A Software 8-VSB Receiver for ATSC Digital Television

by

Chia Y. Wu

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the degree of

Bachelor of Science in Electrical Engineering and Computer Science and Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

~~May 1999~~
[June 1999]

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
May 21, 1999

Certified by....... [illegible] ....
John V. Guttag
Professor and Head, Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by........ [illegible] ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

A Software 8-VSB Receiver for ATSC Digital Television

by

Chia Y. Wu

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 1999, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Electrical Engineering and Computer Science and Master of
Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents the design and evaluation of a software 8-VSB receiver for ATSC digital television (DTV). Based on an unique software radio architecture, employing wideband digitization and a conventional workstation, this receiver uses novel software algorithms to achieve carrier and timing synchronization.

Carrier and timing synchronization are typically accomplished with analog techniques, thereby comprising the most hardware-dependent stages of the DTV processing. The approach of this thesis enables signal processing solutions that are more portable and flexible, by designing software-friendly algorithms to replace exclusively hardware methods. This, in turn, facilitates the construction of an all-software digital television. As digital television becomes more widespread and popular, future demands for DTV services will diversify, making the flexibility of software an attractive quality.

Currently, the performance of the receiver is limited by the lack of adaptive interference suppression. While an investigation of this topic is beyond the scope of this work, the software techniques developed in this thesis successfully meet the requirements of carrier and timing recovery for ATSC DTV signals.

Thesis Supervisor: John V. Guttag
Title: Professor and Head, Electrical Engineering and Computer Science

Acknowledgments

I would first like to thank my advisor, John Guttag, who provided invaluable guidance and gave me the special opportunity of pursuing this research. I would also like to convey my appreciation to members of the SpectrumWare team, especially Matt Welborn for his patience and DSP knowledge, and John Ankcorn for his resourcefulness and enthusiasm. Our discussions contributed to the completion of this thesis in countless ways.

To my family and dearest friends, my experiences for the past five years would not have been possible without your love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The objective of this thesis moves towards an all-software implementation of ATSC digital television (DTV). Such an implementation offers maximum flexibility and dynamic functionality. A software DTV can be dynamically modified to meet changing environmental conditions, user requirements, or new standards. DTV processing utilizes hardware functions which lack efficient software counterparts, e.g. analog phase-locked loops. Therefore, to implement a software DTV, we must design software-suitable techniques for traditionally hardware-dominant stages.

In particular, the DTV front-end is a digital receiver that receives bits from the channel. To receive DTV data, it must synchronize to the carrier and the symbol timing of the incoming signal. In conventional DTVs, the receiver achieves carrier and timing synchronizations using phase-locked loops. In contrast, this thesis presents the design and evaluation of a software digital receiver for ATSC DTV signals that uses novel software algorithms for carrier and timing synchronization.

1.1 Motivation

Digital television is an emerging innovation that delivers high quality video and audio via bandwidth-driven specifications. Aside from the high-definition pictures and larger aspect ratio, the excitement surrounding DTV stems from the possibilities provided by the coupling of digital compression and transmission. For instance, a digital transmission format allows the multiplexing of two or more television programs on the same channel. Furthermore, using excess bandwidth, a broadcaster can transmit non-television data such as web content or stock reports over the air.

However, there are difficulties with the transition of DTV for broadcasters and consumers alike. The root of the problem lies in the cost of adopting DTV and the lack of standardization. First, incompatible with its analog predecessor, the DTV standard is extensive and complicated. Manufacturing DTV equipment requires numerous first-generation components and production lines. As a result, DTV transmitters as well as receivers are remarkably expensive. In addition, broadcasters need to build transmitting towers throughout the country, while every consumer has to purchase either a set-top box for their analog set or a digital television set.

Secondly, because of the potential variety of transmitted information, it is unclear what the video, audio, interface, or communication requirements will be for a commercial receiver. In fact, no I/O standards presently exist to ensure compatibility between commercial DTVs and cable boxes, digital VCRs, or even PCs. Since receiver functionality cannot be changed once manufactured, current models can quickly become obsolete as new requirements are standardized. The cost of adopting DTV combined with the possibility of early obsoletion hinders the deployment of DTV technology. Thus, despite DTV's enormous long term potential, many forecast a prolonged and arduous transition.

If digital television sets were re-programmable, new functionality could simply be downloaded in order to meet new standards. A software implementation, while readily reprogrammable, could also enhance functionality due to its ability to be adaptive, dynamic, and flexible. Additionally, a software DTV can dynamically adapt to changing environments, by simply loading modules intended for different operating conditions. Lastly, a software DTV can readily incorporate improvements in its submodules (faster video decoder) and new software functions (voice-to-text dictation).

In summary, a software DTV implementation has the following benefits:

- Simplify system upgrades
- Ease integration with peripheral applications

- Adapt dynamically to changing operating environment, and
- Provide interoperability between standards

1.2 The *SpectrumWare* Approach

A software digital television can be realized via the *SpectrumWare* approach [3] to wireless communication and signal processing. SpectrumWare is a novel software radio architecture based on wideband digitization, a general purpose processor, and application-level software. This architecture offers distinctive advantages which can be exploited during the design and implementation of a software digital television.

SpectrumWare aims to create communications systems that use wideband digitization, and then perform all of the digital signal processing in software on a general purpose workstation [3]. Such software communication devices are referred to as *virtual radios* [3].

The analog/digital boundary and the software/hardware boundary are moved as close to the antenna as possible [3]. Since current A/D technology and available processors cannot afford the rate required to directly sample wide RF bands, a multi-band hardware front-end is used to convert the desired RF band to the IF frequency. Then, the analog IF waveform is directly digitized, and the samples are transferred into host memory via a specialized PCI I/O system [3]. All subsequent signal processing is performed in user-level application software. Figure 1-1 shows a block diagram of a virtual radio system.

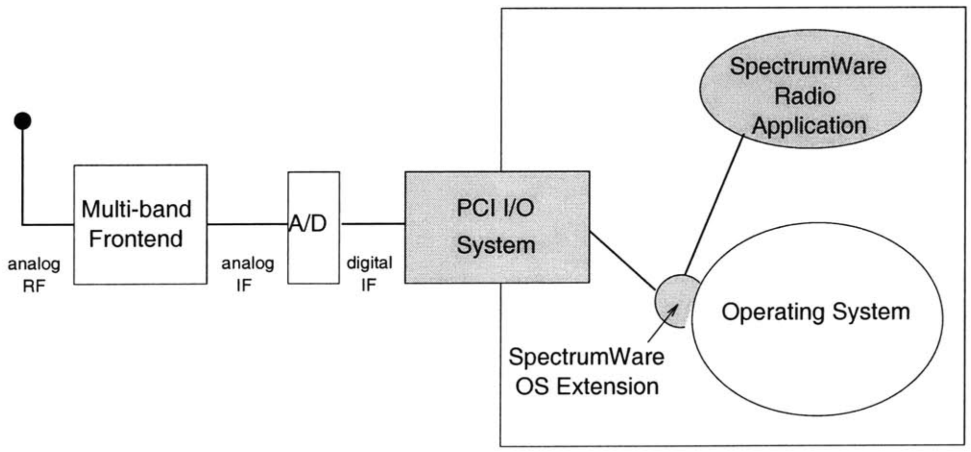

Figure 1-1: Virtual radio block diagram.

