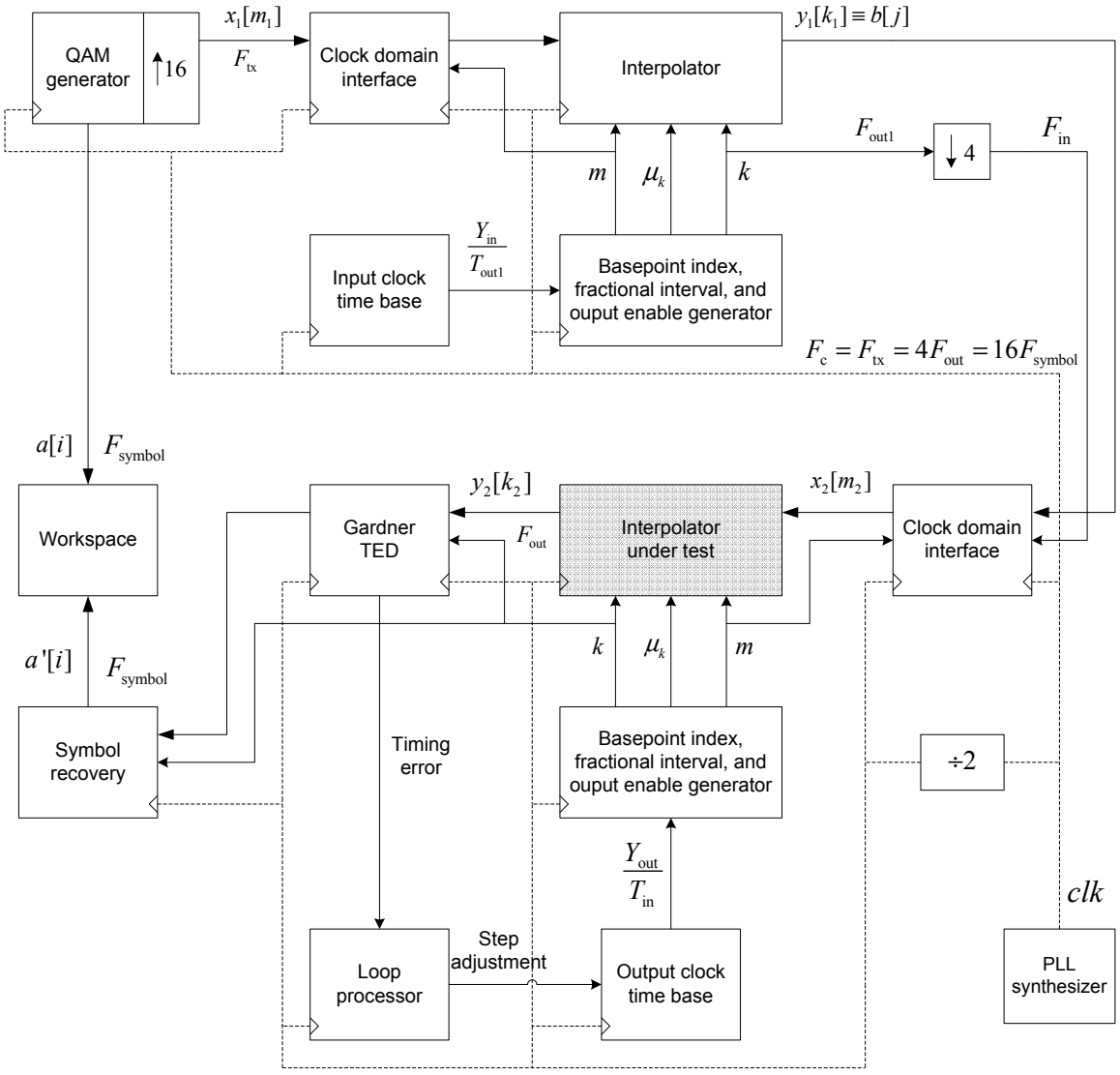

Figure 4.10 Performance analysis circuit for interpolators in timing recovery.

offset and timing offset experienced in the sequence $y_1[k_1]$ or $b[j]$. The sequence $y_1[k_1]$ is then down-sampled by 4 before passing to the timing recovery circuit. The down-sampling by 4 of $y_1[k_1]$ is implemented by simply down-sampling by 4 the output enable k in the resampler.

Sequence $y_1[k_1]$ is written to the clock domain interface with rate F_{out1}. In the steady state (i.e., the step size is fluctuated around a stable value), timing recovery circuit reads from the clock domain interface with the same rate, F_{out1}. Recall that the timing recover circuit with output clock time base generates m at the same rate

as the input samples of the circuit (in this case F_{out1}). The interpolator under test could be a linear interpolator, cubic interpolator or parabolic interpolator.

The workspace analyzes performance of the timing recovery circuit. The module to compute MER from the input QAM symbol sequence and the output QAM symbol sequence is operated at symbol rate F_{symbol}. The workspace uses MATLAB software to do the computation for both Simulink simulation and FPGA implementation.

A summary of the main operations is as follows. A sequence of QAM samples (with 16 samples per symbol) is generated from the QAM generator and upsampled by 16. The resampler with input clock time base operates at rate $F_{\text{c}} = 16F_{\text{symbol}}$ and changes the input QAM samples from rate $F_{\text{tx}} = 16F_{\text{symbol}}$ to $F_{\text{out1}} = 0.99F_{\text{tx}}$. The down sampler by 4 reduces the rate from F_{out1} to $F_{\text{in}} = 0.25F_{\text{out1}}$. The timing recovery circuit with output clock time base operates at clock rate $F_{\text{c}}/4$ and recovers the right sampling rate, i.e., $0.25F_{\text{tx}}$. Note that in the design, the resampler is clocked by $F_{\text{c}} = 16F_{\text{symbol}}$ and the timing recovery circuit is clocked with rate $8F_{\text{symbol}}$. In this way, error from using incommensurate clocks is avoided.

4.2.2 Results and Evaluation

The performance analysis circuit is simulated using MATLAB/Simulink software. To accurately measure the performance of interpolation filters, simulation is set up without the presence of noise. Also, a square 2^{λ}-QAM constellation can be simplified by considering only its inphase component (or its quadrature component). This is equivalent to a $2^{\frac{\lambda}{2}}$-PAM modulation. Figure 4.11 shows a 2-PAM signal as the input of the performance analysis circuit. The filter is a RC with roll-off factor $\beta = 0.25$. The filter's sampling frequency equals 16 times the input symbol rate and the filter's group delay equals to 10 input symbol periods. The delay caused by the RC filter is not shown in the figure for the purpose of illustration.

As discussed in Chapter 1, MER is chosen as performance indicator for an interpolator in timing recovery. In timing recovery, the input sampling frequency is often very close to the transmitter sampling frequency. This implies that the step size of the

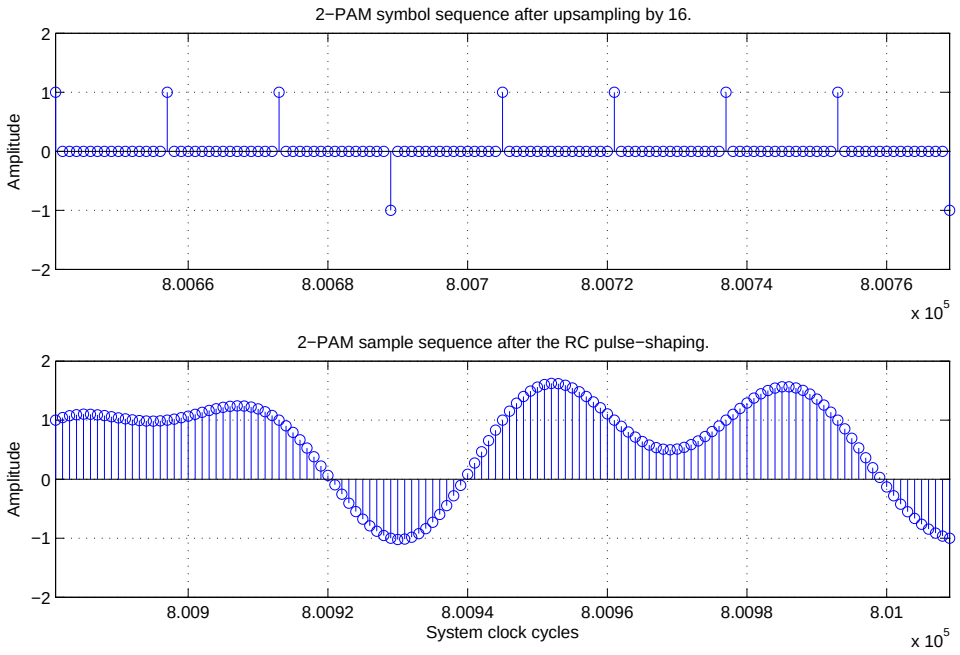

Figure 4.11 2-PAM sample sequences before and after shaping filter.

resampler can be initialized to 0.5 at the startup of the simulation. Recall that the timing recovery part of the performance analysis circuit is clocked at twice the rate of the output interpolants (i.e., $M = 2$). During the simulation, the loop processor should be able to adjust the step size to converge to 0.495 as $F_{\text{out1}} = 0.99 \times F_{\text{tx}}$. The evolution of the step size in the timing recovery part is shown in Figure 4.12 for the case of 2-PAM input signal.

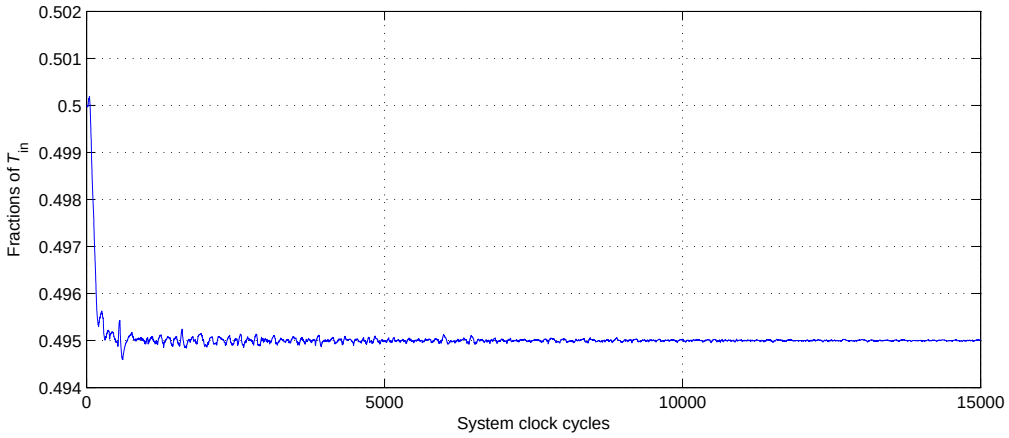

Figure 4.12 Evolution of the step size in timing recovery.

After the loop has found the correct step size and has removed the timing offset, it operates in the steady state. The input sample sequence and the output sample sequence of the interpolator under test are illustrated in Figure 4.13. The figure shows that the circuit has found the correct symbol sequence (see also Figure 4.11).

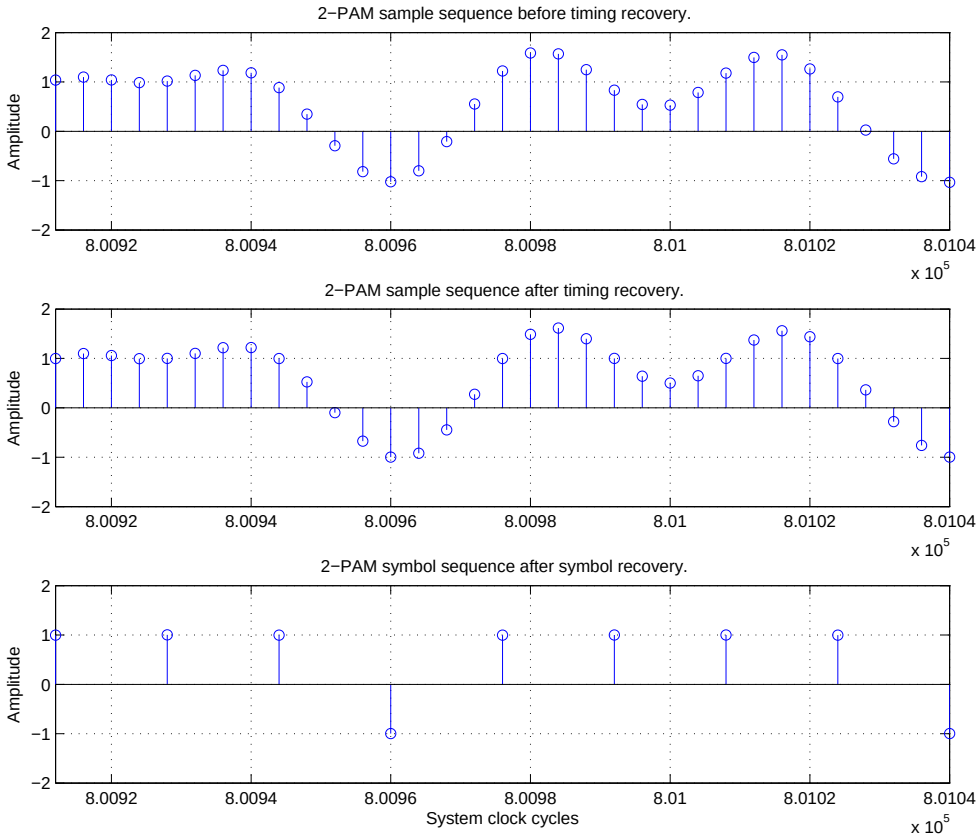

Figure 4.13 Example of 2-PAM signal in the timing recovery part.

MER measurement of the circuit is evaluated from the output symbol sequence and the input symbol sequence. Three of the most common types of polynomial interpolation filters to be evaluated are linear interpolator, parabolic interpolator, and cubic interpolator. Figure 4.14 shows how different filters interpolate between a set of 2-PAM input samples after RC pulse-shaping with 4 samples per symbol.

Pulse-shaping filters are also taken into consideration as they affect the PAM signal as the input of the interpolator under test. The pulse-shaping filter is chosen

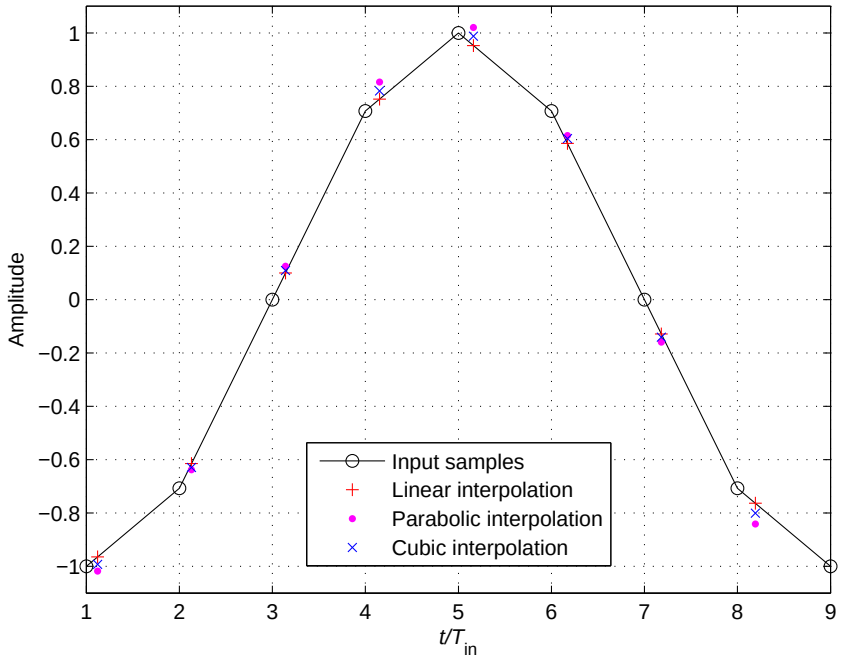

Figure 4.14 Polynomial interpolation.

to be an RC filter with different roll-off factors and different filter's group delays. Figure 4.15 shows the MER performance of a cubic interpolator for 2-PAM signals.

Observe from Figure 4.15 that in the high group delay range, the smaller the roll-off factor is, the better the MER performance becomes. A small roll-off factor means that the bandwidth of the PAM signal at the input of the cubic interpolator is small (see Equation (2.18)) and the interpolation filter causes less distortion to the signal (see also Section A.4). Less distortion to the signal increases the MER performance. This is not true for the small group delay range, when a bigger roll-off factor leads to a better performance. The reason is that truncation has more effect to the filter for a small roll-off factor than that of a bigger roll-off factor as illustrated in Figure 4.16.

The simulation results show that the cubic interpolator offers good MER performance for 2-PAM signals. In a practical system where hardware resource is an important consideration, a RC filter with roll-off factor $\beta = [0.2, 0.3]$ can provide a MER of about 52 dB without the need of having a long pulse-shaping filter's length.

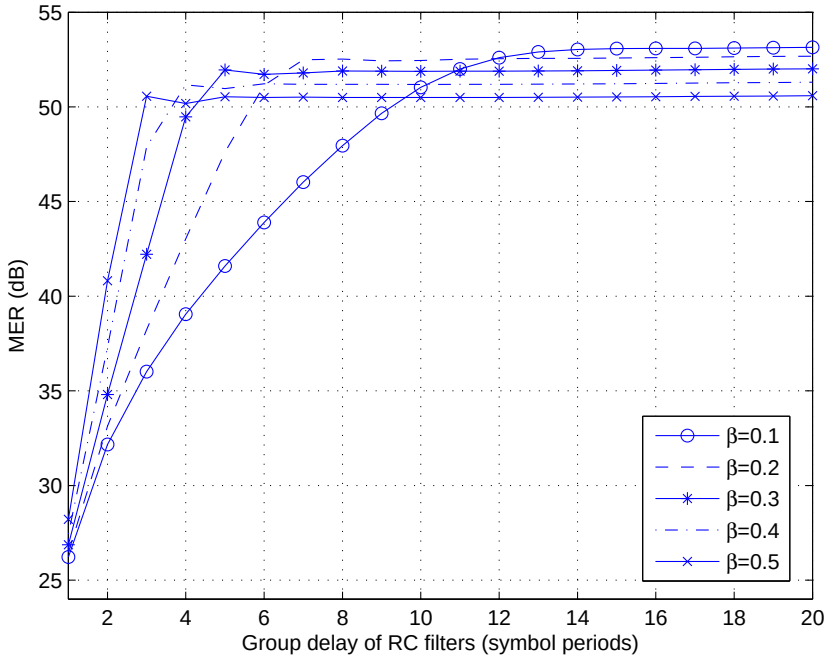

Figure 4.15 MER performance with the cubic interpolation filter for 2-PAM signals.

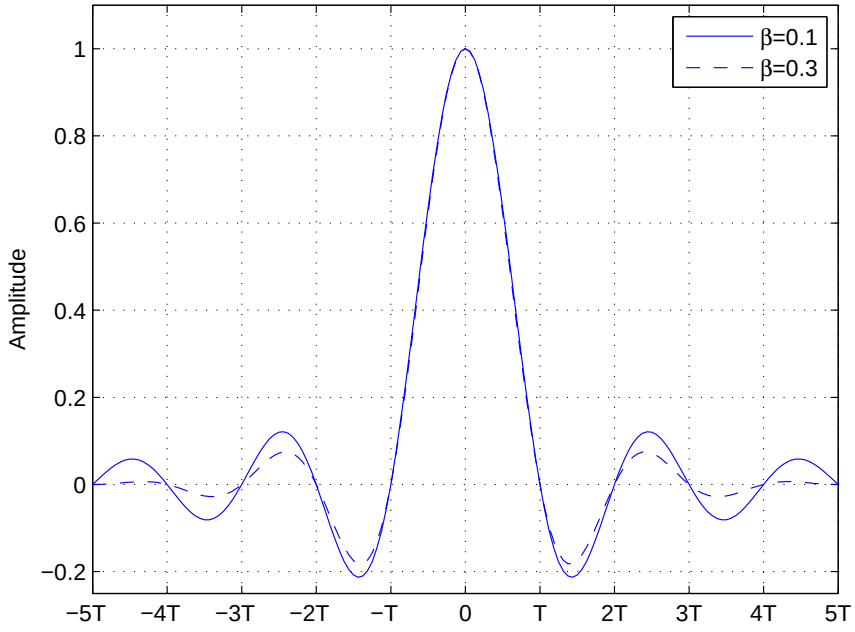

Figure 4.16 Raised cosine filters with different roll-off factors.

Figure 4.17 shows the MER performance with the linear interpolator. As can be seen from Figures 4.15 and 4.17, the cubic interpolator outperforms the linear interpolator in MER performance. The difference is approximately 20 dB. The MER performance with the parabolic interpolator is shown in Figure 4.18. It is interesting to note that in the large roll-off factor range, an RC filter with a short filter length offers a better performance than the one with a longer filter length. This can be explained by the fact that the parabolic interpolator adds some amplitude gain to the signal (see Appendix A).

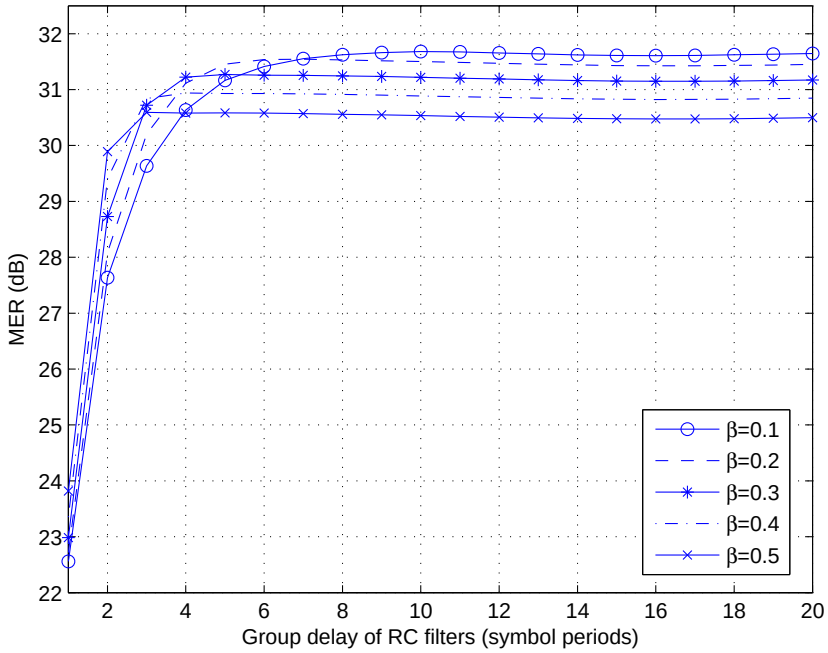

Figure 4.17 MER performance with the linear interpolation filter for 2-PAM signals.