While the I/O system provides high-bandwidth, low-latency access to digitized signals, a programming environment, consisting of a library of portable signal processing routines, provides an infrastructure for building virtual radio applications [3].

Using a general purpose workstation to do signal processing can impose a significant cost. In comparison to special-purpose devices, workstations are very expensive and consume more power. However, exploiting the resources available on the workstation offers many potential advantages [13]:

- The programming environments available on a workstation make it easier to develop and experiment with new algorithms,
- The ability to use off-the-shelf components, including workstations, software applications, and programming environments, simplifies development
- Larger memory and computational flexibility can lead to better approaches to signal processing,
- Portable software modules can ride the curve of rapidly improving workstation performance, and
- Using a single platform improve resource utilization, by applying cycles saved in one function to another.

These properties of the SpectrumWare architecture greatly facilitate software-based DTV technology, where the design, implementation, and analysis of software-friendly algorithms are crucial.

In this thesis, the ability to use off-the-shelf components simplifies development. The SpectrumWare architecture, by placing digitized IF samples in host memory, provides data access for off-the-shelf programming environments and applications, such as MatLab. Using SpectrumWare and MatLab I/O features allow MatLab routines to operate on 'real' data, rather than contrived simulation examples. Even though MatLab is not intended for real-time performance, the availability of 'real' data offers a close assessment of algorithm performance and accuracy. In addition, algorithm design can take advantage of the MatLab signal processing toolbox, a library of portable signal processing modules. Verified MatLab designs can be translated to other programming languages for use in other environments.

Also, the large memory and computational flexibility of a general purpose processor permits the implementation of unconventional algorithms. Large memories facilitate the constructions of algorithms that make multiple passes on the data, perform out of order computation, and store waveforms for future use [13]. The memory also allows for temporal de-coupling of the data. With the waveform stored in memory, the timing can be determined by scanning the waveform and determining the appropriate timing information, such as the time at which the transmission commenced. This can greatly simplify the implementation of some systems, and reduce the amount of work required to maintain synchronization [13].

1.3 Thesis Scope

The ATSC (Advanced Television Systems Committee) digital television specification is composed of complex compression and transmission standards (see Figure 1-2). The video compression algorithm conforms to MPEG-2, while the audio compression is achieved with Dolby AC-3. The multiplexor formats the audio and video data into MPEG-2 transport packets, which are then multiplexed into a single data stream for digital broadcasting. This serial bit stream is coded for forward error correction (FEC), which adds redundancy to the data for the ability to detect and correct errors. Following channel coding, the transmitter converts the data into a 6-MHz signal by 8-VSB modulation [1].

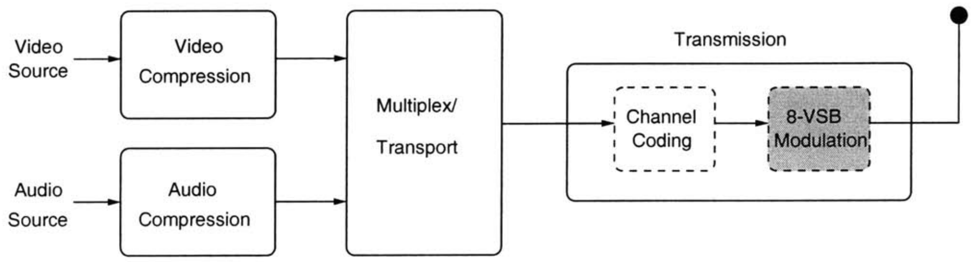

Figure 1-2: DTV transmitter architecture.

The goal of this work is to enable an all-software implementation of an ATSC digital television receiver, which must undo the processing performed at the transmitter. Figure 1-3 shows the corresponding receiver architecture [2]. Fortunately, software solutions for submodules of the receiver are available or straightforward to implement. For instance, compression technology is well-understood, and source code for MPEG-2 and AC-3 decoders can be obtained. The demultiplexor, which examines packet headers to separate audio from video data, is natural to implement in software. Source code for a channel decoder is available as well.

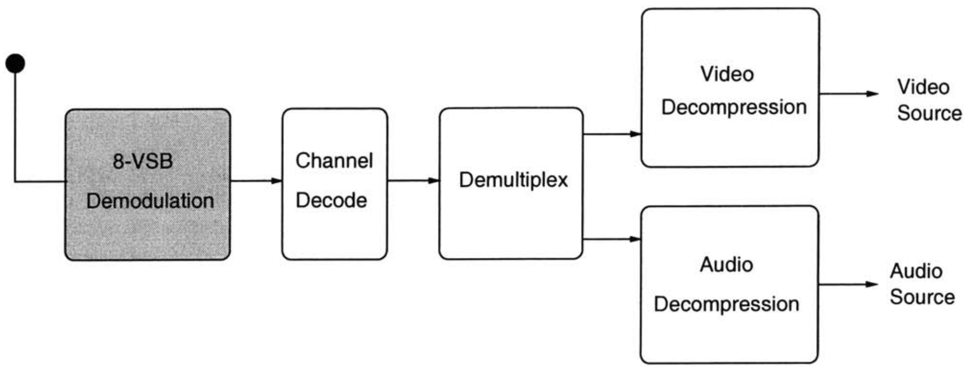

Figure 1-3: DTV receiver architecture.

The 8-VSB demodulation recovers, from the channel, a coded baseband DTV signal in the form of bits. However, stages of its processing are accomplished with specialized hardware functions. Since such functions are not designed with SpectrumWare resources in mind, a direct software translation would be inefficient. To perform this processing in software, more software-suitable techniques must be developed to increase efficiency.

Specifically, 8-VSB demodulation involves a hierarchy of signal processing problems. The first stage is a RF tuner, which can be tuned to DTV channels, that subsequently translates the RF signal to an IF frequency. Next, the IF signal is demodulated to recover the baseband signal. In order to synchronously translate an IF signal to baseband, as discussed in Chapter 3, the demodulation must synchronize to the carrier frequency and phase. This is referred to as *carrier recovery*. Then, the receiver must synchronize to the symbol timing in order to receive baseband symbols at the *optimal* times, also explored in detail in Chapter 3. This process of symbol synchronization is called *timing recovery*. The received symbols often suffer distortion, either from the transmission channel or from hardware imperfections within the receiver. Therefore, an *equalizer* stage is needed to compensate for linear channel distortions. After equalization, a *slicer* quantizes the symbols to produce bits for each symbol.