In general, the cubic interpolation filter offers a significant MER performance advantage over the parabolic and linear interpolators. The linear interpolation filter, not surprisingly, has the worst performance. Also, there is not much difference in performance between the linear interpolator and the parabolic interpolator, even the hardware structure of the linear interpolator is much more simpler than that of the parabolic interpolator. Finally, results show that the MER performance of an interpolator does not depend at all on the order of PAM modulation. This is illustrated

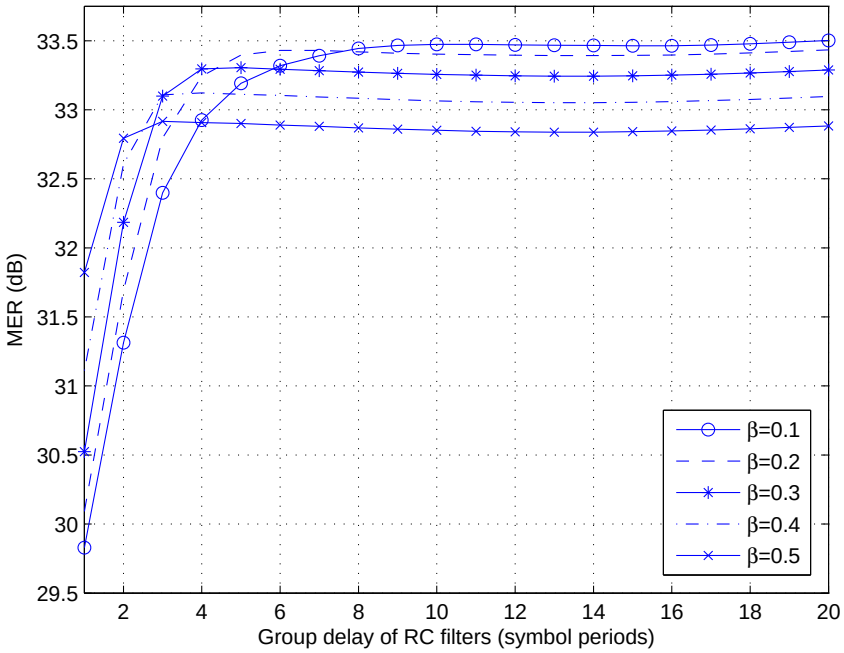

Figure 4.18 MER performance with the parabolic interpolation filter for 2-PAM signals.

in Figures 4.19 and 4.20 for a cubic interpolator with 4-PAM and 8-PAM signals, respectively.

4.3 Summary

In the first part of this chapter, two timing recovery circuits were described. The circuits were devised based on the idea of time base generation and synchronization in Chapter 3. The operations of the two timing recovery circuits are similar to that of the two resampler circuits discussed in Chapter 3. The timing recovery circuit with the output clock time base does not require any division to compute the position of the new samples. This is a significant advantage in FPGA implementation.

The second part of this chapter is devoted to the performance analysis of polynomial interpolation filters in timing recovery context. A model was proposed and a performance analysis circuit was devised. The circuit was simulated in MATLAB/Simulink software. Three types of interpolation filter, namely cubic interpo-

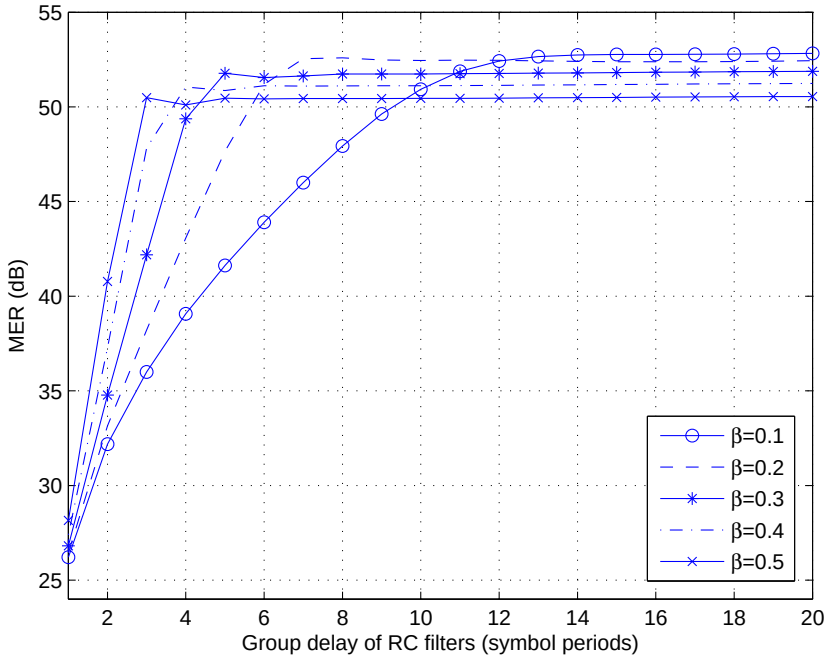

Figure 4.19 MER performance with the cubic interpolation filter for 4-PAM signals.

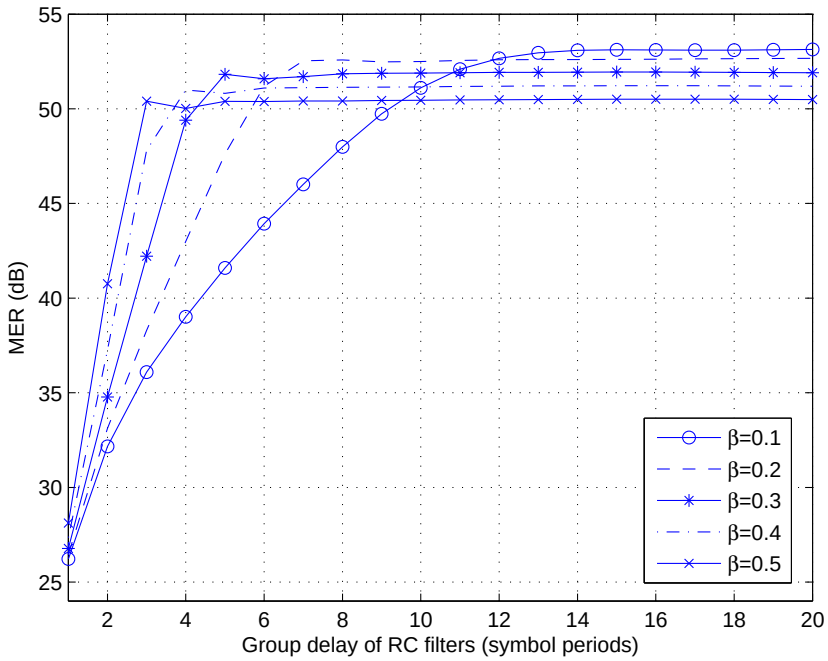

Figure 4.20 MER performance with the cubic interpolation filter for 8-PAM signals.

lator, parabolic interpolator and linear interpolator, were evaluated in terms of MER for PAM/QAM signals. Simulation results show that the cubic interpolator offers very good MER performance for RC-shaped PAM/QAM signals and it outperforms linear interpolator and parabolic interpolar by about 20 dB.

The main contributions of this chapter are the comparison between the two methods of timing recovery and the MER performance analysis of polynomial interpolation filters for PAM/QAM signals. The results can provide a good reference for a system designer on the selection of an interpolation filter and a pulse-shape filter given performance requirements and hardware resource availability.

5. Hardware Implementation in FPGA

This section presents the FPGA implementation of the performance analysis circuit for timing recovery in Figure 4.10 (Chapter 4). It focuses on the design of "polynomial interpolators" in Farrow structure, "input enable, fractional interval, and output enable generators", "clock domain interface", and "loop processor".

5.1 Polynomial Interpolators

The three interpolators considered in this thesis are linear interpolator, parabolic interpolator and cubic interpolator. Those interpolators are three most common type of polynomial interpolators. The mathematical background for these interpolators is reviewed in Appendix A.

Linear Interpolator

For a linear interpolator, Equation (3.7) is written as follows:

$$y[k] = x[m_k + 1]\mu_k + x[m_k](1 - \mu_k). \tag{5.1}$$

In essence, a linear interpolator can be considered as a filter with 2 coefficients, μ_k and $\mu_k + 1$. The design of a linear interpolation filter is shown in Figure 5.1 where input enable is an enable signal for the registers that store the input samples.

The input samples and output interpolants are chosen to be 14-bit signed numbers as many ADC/DAC standards use 14-bit length for digital signals. Signed numbers are represented in two's complement arithmetic. The fractional intervals are chosen to be 20-bit signed number. As $0 \leq \mu_k < 1$, the signed bit of μ_k, i.e., the most

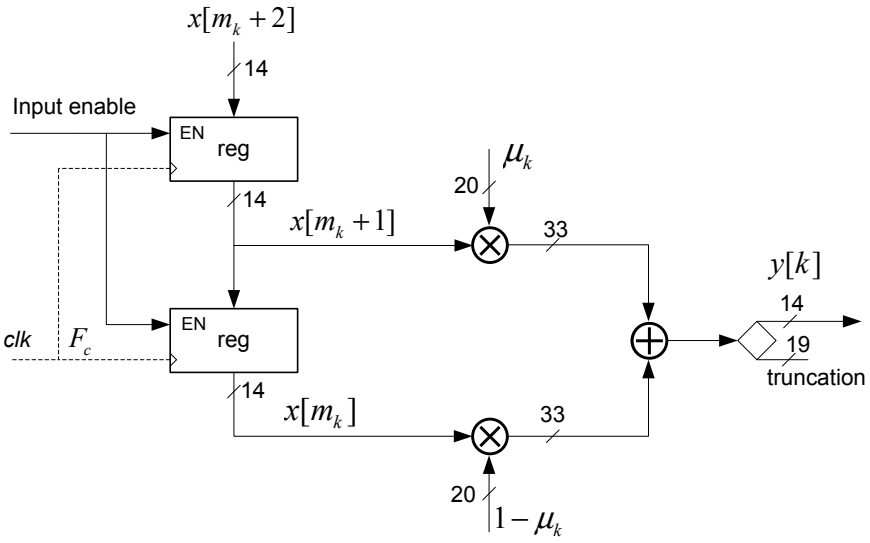

Figure 5.1 FPGA design of a linear interpolator.

significant bit, is always equal to 0. A fractional interval number is then quantized with an accuracy of 2^{-19}. There is a 19-bit truncation when computing the output $y[k]$. The design can be reduced to use only 1 multiplier if Equation (5.1) is rewritten as follows:

$$y[k] = (x[m_k + 1] - x[m_k])\mu_k + x[0]. \tag{5.2}$$

Figure 5.2 shows the equivalent FPGA design for Equation (5.2). Two adders are used instead of one adder in the design in Figure 5.1. But since a multiplier costs much more resource than an adder, the design which uses one multiplier is preferred.

The output interpolant $y[k]$ is then written to a register when output enable k is high. The output enable is also an enable signal for the register that stores $y[k]$. For the purpose of illustration, the design does not show delays in multipliers and adders. Normally the output of a multiplier or an adder is delayed by one clock cycle. In a real implementation, the input of an interpolator needs to be delayed according to the delay caused by adders and multipliers. For example, in Figure 5.2, if the first adder (the one that subtracts $x[m_k]$ from $x[m_k + 1]$) has one clock cycle delay, then the fractional interval μ_k has to be delayed one clock cycle before being used to multiply with the output of the first adder. Also, $x[k]$ needs to be delayed two clock cycles before being added to the output of the multiplier.

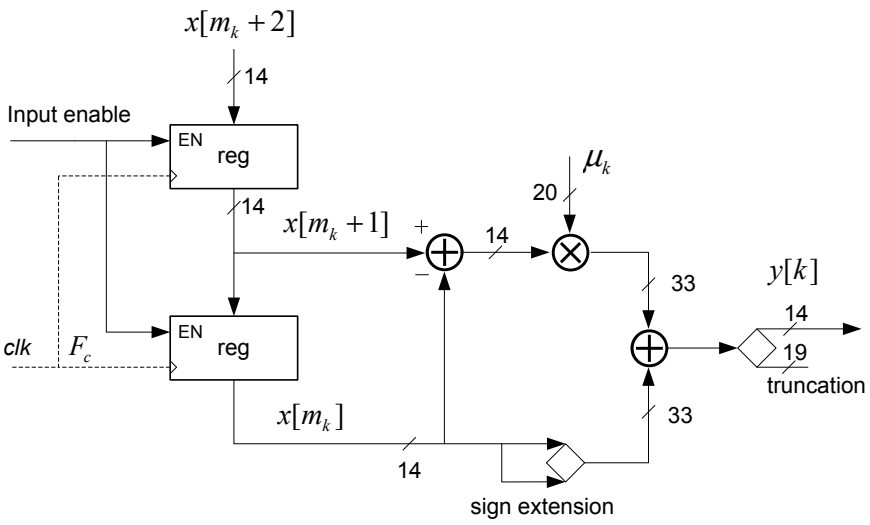

Figure 5.2 FPGA design of a linear interpolator with one multiplier.

An interpolation filter can compute interpolants basically in two approaches. The first approach is to compute the filter coefficients as functions of μ_k for each interpolant. In this way, all the filter coefficients need to be computed before generating the output interpolant. This approach is not efficient as the filter coefficients are polynomials of μ_k and computations require more resources. The second method is to the compute interpolants directly without finding filter coefficients. Since the purpose is to generate interpolants, not to explicitly build the filters, the second approach is the preferred one in a real implementation. For a linear interpolator, the two approaches are not much different. For the designs of parabolic interpolator and cubic interpolator, the two approaches are significantly different. The hardware structure for the second method was originally proposed by Farrow [3]. The designs of parabolic interpolator and cubic interpolator following the Farrow structure are described next.

Cubic Interpolator

The impulse response of a polynomial filter can also be represented in the following form [2]:

$$h_I[(i+\mu_k)T_{\text{in}}] = \sum_{l=0}^{N} b_l(i)\mu_k^l, \;\; i = I_1, \ldots, I_2. \tag{5.3}$$

Substituting (5.3) into (3.5) yields:

$$\begin{aligned} y[k] &= \sum_{i=I_1}^{I_2} x[m_k - i] \sum_{l=0}^{N} b_l(i)\mu_k^l \\ &= \sum_{l=0}^{N} \mu_k^l \underbrace{\sum_{i=I_1}^{I_2} b_l(i)x[m_k - i]}_{v(l)} \\ &= \sum_{l=0}^{N} \mu_k^l v(l), \end{aligned} \tag{5.4}$$

where

$$v(l) = \sum_{i=I_1}^{I_2} b_l(i)x[m_k - i]. \tag{5.5}$$

The most efficient approach to evaluate $y[k]$ is to use nested evaluation. For a cubic interpolator, it has the following form:

$$y[k] = [v(3)\mu_k + v(2)\mu_k + v(1)]\mu_k + v(0). \tag{5.6}$$

The evaluation of Equation (5.6) can be effectively implemented in hardware by using the *Farrow structure*. The computation of $y[k]$ is then performed by a cascade of N multiplications where N is the order of the polynomial. Here $N = 3$ for cubic interpolation. Each multiplier has the fractional interval μ_k as one of its input. In this way, the structure transfers only μ_k for each interpolation instead of I filter coefficients. The block diagram for cubic interpolation is shown in Figure 5.3, where $I_1 = -2$ and $I_2 = 1$. The Farrow coefficients $b_l(i)$ for cubic interpolator are found from Equation (A.16) and they are shown in Table 5.1.

Table 5.1 Farrow coefficients $b_l(i)$ for cubic interpolator.

i	$l=0$	$l=1$	$l=2$	$l=3$
-2	0	$-\frac{1}{6}$	0	$\frac{1}{6}$
-1	0	1	$\frac{1}{2}$	-1
0	1	$-\frac{1}{2}$	-1	$\frac{1}{2}$
1	0	$-\frac{1}{3}$	$\frac{1}{2}$	$-\frac{1}{6}$

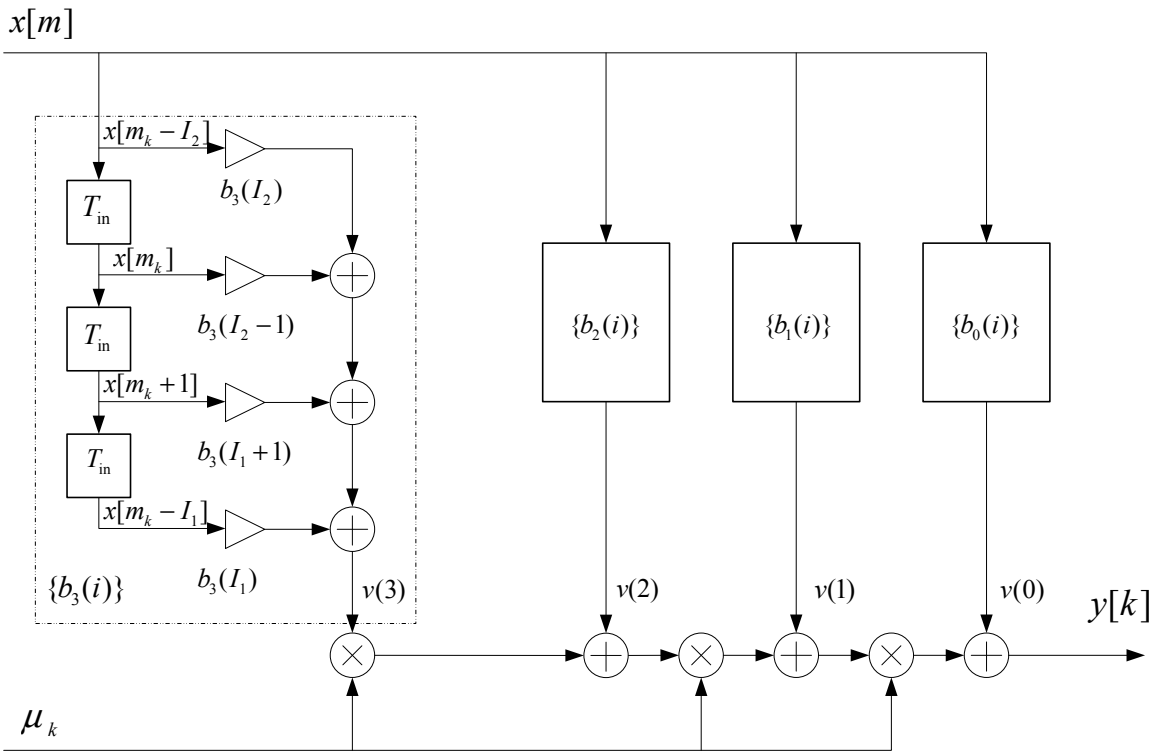

Figure 5.3 Farrow structure for cubic interpolator

As coefficients $b_l(i)$ are fixed, they can be implemented with a lookup table or using shift/add operations for some special cases. In this thesis, the coefficients in Table 5.1 are scaled up 6 times and the result are divided by 6. In this way, coefficients $b_l(i)$ become 0, ± 1, ± 2, ± 3, and ± 6. The multiplication by 2 can be easily implemented by a shift operation. Similarly, the multiplication by 3 can be implemented by a shift operator and an adder. The multiplication by 6 can be computed by a multiplication by 3 and a multiplication by 2. Also, the division by 6 of the result can be implemented by a shift operator and a multiplication by a constant 1/3.

Parabolic Interpolator

The coefficients $b_l(i)$ for a parabolic interpolator can be derived from Equation (A.20). Table 5.2 lists the coefficients where α is the parameter that controls the piecewise parabolic function (see Appendix A).

It can be seen that the coefficients become convenient to use when $\alpha = 0.5$. The

Table 5.2 Farrow coefficients $b_l(i)$ for piecewise parabolic interpolator.

i	$l=0$	$l=1$	$l=2$
-2	0	$-\alpha$	α
-1	0	$\alpha+1$	$-\alpha$
0	1	$\alpha-1$	$-\alpha$
1	0	$-\alpha$	α

multiplication by 0.5 can be implemented as a shift operator. The multiplication by 1.5 can be implemented as a shift operator and an adder. The Farrow structure for a parabolic interpolator is similar to the structure of a cubic interpolator. It is actually simpler with only 2 cascade multiplications by μ_k.

Farrow structure provides an efficient method to implement polynomial interpolators in hardware. In a real implementation, as discussed for the case of linear interpolator, proper clock delays caused by adders/multiplicators need to be taken into account. Also, enable signals for registers need to be properly controlled.

5.2 Fractional Interval Generator

"Fractional interval generator" is a short name used for the module "Input enable, fractional interval, and output enable". The design of a fractional interval generator is similar for both resampler/timing recovery methods (i.e., input clock time base or output clock time base). The only difference is that in the input clock time base method, in the process of generating fractional intervals, the circuit also generates output enable signal k. On the other hand, with the output clock time base method, the circuit generates the input enable m. The design of each method is discussed separately in the following.

Input Clock Time Base Method

In the performance analysis circuit (see Figure 4.10), the resampler uses fixed step size 0.99 to resample the rate F_{tx} to $0.99F_{\text{tx}}$. As discussed in Chapter 4, the resampler

can be clocked at rate F_{tx}. Since the resampler is clocked with the same rate as the input sampling rate, $M = 1$, $l_k = 1$, and the step size $\Delta = \hat{T}_{\text{in}}/T_{\text{out}} = 0.99$. Then Equation (3.11) becomes

$$\mu_k = 1 - \frac{1}{\Delta} \times \text{frac}\left[\frac{Y_{\text{in}}}{T_{\text{out}}}\right]. \tag{5.7}$$

This setting makes the hardware design much more easier since the input clock time base, the output clock time base and the fractional interval generator blocks can be integrated into one single block as illustrated in Figure 5.4.

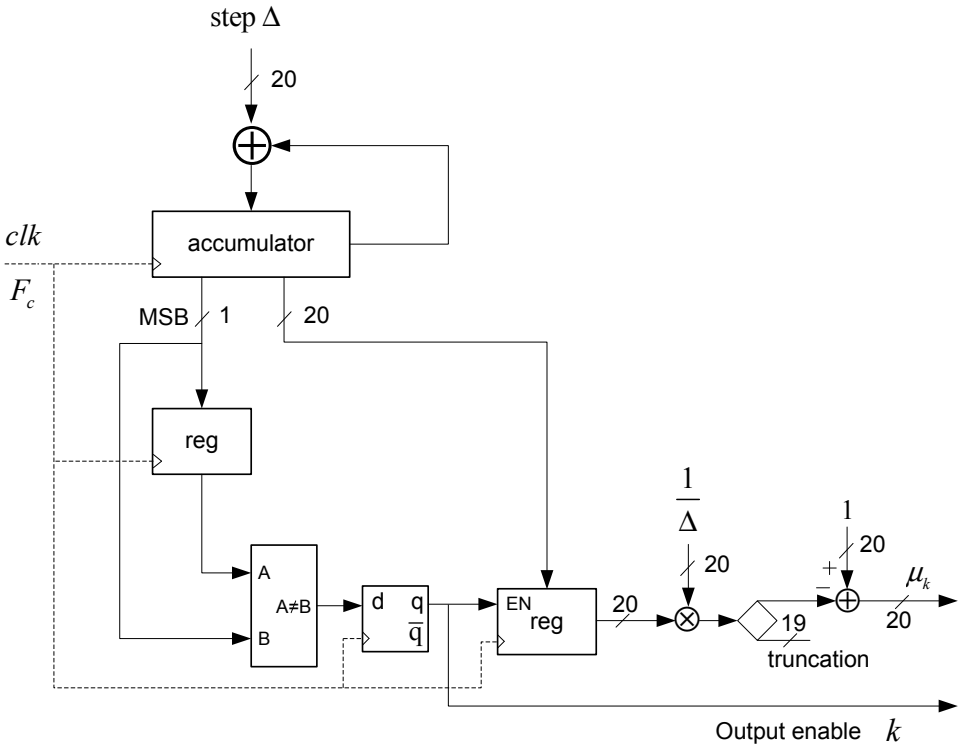

Figure 5.4 Fractional interval generator for input clock time base method.