Figure 1-4 demonstrates the difference between a hardware 8-VSB receiver and a software 8-VSB receiver model based on SpectrumWare. In the hardware implementation, carrier recovery and timing recovery both operate on analog signals prior to digitization. They employ analog phase-locked loops to ensure that the frequency translation and the analog-to-digital sampling occurs synchronously [2]. On the other hand, the SpectrumWare model operates on digitized data, sampled without carrier or timing synchronization. Thus, to allow an all-software DTV approach, it is necessary to design sychnronization algorithms for discrete samples, without compromising accuracy or performance.

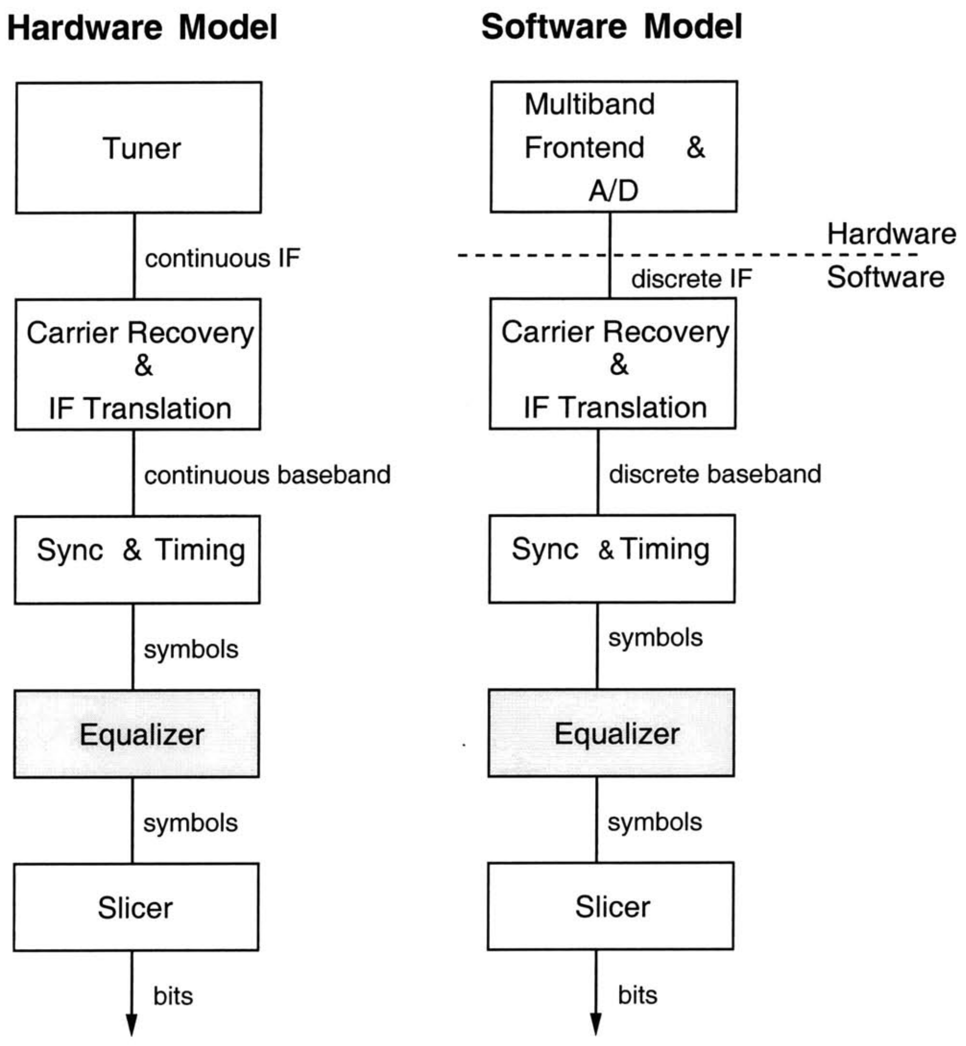

Figure 1-4: Software and hardware models of an 8-VSB receiver

The scope of this thesis does not concern equalization. The design and implementation of a software equalizer, though very involved, can still be directly borrowed from a hardware equalizer without significant re-design. Bypassing equalization means that the receiver cannot compensate for signal distortions. As an alternative, the software 8-VSB receiver will be verified on data collected with minimal distortion (for instance, at the transmitting tower location). Such distortion-free DTV signals approximate signals obtained at remote locations with equalization. Therefore, with a future stage of equalization, the software receiver will achieve the same performance at remote locations as well.

The following chapters describe the design, implementation, and analysis of a software 8-VSB receiver, using novel software algorithms for carrier and timing recovery. Chapter 3 describes the baseband DTV signal and the modulated RF signal in more detail. Then, Chapter 3 examines conventional signal processing tasks employed to achieve carrier and timing synchronization for DTV signals. In Chapter 4, the proposed carrier recovery algorithm is outlined, while Chapter 5 reports the corresponding timing recovery algorithm. Chapter 6 presents the performance of these algorithms as well as that of the overall receiver. Finally, Chapter 7 discusses future work for the software DTV receiver and the contribution of this thesis.

Chapter 2

DTV Transmission Standard

According to the ATSC Digital Television Standard, coded video and audio packets must undergo processing that inserts various synchronization signals. The presence of such signals allows reliable carrier and symbol timing synchronization even in the presence of heavy interference and severe distortion.

To aid carrier recovery, the transmitted RF signal contains an unmodulated component of the sinusoidal carrier, called a *pilot tone*. Without the pilot tone, the receiver would need to extract the carrier frequency and phase information from the data itself. However, the pilot carrier allows the receiver to directly lock onto the carrier frequency and phase, by locking onto the unmodulated pilot. Symbol timing, on the other hand, is solved by the repetitive insertion of special bits or words into the data sequence solely for synchronization purposes. By detecting these periodic messages, the receiver can maintain synchronization independently of the data.

The following sections describe relevant parameters of the baseband DTV signal as well as the modulated RF signal, while paying special attention to synchronization processing and the modulation format.

2.1 Baseband DTV Signal

The channel-coded MPEG-2 data is a serial stream of randomized symbols, where each symbol can be one of eight levels with equal likelihood. Prior to transmission, a multiplexor inserts various synchronization signals into this coded packet stream.

A two-level (binary), 4-symbol *data segment sync* is inserted into the 8-level digital data stream at the beginning of each *data segment.* A complete segment consists of 832 symbols: 4 symbols for data segment sync, and 828 data plus parity symbols. The 828 8-level symbols carry 3 bits per symbol, resulting in 2,484 bits of data. Since symbols are transmitted at a rate of 10.76 Msymbols/sec, the data segment sync occurs regularly at an interval of 77.3 us, and is the only signal repeating at this rate. Unlike the data, the four symbols for the data segment sync are not channel-coded, therefore timing synchronization can take place without decoding. The data segment sync pattern is a 1001 pattern, as shown in Figure 2-1.