The accumulator in Figure 5.4 is the modified input clock time base. The accumulator uses 20 least significant bits to keep the fractional part of the input clock time base and 1 most significant bit (MSB) to detect integer-cross overs. At every positive edge of clock that MSB changes, an integer-cross over is detected, and the output enable k is set to high. At a positive edge of clock that MSB does not change, there is no integer-cross over and k is set to low. Recall that integer-cross over means the input clock time base crosses the output clock time base. When k is high and μ_k

is available, a new interpolant is generated. Note that in the design, there is no need to actually generate output clock time base. It is implied that values of output clock time base are integer numbers.

Output Clock Time Base Method

A similar approach is applied to the fractional interval generator in the output clock time base method. The input clock time base, the output clock time base and the fractional interval generator blocks are integrated into one single block. The design is shown in Figure 5.5 for a general case and it is much simpler than in the input clock time base method.

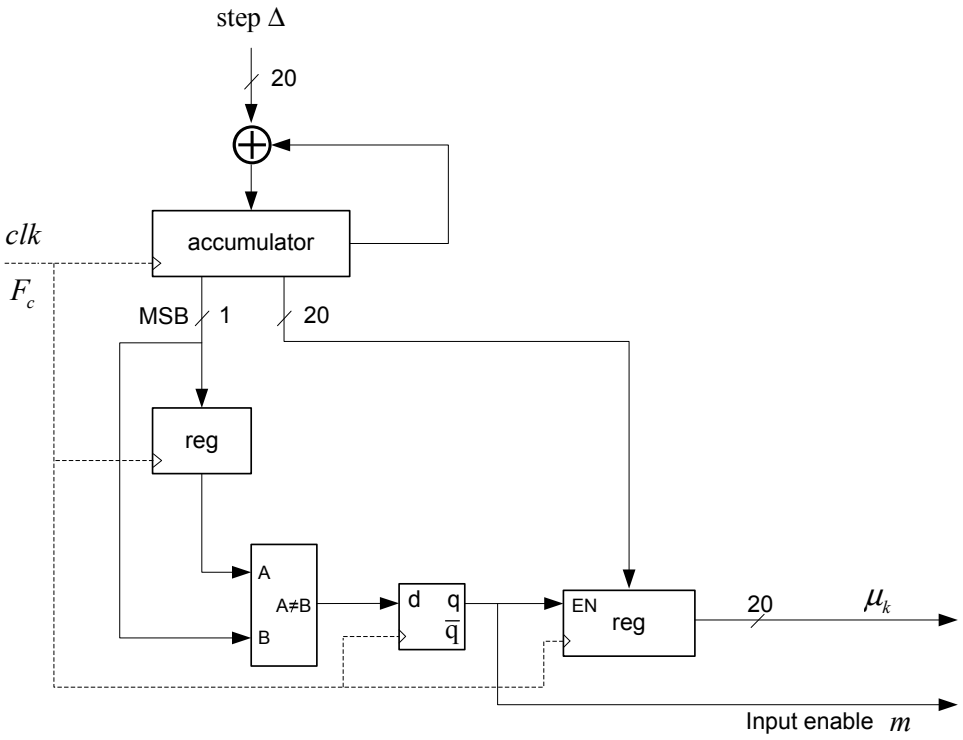

Figure 5.5 Fractional interval generator for output clock time base method.

Similarly, the accumulator in Figure 5.5 is the modified output clock time base. The fractional part of the output clock time base is kept in the 20 least significant bits of the accumulator. Integer-cross overs are detected from the accumulator's 1 MSB. At every positive edge of clock that MSB changes, an integer-cross over is detected, and the input enable m is set to high. That means one input sample is loaded to

the interpolator. At a positive edge of clock that MSB does not change, there is no integer-cross over and m is set to low. There is no need to actually generate input clock time base as it is understood to be an integer.

5.3 Loop Processor

Figure 5.6 shows a design of the loop filter inside the loop processor (see Figure 4.5). The filter is enabled by the symbol enable signal as timing error is only available every symbol period.

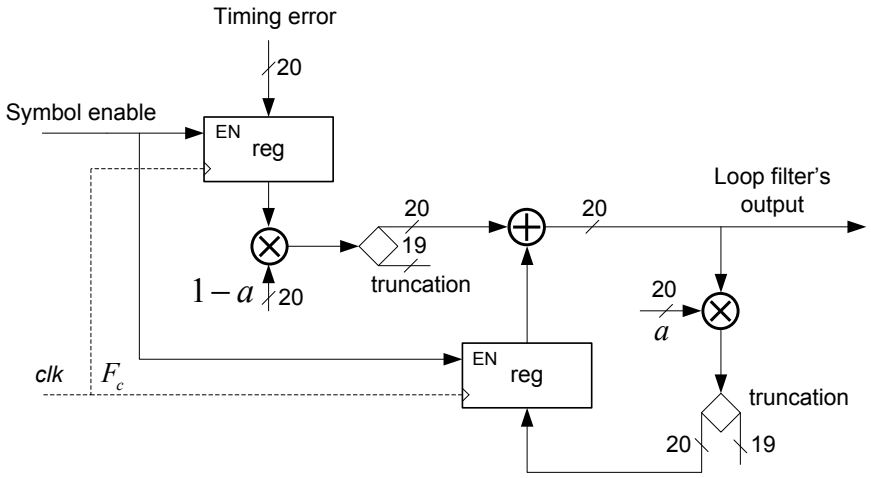

Figure 5.6 FPGA design of a loop filter.

The designs of Gardner TED and and other parts of the loop processor are straightforward from the descriptions in Figures 4.4, 4.5, respectively.

5.4 Clock Domain Interface

A "clock domain interface" interfaces two blocks/systems working at two different clocks. This interfacing is one of the main challenges in hardware design as data transfers may suffer from setup and hold violation, metastability and unreliable. There are several approaches to solve the problem of interfacing. Two of the most common ones are handshake signaling and asynchronous FIFO (First In First Out). In handshake signaling, sender asserts the request-to-send signal, and receiver asserts the acknowledge-to-accept signal. Data is only transferred after the handshaking has

been setup. An asynchronous FIFO stack has two interfaces. One interface for writing the data into the FIFO by a writing clock and a writing enable signal. The other for reading the data out by a reading clock and a reading enable signal. The second method is more efficient in our system as it provides faster exchange rates than in the first method, where the handshaking takes several clock cycles to set up.

The design for a clock domain interface is shown in Figure 5.7 [7]. The design builds on the idea of asynchronous FIFO with some modifications. For example, there is no full or empty signal as in a common FIFO .

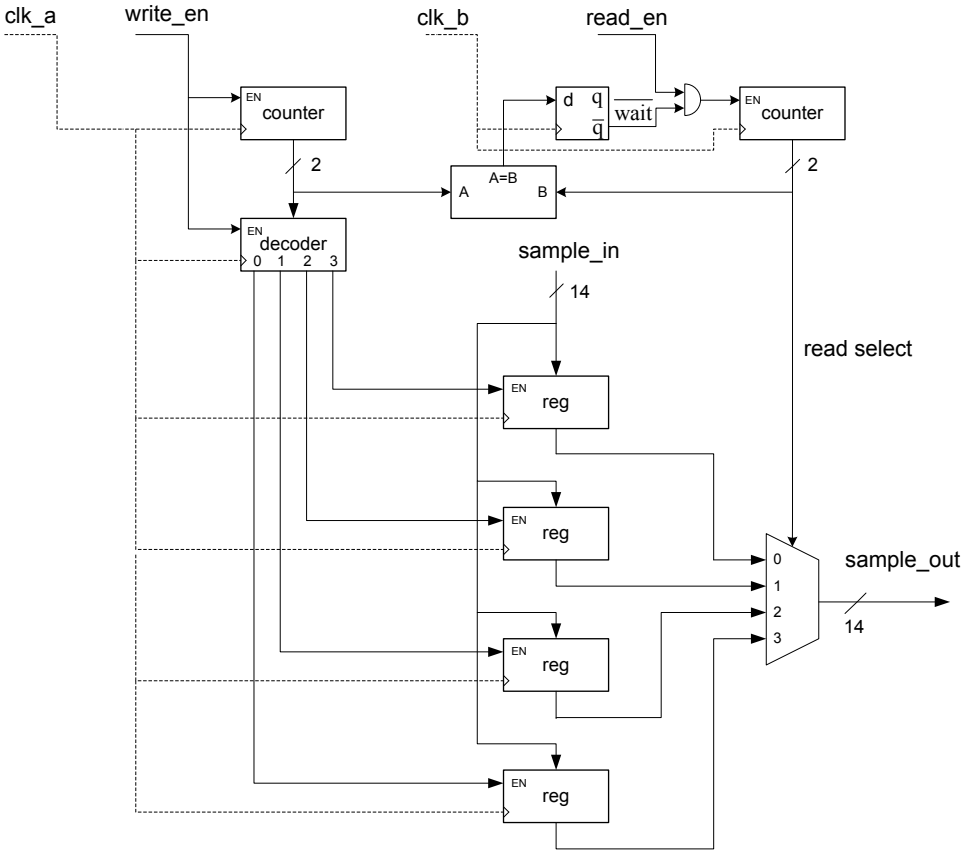

Figure 5.7 FPGA design of a clock domain interface.

Samples are written to the interface by clk_a and controlled by write_en (write enable) signal. Samples are read from the interface by by clk_b and controlled by read_en (read enable) signal. There are 4 registers to store samples in the interface. Two counters are included to control the writing and reading inside the interface. Also, a $\overline{\text{wait}}$ signal is introduced to avoid the situation of writing and reading one

register at the same time. In the steady state, $\overline{\text{wait}}$ will be inactive.

5.5 Performance Analysis

The performance analysis circuit was designed using Verilog HDL (Hardware Description Language) and implemented in an Altera DE2 board. The generation of random input binary sequence in hardware can be implemented by means of a linear feedback shift register (LFSR). A LFSR can produce a sequence of bits which appears random, i.e., pseudo-random sequence.

Data from the FPGA device is captured via a SignalTap module and then transferred into MATLAB workspace for performance analysis. The length of the symbols to be captured in SignalTap depends on the memory of the FPGA device. For a 2-PAM signal, the signal is constructed by passing a sequence ± 1 through a RC filter with roll off factor $\beta = 0.25$. The choice of $\beta = 0.25$ follows from the simulation setup in Chapter 4, which provides good MER performance without the need of a long RC filter's length. Figures 5.8 and 5.9 show the input symbol sequence and the output symbol sequence after timing recovery for MER values of 26.5 and 50dB, respectively.

For Figures 5.8 and 5.9, 2-PAM symbols are two's complement numbers with 14-bit length. The input signal is a pseudo-random $\pm 2^{11}$ sequence generated from the LFSR module. The result suggests the circuit works properly since timing has been recovered and symbols have been detected correctly. Figure 5.10 shows the MER performance of the cubic interpolation filter for different lengths of RC pulse-shaping filters.