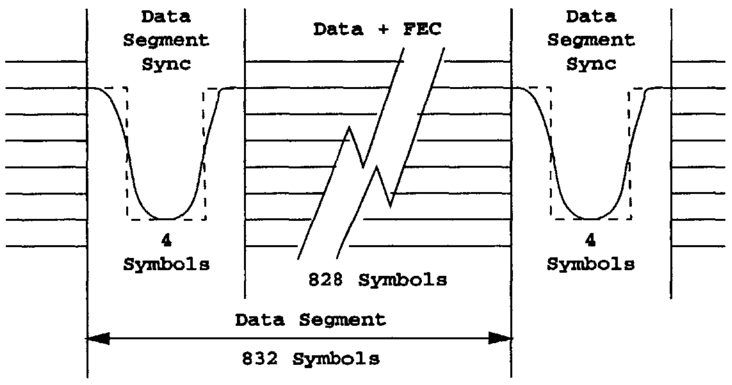

Figure 2-1: VSB data segment.

The data is not only divided into data segments, but also into *data fields*, each consisting of 313 segments. Each data field (24.2 *ms*) starts with one complete data segment of the *data field sync*, as shown in Figure 2-2. Each two-level symbol represents one bit of data. The 832 unencoded symbols in the field sync segment are defined in the ATSC Standard. The presence of field syncs not only permits data field synchronization; the syncs can also be used as a known reference training signal for the receiver equalizer, or as means for direct measurement of received signal conditions e.g. signal-to-noise ratio.

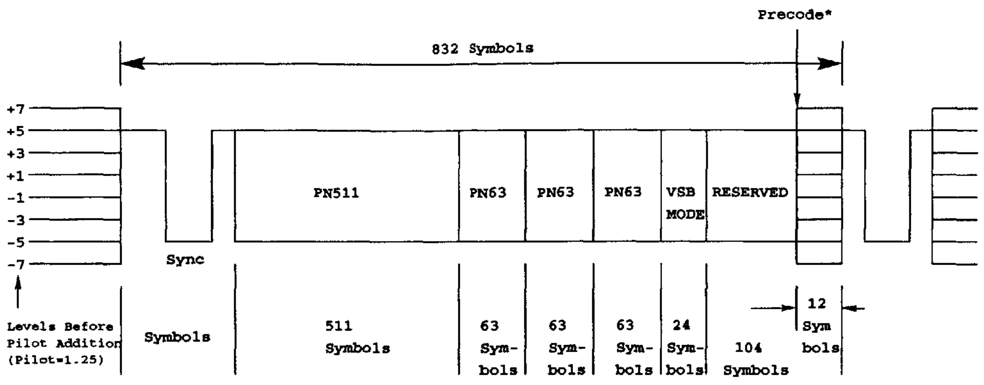

Figure 2-2: VSB data field.

2.2 8-VSB Modulation

8-VSB modulation refers to the digital transmission system developed for ATSC digital television. It employs the Vestigial Sideband (VSB) modulation format, a variant of Pulse Amplitude Modulation (PAM), in which a baseband message signal is used to modulate a single carrier. Traditional PAM spectrum exhibits even symmetry around DC, and therefore, is a *double-sideband* signal because it transmits both the symmetric upper and lower sideband of the message signal. This requires significant bandwidth. To increase bandwidth efficiency, single sideband PAM is often used by removing one of the sidebands. However, this method is ineffective for signals that contain critical information at or near DC. In this case, the upper and the lower sidebands of the signal cannot theoretically be separated [7].

VSB modulation has been developed to overcome the frequency bandwidth of double sideband PAM and the deficiency of a single sideband modulation scheme. As the name implies, it includes a *vestige*, or trace, of the second sideband. The VSB modulation technique transmits the upper sideband and a portion of the lower sideband of the message signal via a carrier frequency. This ensures that relevant DC data is transmitted along with the upper sideband. In digital television, this property allows the transmission of a DC pilot, intended to ease carrier recovery.

In 8-VSB modulation for digital television, the baseband 8-level symbols combined with the binary segment syncs and field syncs undergo the following processing steps to produce a nominal VSB transmission shown in Figure 2-3.

1. *Pilot insertion*

In order to synchronously demodulate a RF signal, a receiver is required to synchronize with the frequency and phase of the sinusoidal carrier. To aid data-independent carrier recovery, a pilot tone is added to the DTV broadcast signal. This pilot tone is generated by adding a small digital DC value (1.25) to every symbol of the digital baseband data plus sync signals.

2. *Lower Sideband Removal*

   The 8-level data plus syncs and DC pilot have two sidebands in the frequency domain, occupying 11.38 MHz. Before transmission, however, most of the lower sideband (all except .31 MHz) is removed. The resulting 6 MHz spectrum is flat, except for the band edges where a nominal square root raised cosine response results in 620 kHz transition regions.

3. *Modulation*

   The 8-level symbols combined with the binary syncs and the DC level are used to modulate a single carrier. After modulation, the DC value causes an unmodulated component of the sinusoidal carrier (pilot) to appear in the data spectrum for transmission.

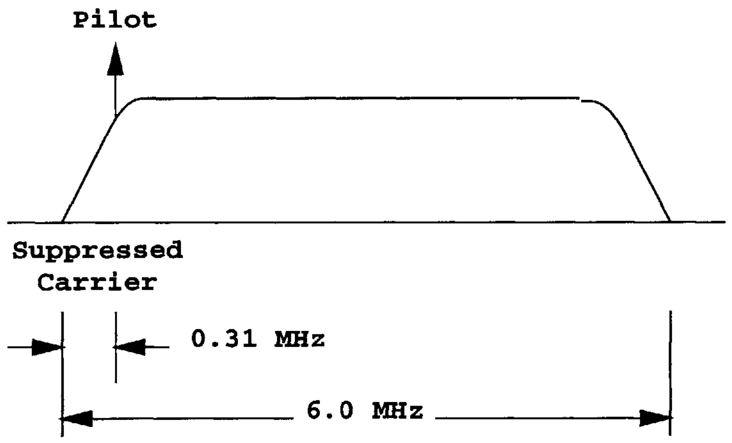

Figure 2-3: VSB channel occupancy.

Note the small pilot at the suppressed-carrier frequency, 310 kHz from the lower band edge. The frequency of the pilot is the same as the suppressed-carrier frequency.

2.3 Table of VSB Characteristics

The parameters characterizing the DTV signal are summarized in the following table.

Parameters	**8-VSB**	**Units**
Channel Bandwidth	6.0	*MHz*
Excess Bandwidth	11.5	*percent*
Symbol Rate	10.762	*Msymbols/sec*
Bandwidth Efficiency	3	*bits/symbol*
Segment Length	832	*symbols*
Segment Sync Duration	4	*symbols*
Field Sync Duty Cycle	1/313	*segments*
Payload Data Rate	28.9	*Mbits/sec*
Pilot Power Contribution	0.3	*dB*

Table 2.1: VSB characteristics

Chapter 3

Background

Before moving on to detailed descriptions of software-friendly algorithms for carrier and timing recovery, a brief review of existing techniques is in order. In this chapter, we first review the concepts of *coherent demodulation* and *symbol timing*, emphasizing the importance of synchronization. This is followed by a qualitative description of *phase-locked loops* (PLL), a common synchronization solution in wireless systems. The introduction to PLL can be applied to the carrier and timing recovery methods recommended in the *Guide to the Use of the ATSC Standard* [2]. Finally, the suitability of such recommended techniques for SpectrumWare is formulated, and the need for alternative approaches is established.

3.1 Coherent Demodulation

At the receiver in a communication system, a information-bearing signal, $a(t)$, is recovered through demodulation. The message $a(t)$ is usually a real-valued, baseband signal. To increase bandwidth-efficiency, $a(t)$ is modified by the modulation scheme to result in a complex baseband signal $B(t)$. This signal is then multiplied by a transmitting sinusoidal carrier, which effectively translates the baseband signal $B(t)$ to a high-frequency passband channel [9].

$$y(t) = B(t)\exp(j\theta)\exp(j2\pi F_c t) \tag{3.1}$$

The frequency F_c is referred to as the *carrier frequency*, and θ is the *carrier phase*. From 3.1, it is clear that $B(t)$ can be recovered, at the receiver, from the modulated signal $y(t)$ by multiplying it with the complex exponential $\exp -(j\theta) \exp -(j2\pi F_c t)$. This is referred to as *coherent demodulation* because it requires the parameters used during modulation.

The receiver is responsible for generating this complex exponential, called a *reference carrier*, therefore it must estimate the parameters F_c and θ. However, inaccuracies in the estimates corrupt the demodulated signal. To demonstrate such distortions, assume that F_c is somehow available at the receiver and $\hat{\theta}$ is an *estimate* of θ. The reference carrier $r(t)$ is expressed in (3.2):

$$r(t) = \exp -(j\hat{\theta}) \exp -(j2\pi F_c t) \tag{3.2}$$

A coherent demodulator translates the passband signal back down to baseband by multiplying $r(t)$ and $y(t)$ together, to produce $z(t)$.

$$z(t) = B(t) \exp(j\theta - j\hat{\theta}) \tag{3.3}$$

When $\theta = \hat{\theta}$, $B(t)$ is received without distortion. Depending on the modulation scheme, the message signal $a(t)$ can be recovered using the particular relationship between $B(t)$ and $a(t)$. When $\theta \neq \hat{\theta}$, $B(t)$ is scaled by $\exp(j\theta - j\hat{\theta})$. The effect of this distortion on the desired message signal $a(t)$ depends on the modulation scheme.

In digital television, the baseband DTV signal plus pilot is the message signal, $a(t)$. The VSB modulation is performed by creating $B(t)$, where $B(t) = a(t) + j\tilde{a}(t)$ and $\tilde{a}(t)$ is related to $a(t)$ by a time-invariant filtering operation which causes a cancellation of a major portion of one of the sidebands. In this case of nearly complete cancellation of a sideband, $\tilde{a}(t)$ is approximately the *Hilbert transform* of $a(t)$, and $a(t) \simeq real[B(t)]$ [9].

Figure 3-1 demonstrates coherent VSB demodulation.

From Figure 3-1, the coherent demodulator output for VSB signals is

$$z_1(t) = \frac{1}{2}a(t)\cos(\theta - \hat{\theta}) - \frac{1}{2}\tilde{a}(t)\sin(\theta - \hat{\theta}) \tag{3.4}$$

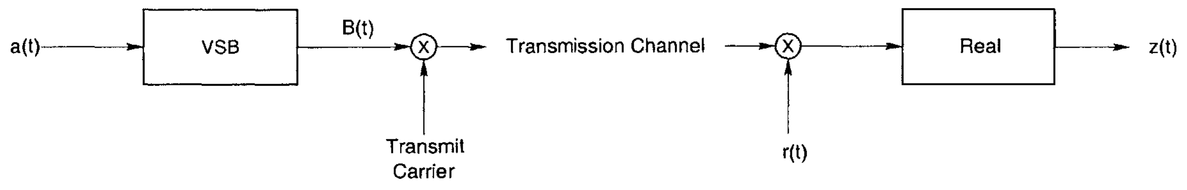

Figure 3-1: Coherent demodulation for VSB.

Equation (3.4) shows that a phase error can produce a severe distortion for VSB modulation. When $\theta = \hat{\theta}$ and thus $z_1(t) = a(t)$, the demodulator properly receives the message signal. Otherwise, the first term reduces the power of $a(t)$ by a factor proportional to $\cos(\theta - \hat{\theta})$. The second term, however, introduces an interference called *quadrature distortion* when $\theta \neq \hat{\theta}$. As $\tilde{a}(t)$ has roughly the same power level as $a(t)$, a relatively small phase error must be maintained to minimize the quadrature distortion [9]. Thus, maintaining close synchronization between θ and $\hat{\theta}$ is crucial.

Equation (3.2) assumes that the carrier frequency F_c is available at the receiver. But in realistic situations, this is often not the case. Therefore we modify the reference carrier frequency to an estimate $\hat{F}_c$. The effects of a residual frequency offset, $F_c - \hat{F}_c$, contributes to a time-varying phase parameter $\theta = \int (F_c - \hat{F}_c)dt = (F_c - \hat{F}_c)t$. The total phase at the receiver is

$$\hat{\theta}_{total}(t) = \int (F_c - \hat{F}_c)dt + \hat{\theta} \tag{3.5}$$

and the phase error is now $\theta - \hat{\theta}_{total}$, where $\hat{\theta}_{total}$ rotates (modulo 2π) at the rate of $F_c - \hat{F}_c$. The overall distortion is still quantified with (3.4), by replacing $\theta - \hat{\theta}$ with $\theta - \hat{\theta}_{total}(t)$, and evaluating $\hat{\theta}_{total}(t)$ for each instant in time.

In summary, carrier frequency and phase synchronization involves the estimation of F_c and θ at the receiver. The process of deriving the carrier frequency and phase from the data-bearing signal is called *carrier recovery*. Once carrier recovery allows coherent demodulation to take place with minimal errors, the receiver must overcome another synchronization problem, in order to extract symbol values from the baseband signal. The following section describes the relevance of symbol timing and establishes the need for timing recovery.

3.2 Symbol Timing

In digital television, an alphabet of 8 finite symbol values is used to multiply a sequence of pulses, occurring at a rate of 10.76 MHz. This converts the discrete symbol values into a continuous signal for transmission, expressed below:

$$Q(t) = \sum_{m=-\infty}^{m=\infty} A_m p(t - mT) \tag{3.6}$$

where A_m is the m^{th} 8-level symbol amplitude, and $p(t)$ is the transmitting pulse. The objective of a receiver, therefore, is to extract the discrete amplitudes A_m, for $m = -\infty\ to\ \infty$, from a continuous signal of pulses.