The MER performance curve in Figure 5.10 matches very well with the simulation results. Obviously there is always some difference in performance between a real implementation and a MATLAB/Simulink simulation. The difference in this case is about 2.5 dB. This is most likely caused by the effects of truncation in the real implementation. The agreement between implementation and simulation results suggests that the design methodology and circuits in this thesis are practical and suitable for

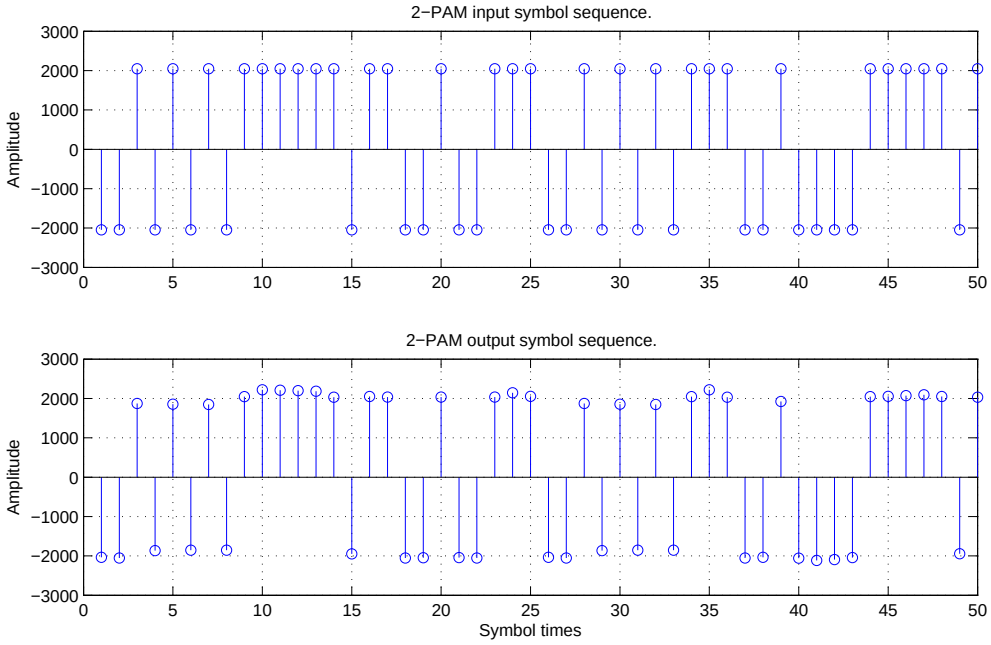

Figure 5.8 2-PAM input symbol sequence and output symbol sequence for MER of 26.5 dB.

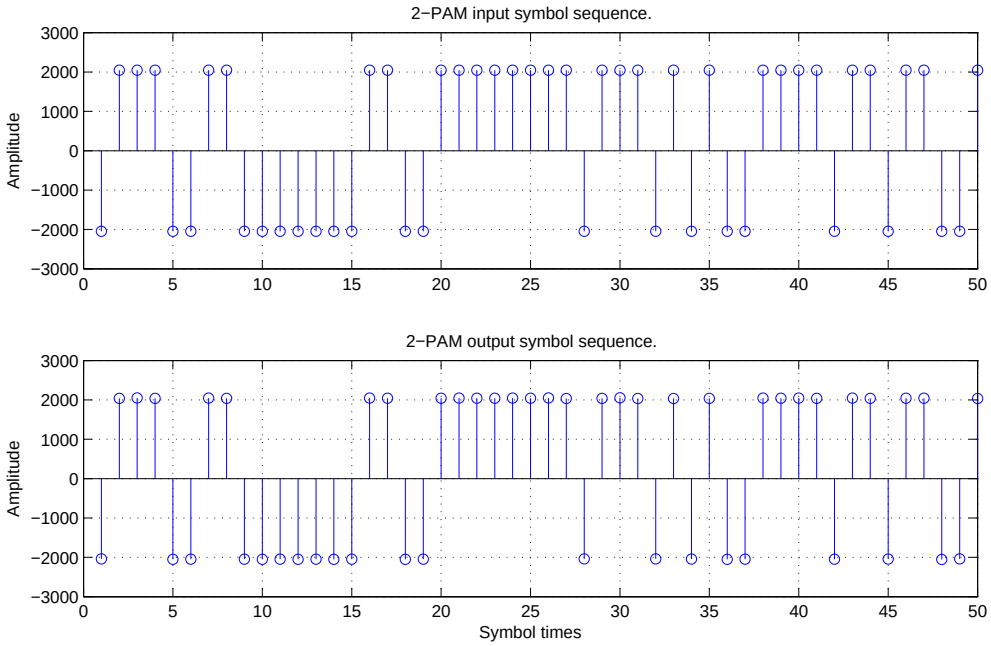

Figure 5.9 2-PAM input symbol sequence and output symbol sequence for MER of 50 dB.

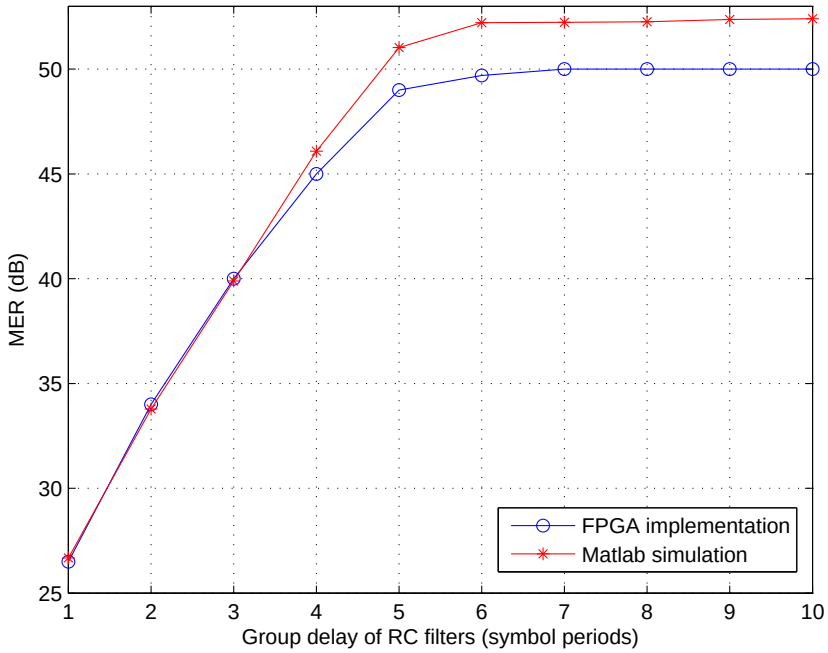

Figure 5.10 MER performance of the cubic interpolation filter for 2-PAM signals in FPGA implementation and MATLAB simulation for $\beta = 0.25$.

hardware implementation.

5.6 Summary

This chapter presented the FPGA implementation of the performance analysis circuit for timing recovery of PAM/QAM signals introduced in Chapter 4. The hardware structure of the circuit is described in detail and the circuit was implemented in a real FPGA device. The implementation results show that the circuit works properly, which validates the practicality of the resampler and timing recovery circuits described in this thesis.

6. Conclusions

The methodology of time base generation and synchronization provides a very practical and convenient approach to digital resampling. Based on the methodology, two methods were applied to devise resampling circuits, which can perform virtually any rate conversion. Both circuits offer similar performances but one of them, namely the resampler circuit which uses the output clock time base, does not require any division to compute the position of the new samples with respect to the incoming samples. Also, the computation is more desirable as it happens at positive edges of the output sampling clocks and this makes the circuit even more simpler. These are significant advantages in hardware implementation.

The performance of polynomial interpolation filters was investigated in terms of MER. The proposed model was able to measure effectively the MER performance due to the use of different interpolation filters. It was shown that MER performance depends strongly on the spectrum of the input signal, which is governed by the pulse-shaping filter. In general, the cubic interpolator offers an excellent performance, which can be as high as about 52.5 dB. The linear interpolator, with a very simple structure, can provide a decent MER performance, about 31.5 dB. This may still lead to very good BER results in some applications. The parabolic interpolator, which is more complicated than the linear interpolator, offers only a slightly better performance (when the parameter α is set to 0.5). The results also show that the order of PAM/QAM modulation has only a small effect to the MER performance.

The excellent agreement between MER performance from MATLAB/Simulink simulation and hardware implementation gives a strong evidence about the perfor-

mance of the interpolation filters if being used in an actual QAM system. The results can provide a system designer with a good selection of pulse-shaping filter's length to meet the requirements on performance and hardware resources.

Future research could be done on some modifications of the interpolation filters so that they can provide a better MER performance for QAM signals. The linear interpolation filter is very attractive for modification as it has a very simple structure. Also, other types of interpolation filters and pulse-shaping filters can be investigated.

A. Polynomial Interpolation Filters

For convenience, the mathematical model of interpolation in Figure 3.2 (Chapter 3) is redrawn in Figure A.1.

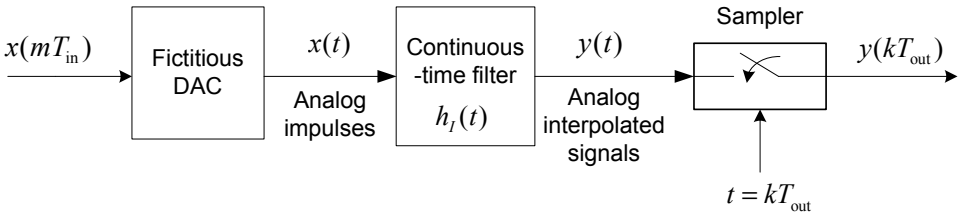

Figure A.1 Rate conversion with continuous-time filter.

The interpolants are computed at time $kT_{\text{out}} = (m_k + \mu_k)T_{\text{in}}$ and the interpolation equation is

$$y(kT_{\text{out}}) = \sum_{i=I_1}^{I_2} x((m_k - i)T_{\text{in}})h_I((i + \mu_k)T_{\text{in}}). \tag{A.1}$$

The bandlimited signal $x(t)$ in Figure A.1 can be reconstructed, i.e., $y(t) = x(t)$, if the filter $h_I(t)$ is an ideal filter with impulse response [17]

$$h_I(t) = \frac{\sin(\pi t/T_{\text{in}})}{\pi t/T_{\text{in}}}. \tag{A.2}$$

As can be seen, the ideal filter is not practical since it is an IIR and noncausal filter. That means $y(t)$ can never be the same as $x(t)$ in practise. Since the objective here is to compute the interpolants, a practical interpolation filter does not need to recover the input waveform and can be derived from many other types of mathematical functions. Among which the most efficient one for hardware implementation is polynomial [2]. This means that the impulse response $h_I(t)$ of the continuous filter in Figure A.1 is a polynomial in t and $y(t)$ is approximated by a varying continuous-time

polynomial $p_k(t)$, i.e., $y(kT_{\text{out}}) = p_k(kT_{\text{out}})$. The subscript k in $p_k(t)$ indicates that the approximating polynomial is different for each $y(kT_{\text{out}})$.

As a special case, if $p_k(mT_{\text{in}}) = x(mT_{\text{in}})$ for all I points of the k^{th} basepoint set, and for all k, then $p_k(t)$ is said to be an interpolating polynomial. An interpolating polynomial can be described in terms of its Lagrange coefficients. The coefficients are polynomials of degree $I-1$ in t, or equivalently μ_k if a basepoint index m_k is explicitly defined. To obtain a unique basepoint set for an interpolant, the two following conditions must be satisfied [17]:

- the number of samples in the basepoint set must be even,
- interpolation is performed only in the central interval of the basepoint set.

The three most common types of interpolating polynomials are linear interpolation, parabolic interpolation, and cubic interpolation. Those interpolators are discussed next.