Sampling (3.6) at interval $T = \frac{1}{10.76MHz}$ yields

$$Q_k = A_k p(0) + \sum_{m \neq k} A_m p(kT - mT) \tag{3.7}$$

Each sampled result Q_k contains not only the k^{th} symbol amplitude, but a second term summing contributions from all other non-k^{th} symbols. This term is called the *intersymbol interference* (ISI). If $p(t)$ crosses zero at non-zero multiples of T,

$$p(kT) = \delta_k \tag{3.8}$$

for all integers k, then output of the sampler is

$$Q_k = Q(kT) = \sum_{m=-\infty}^{m=\infty} A_m \delta_{k-m} \tag{3.9}$$

In this case there is no ISI [4]. Figure 3-2 shows that neighboring symbols do not interfere with one another at the optimal sampling times 0 and T.

The ATSC Standard defines the DTV transmitting pulse to satisfy the criterion of (3.8), called the *Nyquist criterion.* To extract ISI-free symbol amplitudes from the received signal, a receiver must perform *timing recovery* by synchronizing to the correct sampling instants.

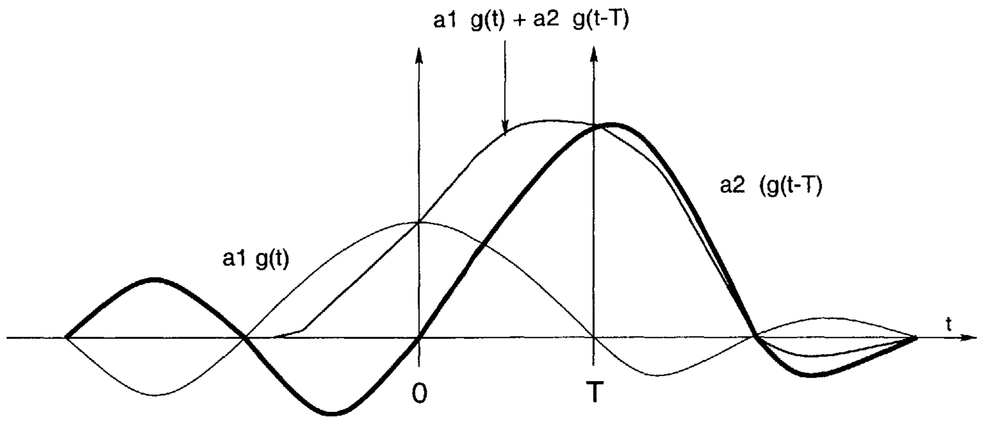

Figure 3-2: Effects of ISI.

For pulses meeting the Nyquist criterion, the optimal sampling point occurs at the center of a symbol pulse, where ISI is minimal.

Assuming that there is a constant sampling-phase offset, τ, such that the sampling does not occur at 0 and T of Figure 3-2, but at τ and T + τ. In this case, the extracted amplitudes do not correspond to the transmitted symbol amplitudes since they suffer from ISI. Timing recovery is also perceived as a process through which the sampling phase error τ is reduced to zero.

3.3 Phase-locked Loops

Underlying most synchronization techniques is the *phase-locked loop* (PLL). A PLL takes a noisy, nominally periodic input signal, and generates a clean periodic output by synchronizing to the input frequency and phase. PLLs find wide application in areas such as communications and wireless systems. This section provides a qualitative description of PLLs, intended to lend insights to the ATSC recommendations in sections 3.4 and 3.5.

3.3.1 Topology

The basic PLL structure is demonstrated in Figure 3-3.

The PLL is a very basic feedback control system consisting of a *phase detector* (PD), a *loop filter,* and a *voltage-controlled oscillator* (VCO). The VCO varies the frequency of $v(t)$ in

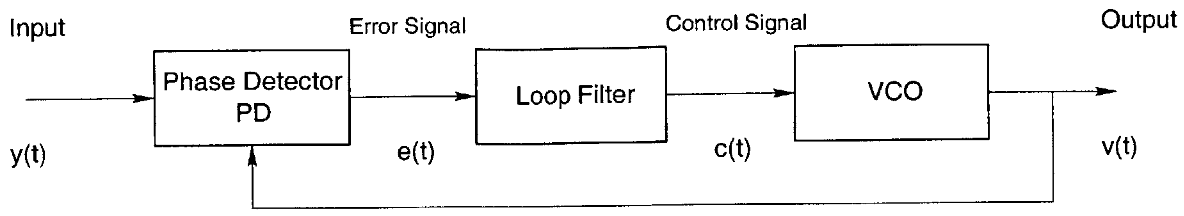

Figure 3-3: PLL topology.

response to a *control voltage* $c(t)$. Ideally, the objective is for $v(t)$ to track the phase of the input $y(t)$. The PD measures the phase error between the input $y(t)$ and the VCO output $v(t)$. The resulting error signal can be filtered to become a control signal that drives the VCO. This basic configuration shows that an error signal is fed back to the system input to drive the error term to zero at steady-state. When the error is zero, $v(t)$ is synchronized to $y(t)$.

Any synchronization structure employing a PLL operates in one of two modes: *acquisition* mode and *tracking* mode. Acquisition mode refers to a transient state during which the error signal varies with time. Once the error converges below some pre-defined limit, the PLL is said to be phase-locked or in *tracking mode.* Since frequency is simply the time derivative of the phase, a phase-locked PLL is also frequency-locked. The transient response of a PLL is generally a nonlinear process that cannot be formulated easily [14]. Therefore, in Section 3.3.2, we resort to a purely qualitative presentation of design and performance issues for PLLs.

3.3.2 Performance

The performance of a PLL is evaluated by the following operating characteristics:

- *Acquisition Time*

  The amount of time it takes for the value of the error signal to converge, or to reach steady-state,

- *Accuracy*

  The reliability of the steady-state VCO output phase, measured by the time variance of this phase,

- *Capture Range*

  The range of input frequencies over which an initially unlocked PLL can acquire lock, and

- *Noise Performance*

  Threshold signal-to-noise condition above which a PLL can operate properly.

These characteristics of a PLL are determined by the following design parameters: its *loop bandwidth* and *loop order.* The loop bandwidth affects the noise performance, while the order controls the accuracy of the PLL. The performance implications of these two parameters necessitate a number of compromises to be considered in the design of a PLL. For this thesis, a qualitative summary, rather than the complete mathematical derivations of such tradeoffs, will suffice.

First, an arbitrarily low or narrow loop bandwidth allows the PLL to function in an arbitrarily low signal-to-noise ratio environment. Doing so, however, risks incurring a potentially long acquisition time [14]. Hence, there is a fundamental tradeoff between noise performance and acquisition time. A narrow loop bandwidth is desirable for noise immunity, but clearly it cannot be arbitrarily narrow if reasonable acquisition time is a necessity [14].

Second, low-order PLLs offer maximum accuracy or stability. However, the narrow capture range of such simple, optimally stable PLLs is often inadequate for the large frequency uncertainty in practical applications. Higher-order loops can improve frequency capture range, at the price of accuracy. While theoretically possible, they are often difficult to realize in practice [15]. Therefore, most practical PLLs incorporate additional techniques, with wider capture range, to aid the acquisition of frequency [15].