A.1 Linear Interpolator

For the case of linear interpolator, the interpolating filter takes only two coefficients, with indices $I_1 = -1$ and $I_2 = 0$. The interpolation equation is

$$\begin{aligned} y(kT_{\text{out}}) &= \sum_{i=-1}^{0} x[(m_k - i)T_{\text{in}})]h_I[(i+\mu_k)T_{\text{in}}] && \text{(A.3)} \\ &= x[m_k T_{\text{in}}]h_I[\mu_k T_{\text{in}}] + x[(m_k+1)T_{\text{in}}]h_I[(-1+\mu_k)T_{\text{in}}]. && \text{(A.4)} \end{aligned}$$

Without lost of generality, let $T_{\text{in}} = 1$, and note that $kT_{\text{out}} = (m_k + \mu_k)T_{\text{in}}$. The linear interpolation equation becomes

$$y(m_k + \mu_k) = x[m_k]h_I[\mu_k] + x[m_k+1]h_I[-1+\mu_k]. \tag{A.5}$$

And also, the basepoint index m_k can be set to 0 without any lost of generality. Then we have

$$y(\mu_k) = x[0]h_I[\mu_k] + x[1]h_I[-1+\mu_k]. \tag{A.6}$$

Now we need to find the Lagrange coefficients $h_I[\mu_k]$ and $h_I[(-1+\mu_k)]$ for the linear interpolator. The approach is to fit a first order polynomial, $p_k(\mu) = a\mu + b$, through $x[0]$, $x[1]$ and then compute $y(\mu_k)$ by taking value of $p_k(\mu)$ at $\mu = \mu_k$. To find a and b, notice that

$$\begin{cases} p_k[0] = x[0] = b \\ p_k[1] = x[1] = a + b \end{cases} \tag{A.7}$$

Thefore

$$\begin{cases} a = x[1] - x[0] \\ b = x[0] \end{cases} \tag{A.8}$$

The first order polynomial $p_k(\mu)$ is

$$p_k(\mu) = (x[1] - x[0])\mu + x[0] \tag{A.9}$$

Now to compute $y(\mu_k)$, we take the value of $p_k(\mu)$ at $\mu = \mu_k$:

$$\begin{aligned} y(\mu_k) &= p_k(\mu_k) = a\mu_k + b \\ &= (x[1] - x[0])\mu_k + x[0] = x[1]\mu_k + x[0](1 - \mu_k) \end{aligned} \tag{A.10}$$

From Equations (A.10) and (A.6) we have the following Lagrange coefficients for a linear interpolator:

$$\begin{cases} h_I[\mu_k] = 1 - \mu_k \\ h_I[-1 + \mu_k] = \mu_k \end{cases}$$

The frequency response of a linear interpolator filter is plotted in Figure A.2 for $\mu_k = 0.1, 0.3, 0.5$. Note that the responses for μ_k and $1 - \mu_k$ are the same.

Figure A.2 shows the amplitude and phase distortions caused by the filter to the signal. It can be seen that the linear interpolation filter attenuates the amplitude of a signal more when μ_k is increased from 0 to 0.5. There is a significant distortion for the case of $\mu_k = 0.5$ at the high frequency range.

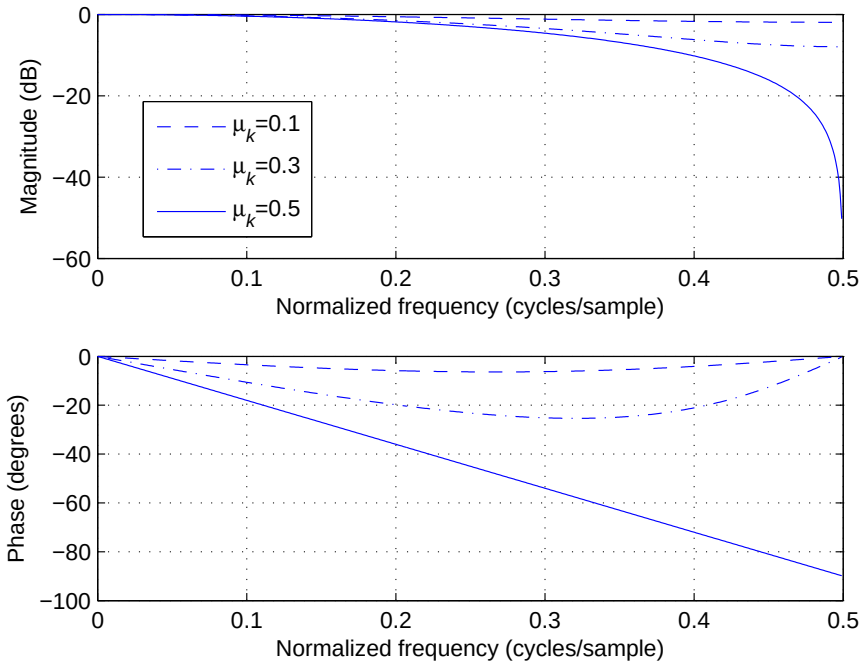

Figure A.2 Frequency response of the linear interpolator.

The group delay of a filter, denoted by τ_{g}, is defined as the derivative of radian phase shift with respect to radian frequency [18]:

$$\tau_{\mathrm{g}} = -\frac{d\phi(\omega)}{d\omega}, \tag{A.11}$$

where ω is the angular frequency and $\phi(\omega)$ is the radian phase shift introduced by the filter. Equation (A.11) suggests that if the phase response of a filter is linear, then the filter has a constant group delay at all frequencies. This constant group delay is a desirable property of a filter as there is no phase distortion for selected frequencies.

As the phase response of the linear interpolation filter with $\mu_k = 0.5$ is linear, the group delay is equal to minus the slope of the linear phase response of the filter. It is 0.5 samples in this case. The linear phase is a good filter characteristic, but it is not the case for other values of μ_k. When the phase response is not linear, the filter introduces phase distortion to the output signal.

A.2 Cubic Interpolator

For a cubic interpolator, there are four coefficients, and $I_1 = -2$ and $I_2 = 1$. The interpolation equation is

$$y(kT_{\text{out}}) = \sum_{i=-2}^{1} x[(m_k - i)T_{\text{in}}]h_I[(i + \mu_k)T_{\text{in}}] \tag{A.12}$$

Without lost of generality, we can set T_{in} equal 1 and eliminate T_{in} from the above equation. Noting that $kT_{\text{out}} = (m_k + \mu_k)T_{\text{in}}$, the cubic interpolation equation become

$$y(m_k + \mu_k) = \sum_{i=-2}^{1} x[m_k - i]h_I[i + \mu_k] \tag{A.13}$$

The basepoint index m_k can also be set to 0 without any lost of generality. This gives

$$y(\mu_k) = \sum_{i=-2}^{1} x[-i]h_I[i + \mu_k] \tag{A.14}$$

To find the Lagrange coefficients $h_I[(i + \mu_k)]$ for $i = -2, -1, 0, 1$, pass a third order polynomial, $p_k(\mu) = a\mu^3 + b\mu^2 + c\mu + d$, through $x[-1]$, $x[0]$, $x[1]$, $x[2]$ and then compute $y(\mu_k)$ by taking value of $p_k(\mu)$ at $\mu = \mu_k$. The results are:

$$\begin{cases} a = -\frac{1}{6}x[-1] + \frac{1}{2}x[0] - \frac{1}{2}x[1] + \frac{1}{6}x[2] \\ b = \frac{1}{2}x[-1] - x[0] + \frac{1}{2}x[1] \\ c = -\frac{1}{3}x[-1] - \frac{1}{2}x[0] + x[1] - \frac{1}{6}x[2] \\ d = x[0] \end{cases} \tag{A.15}$$

From a, b, c, d in Equation (A.15) one has an equation for the polynomial $p_k(\mu)$. The interpolant $y(\mu_k) = p_k(\mu_k)$ is computed as follows:

$$\begin{aligned} y(\mu_k) = {} & x[2]\left(\frac{1}{6}\mu_k^3 - \frac{1}{6}\mu_k\right) + x[1]\left(-\frac{1}{2}\mu_k^3 + \frac{1}{2}\mu_k^2 + \mu_k\right) \\ & + x[0]\left(\frac{1}{2}\mu_k^3 - \mu_k^2 - \frac{1}{2}\mu_k + 1\right) + x[-1]\left(-\frac{1}{6}\mu_k^3 + \frac{1}{2}\mu_k^2 - \frac{1}{3}\mu_k\right) \end{aligned} \tag{A.16}$$

From Equation (A.16), the Lagrange coefficients for a cubic interpolator are:

$$\begin{cases} h_I[-2 + \mu_k] = \frac{1}{6}\mu_k^3 - \frac{1}{6}\mu_k \\ h_I[-1 + \mu_k] = -\frac{1}{2}\mu_k^3 + \frac{1}{2}\mu_k^2 + \mu_k \\ h_I[\mu_k] = \frac{1}{2}\mu_k^3 - \mu_k^2 - \frac{1}{2}\mu_k + 1 \\ h_I[1 + \mu_k] = -\frac{1}{6}\mu_k^3 + \frac{1}{2}\mu_k^2 - \frac{1}{3}\mu_k \end{cases} \tag{A.17}$$

The frequency response of a cubic interpolation filter is plotted in Figure A.3 for $\mu_k = 0.1, 0.3, 0.5$.

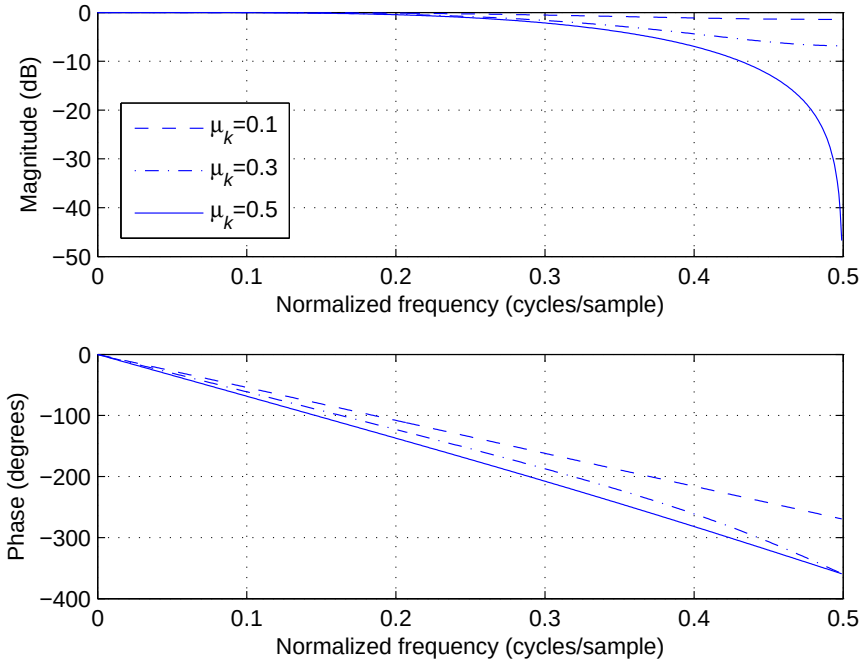

Figure A.3 Frequency response of the cubic interpolator.

Similar to the linear interpolation filter, the cubic interpolation filter causes the worst distortion to the signal amplitude for the case $\mu_k = 0.5$. Comparing Figure A.3 with Figure A.2, it is seen that the cubic interpolator causes less amplitude distortions to the signal than in the case of the linear interpolator.

A.3 Piecewise Parabolic Interpolator

Beside the classical linear and cubic polynomial interpolators, Erup [2] proposed a four points interpolating filter with piecewise parabolic impulse response. Similarly to the case of a cubic interpolator, set $I_1 = -2$, $I_2 = 1$, $T_{\text{in}} = 1$, and $m_k = 0$. The

interpolating filter has four coefficients [11]:

$$\begin{cases} h_I[-2+\mu_k] = \alpha\mu_k^2 - \alpha\mu_k \\ h_I[-1+\mu_k] = -\alpha\mu_k^2 + (\alpha+1)\mu_k \\ h_I[\mu_k] = -\alpha\mu_k^2 + (\alpha-1)\mu_k + 1 \\ h_I[1+\mu_k] = \alpha\mu_k^2 - \alpha\mu_k \end{cases} \tag{A.18}$$

where α is the parameter that controls the piecewise parabolic function. The interpolation equation becomes

$$\begin{aligned} y(\mu_k) &= \sum_{i=-2}^{1} x[(-i)]h_I[(i+\mu_k)] \\ &= x[2](\alpha\mu_k^2 - \alpha\mu_k) + x[1](-\alpha\mu_k^2 + (\alpha+1)\mu_k) \\ &\quad + x[0](-\alpha\mu_k^2 + (\alpha-1)\mu_k + 1) + x[-1](\alpha\mu_k^2 - \alpha\mu_k) \end{aligned} \tag{A.19}$$

Recall that $\alpha = 0.5$ significantly reduces the hardware complexity. The frequency response of the cubic interpolator filter is plotted in Figure A.4 for $\mu_k = 0.1, 0.3, 0.5$ and $\alpha = 0.5$. The filter, again, attenuates the most to the signal amplitude when $\mu_k = 0.5$. The frequency responses of parabolic interpolation filters for different α are plotted in Figure A.4 for $\mu_k = 0.5$.

Figure A.5 demonstrates that a parabolic interpolation filter might add some amplitude gain to a QAM signal. Comparison of polynomial filters is discussed next.

A.4 Comparison

To compare performances of the interpolation filters in terms of their frequency responses, let consider QAM signals with bandwidths $B_1 = 1/8 = 0.125$, $B_2 = 1.25/8 = 0.15625$ and $B_3 = 2/8 = 0.25$ cycles per sample (see Equation (2.18)). The digital bandwidths B_1, B_2, B_3 are the bandwidths of a QAM signal after passing through RC filters with roll-off factors 0, 0.25 and 1, respectively. The frequency responses of the ideal interpolation filter (Equation (A.2), truncated to be at 32-sample length) and polynomial filters with a fractional delay $\mu_k = 0.5$ are plotted in Figure A.6.

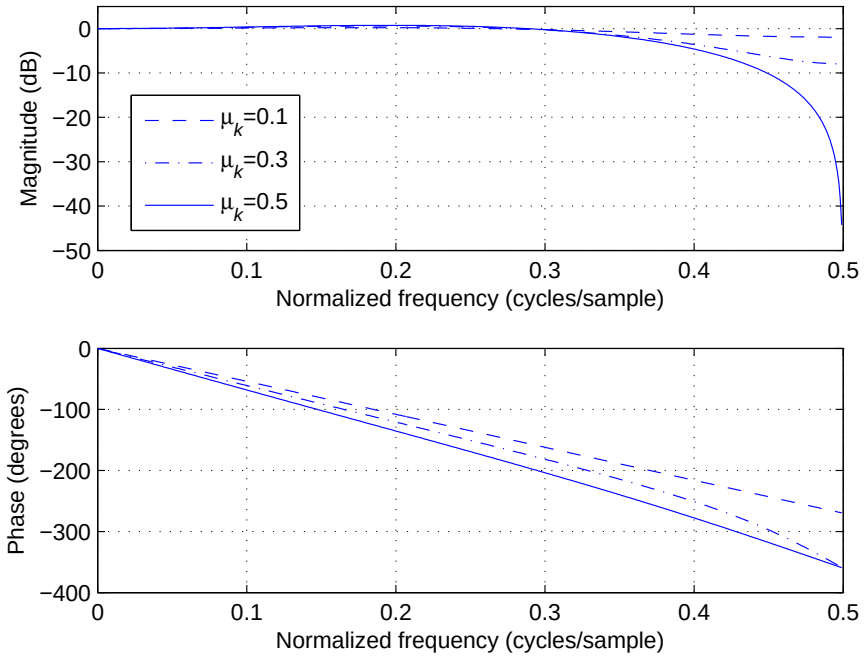

Figure A.4 Frequency response of the parabolic interpolator.

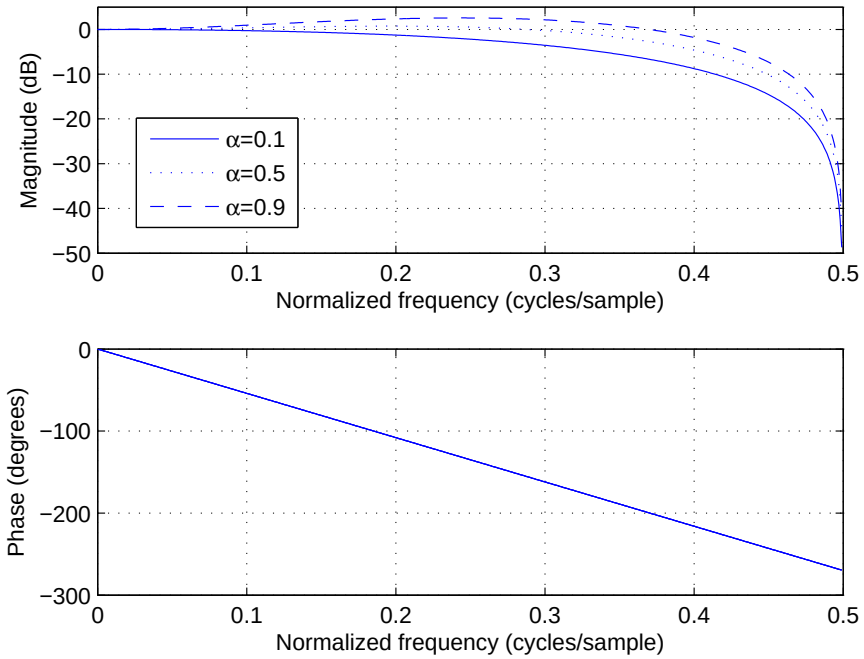

Figure A.5 Frequency response of the parabolic interpolator for $\mu_k = 0.5$ as a function of α.

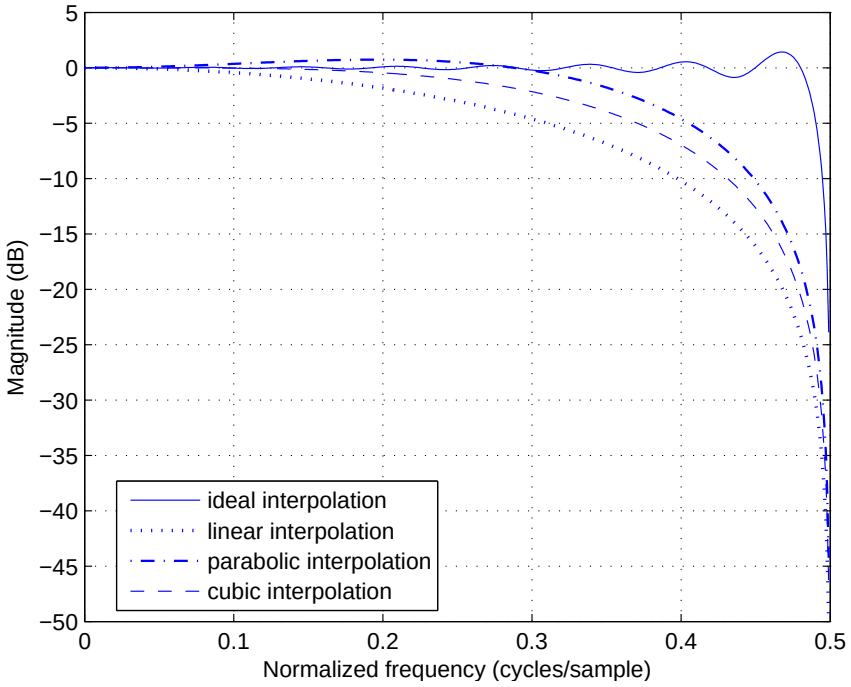

Figure A.6 Frequency responses of the ideal interpolation filter and polynomial filters for $\mu_k = 0.5$.

From the frequency responses of the filters, distortions (measured in dB) caused by interpolation filters are shown in Table A.1. Note that the parabolic filter is implemented with $\alpha = 0.5$. See also Figure 4.14 for comparison.

Table A.1 Amplitude distortions (dB) caused by interpolation filters for $\mu_k = 0.5$.

Frequency	Ideal	Linear	Parabolic	Cubic
0.125	0.02	0.68	−0.48	0.08
0.15625	−0.1	1.1	−0.65	0.2
0.25	0.12	3	−0.5	1.1

In Table A.1, a negative distortion means a filter adds some amplitude gain to the output signal. The table shows that the cubic interpolator offers a good performance for QAM signal in general while the linear interpolator only offers a good performance at a low frequency range (or at a high sampling frequency). Also, the parabolic interpolator gives good filter characteristics from its frequency responses. Recall the

big difference in MER performances with the use of the cubic interpolation filter in comparison with the linear and parabolic interpolation filters (Chapter 4). It is clear that frequency responses do not tell much about actually effects of interpolation on QAM signals.

References

[1] F. Gardner, "A BPSK/QPSK timing-error detector for sampled receivers," *IEEE Transactions on Communications*, vol. 34, pp. 423–429, May 1986.

[2] L. Erup, F. Gardner, and R. Harris, "Interpolation in digital modems. II. Implementation and performance," *IEEE Transactions on Communications*, vol. 41, pp. 998–1008, Jun. 1993.

[3] C. Farrow, "A continuously variable digital delay element," in *IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 2641–2645, Jun. 1988.

[4] E. R. Pelet, *Synchronization in All-Digital QAM Receivers*. PhD thesis, University of Saskatchewan, Saskatoon, Saskatchewan, Canada, 2009.

[5] J. Ketola, J. Vankka, and K. Halonen, "Synchronization of fractional interval counter in non-integer ratio sample rate converters," *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003.*, vol. 2, pp. II–89 – II–92, May 2003.

[6] J. Vankka, J. Ketola, O. Vaananen, J. Sommarek, M. Kosunen, and K. Halonen, "A GSM/EDGE/WCDMA modulator with on-chip D/A converter for base station," *IEEE International, Solid-State Circuits Conference, 2002.*, vol. 1, pp. 236–463, 2002.

[7] E. Salt, *Lecture Notes on Resampling in Hardware*. University of Saskatchewan, Saskatoon, Saskatchewan, Canada, 2009.

[8] R. Pulikkoonattu, H. Subramanian, and S. Laxman, "Least square based piecewise parabolic interpolation for timing synchronization," in *IEEE Radio and Wireless Symposium*, pp. 155 –158, Jan. 2008.

[9] K. Bucket and M. Moeneclaey, "The effect of interpolation on the BER performance of narrowband BPSK and (O)QPSK on Rician-fading channels," *IEEE Transactions on Communications*, vol. 42, pp. 2929–2933, Nov. 1994.

[10] H. H. Nguyen and E. Shwedyk, *A First Course in Digital Communications*. Cambridge University Press, 2009.

[11] M. Rice, *Digital Communications - A Discrete-Time Approach*. Pearson Prentice-Hall, 2009.

[12] S. Daumont, B. Rihawi, and Y. Lout, "Root-raised cosine filter influences on PAPR distribution of single carrier signals," *3rd International Symposium on Communications, Control and Signal Processing, 2008*, vol. 41, pp. 841–845, 2008.

[13] F. Gardner, "Interpolation in digital modems. I. Fundamentals," *IEEE Transactions on Communications*, vol. 41, pp. 501–507, Mar 1993.

[14] F. M. Gardner, *Phasedlock Techniques, 3rd edition*. 2005.

[15] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, 1996.

[16] J. G. Proakis, *Digital Communications, 4th edition*. McGraw-Hill, 2000.

[17] R. Schafer and L. Rabiner, "A digital signal processing approach to interpolation," *Proceedings of the IEEE*, vol. 61, pp. 692–702, Jun. 1973.

[18] J. Blauert and P. Laws, "Group delay distortions in electroacoustical systems," *Journal of the Acoustical Society of America*, vol. 63, pp. 1478-1483, May 1978.