One such technique, derived from PLL principles, is a *frequency-locked loop* (FLL). Unlike the PLL, it contains a *frequency difference detector* (FD) in place of the phase detector. The carrier recovery structure proposed by ATSC uses a combined frequency-and-phase-locked loop (FPLL), similar to the one illustrated in Figure 3-4.

During acquisition under the condition of a large frequency uncertainty, the PD produces a phase error, oscillating at a high frequency. LPF of such high-frequency errors results in a small, DC value. The frequency error will start out large but with a strong low-frequency component. The additively combined error (and VCO control) is dominated by

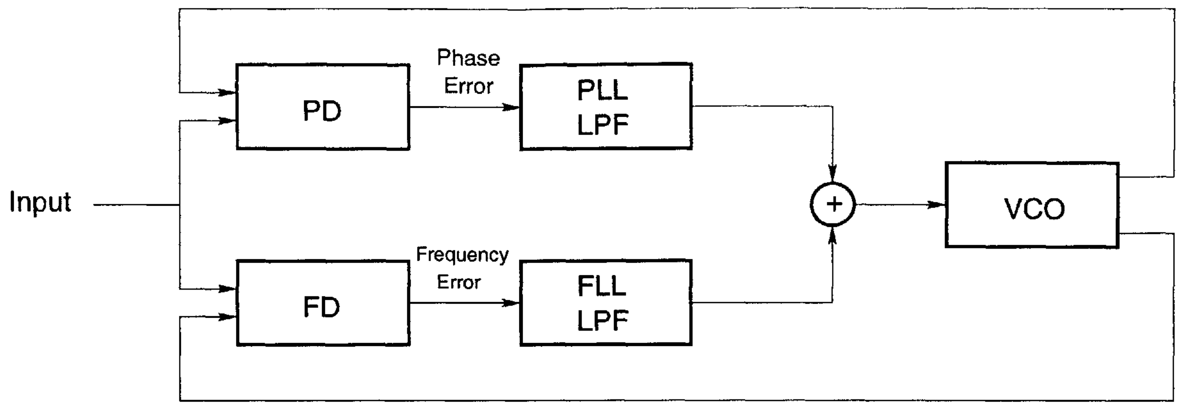

Figure 3-4: Combined PLL and FLL.

the frequency error during frequency acquisition. But the frequency error converges to zero very rapidly. As the frequency error gets smaller, the phase error starts to dominate, causing the VCO to be locked in phase as well.

3.3.3 Applications

In conclusion, the design of a PLL is dominated with choosing the right loop order and loop bandwidth to meet the needs of the specific application. In carrier recovery, the objective is to track the phase of the carrier on the input signal as closely as possible, while at the same time minimizing the effect of noise. Transmission channels can introduce significant fluctuations in the carrier frequency phase, to properly demodulate a RF signal, these fluctuations should be replicated on the reference carrier used by the receiver [4]. In timing recovery, by contrast, the symbol timing phase is unaffected by channel effects. Therefore, the objective is to generate a stable single-frequency tone. The frequency of this tone should equal the average symbol rate of the input, but transient variations in the symbol rate should be ignored, as should noise or other interference [4]. Proper carrier or timing synchronization structures must be designed to satisfy these requirements.

3.4 ATSC Recommended Carrier Recovery

The *Guide to the Use of the ATSC Digital Television Standard* [2] describes the characteristics and design considerations for receivers of the digital television system. It is designed as a tutorial, intended to serve as a guideline and literature reference for receiver manufacturers. It also illustrates design choices, based on system implementations used for laboratory tests [2]. This section presents the carrier recovery technique recommended by this guide.

Recall from Chapter 2 that a pilot signal is inserted in the DTV signal for carrier synchronization. When the receiver reference carrier is synchronized in frequency, the demodulated pilot is simply a constant DC level. Otherwise, it is a low-frequency sinusoid whose frequency represents the offset from the desired demodulating frequency. When the demodulation is phase-locked, the DC pilot has a known amplitude and a known polarity, as specified in Chapter 2.

To perform coherent demodulation, carrier recovery is performed on the pilot carrier by a FPLL (frequency-and-phase-locked loop). The randomized DTV data is not used at all in the carrier recovery process. A block diagram of the FPLL is shown in 3-5.

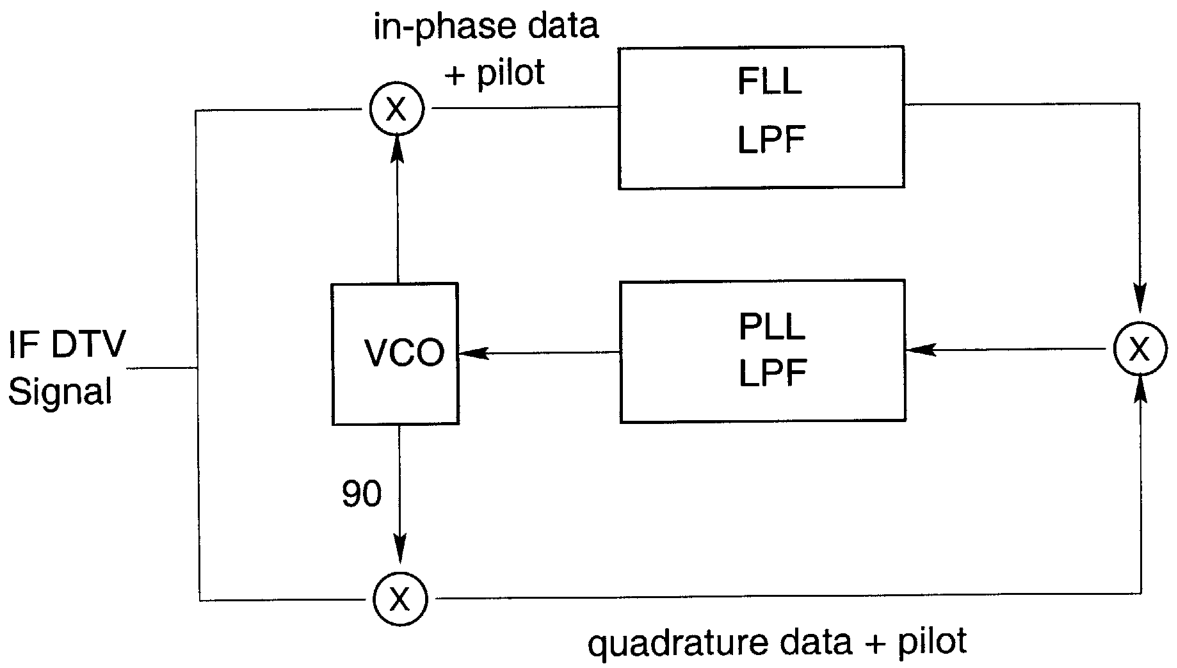

Figure 3-5: ATSC recommended carrier recovery.

Prior to frequency and phase lock, the IF DTV signal is demodulated with the arbitrary frequency and phase of the VCO, as shown in 3-5. Demodulating with zero additional phase recovers the baseband DTV signal plus pilot (called the *in-phase* signal), and demodulating with a 90-degree phase produces the *quadrature* signal. Due to VSB modulation, the quadrature signal is the *Hilbert transform* [9] of the in-phase DTV signal and pilot. Under perfect frequency and phase synchronization, the in-phase pilot is a DC level with a known sign and a known amplitude, while the quadrature pilot is just *zero*. We will derive this important property in Chapter 4. The ATSC carrier recovery technique, in fact, achieves phase-synchronization by reducing the quadrature pilot to zero.

In the absence of frequency synchronization, the demodulated pilot is low-frequency sinusoid, oscillating at the frequency difference between the incoming carrier frequency and the VCO output. In essence, the pilot signal is itself a frequency *error* signal that is fed back to the VCO. Thus, by isolating the pilot with a low-pass filter (LPF), the frequency difference detector (FD) is eliminated.

During frequency acquisition, the FLL LPF isolates the in-phase pilot, subsequently used to multiply the quadrature DTV data. The PLL LPF rejects the quadrature high frequency data to recover the pilot, using it to control the VCO frequency to track the incoming pilot frequency. Prior to frequency lock, the PLL LPF (and VCO control) is dominated by the contribution from the frequency difference.

Once the FLL establishes lock, the output of the FLL LPF is simply a DC level. The multiplier only scales the quadrature data with this DC value, therefore, the FLL is, in a sense, deactivated. The PLL LPF isolates the quadrature pilot, also a DC value, from the high-frequency quadrature data. The VCO is directly adjusted using this value, in order to reduce the quadrature pilot to zero. A zero quadrature pilot satisfies the condition for phase-locking, and the VCO output is thus synchronized in phase with the incoming carrier.

3.5 ATSC Recommended Timing Recovery

In timing recovery, the objective is to localize the center of the symbol interval (ISI-free instants) for sampling. The repetitive 4-symbol data segment syncs, described in Chapter 2, are present solely for the purpose of timing recovery. A PLL, with a narrow loop bandwidth, can be used to synchronize a reference sampling clock with the incoming syncs. Unlike carrier recovery, additional frequency acquisition is not needed since the symbol frequency is relatively stable.

The ATSC guide recommends the timing structure of Figure 3-6.

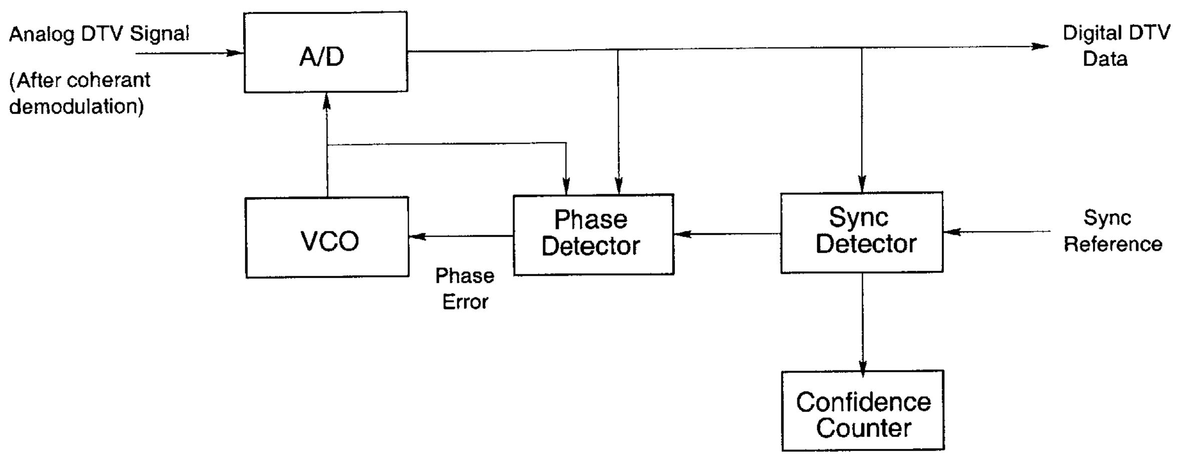

Figure 3-6: ATSC recommended timing recovery.

This technique requires three major components: an analog-to-digital converter, a PLL generating the sampling clock for the A/D, and a *sync detection* module which selectively activates the PLL.

Initially, with the PLL unlocked or free-running, the A/D converter samples the analog baseband data at the symbol rate of 10.76 MHz. The sync detection structure correlates periodically the incoming data with a reference sync pattern at the sync repetition rate. A *confidence counter* is incremented by one for each high correlation result. Upon reaching a pre-defined level of confidence, as a result of consecutive high correlations, the segment sync is detected from the data. Due to the random nature of the data and the periodic nature of the syncs, this topology can reliably extract the segment syncs even in the presence of distortions.

The phase detector (PD), otherwise suspended, is activated only during sync times by the segment sync detector. The PD first finds the point of symmetry in the incoming sync sequence (1001), corresponding to a boundary between two symbols. This boundary is half of a symbol interval away from the optimal sampling time. Then, the PD produces an error voltage proportional to the phase difference between the VCO output and the timing derived from the sync. The VCO is adjusted until the proper sampling time is reached. The output of the VCO is a stable, single-toned sampling clock, undisturbed by transient symbol rate jitters.

3.6 Suitability for SpectrumWare

The SpectrumWare approach moves all signal processing functions into application software, on a general-purpose processor. The ATSC guide provides the carrier and timing recovery algorithms, but they cannot be readily implemented in SpectrumWare for a number of reasons.

First, the carrier recovery and timing recovery structures both operate on analog signals. However, a SpectrumWare implementation is based on wideband digitization, where all samples available for processing have been sampled and digitized at a high rate without symbol timing synchronization. Therefore, the ATSC techniques must be adapted for digital processing, or alternate discrete-time techniques need to be proposed.

Second, the ATSC recommendations are targeted toward hardware implementations, even though design considerations differ vastly between hardware and software. For example, minimizing the circuitry is often a priority in hardware design. These recommendations are not designed to take advantage of the larger memory capacity in a general-purpose computer, nor the innate flexibility of a software implementation.

Finally, a software implementation lags behind specialized hardware in speed; a direct translation of hardware techniques to software is often inefficient. The ATSC algorithms are *closed-loop* in nature, since a PLL feeds an error signal back to the input. As a result of this feedback path, subsequent signals cannot be processed or synchronized until the PLL reaches steady-state. The time it takes for the error to reach steady-state (the acquisition time) depends on the *precision* of this error. A low-precision error estimate can result in a prolonged transient response, while a high-precision error will converge quickly. In hardware solutions, low-precision is generally the case, because it minimizes the error detection circuitry. The acquisition time of the same low-resolution technique, implemented in software, can be more than an order-of-magnitude longer. This limits the ability of the software synchronization module to track transient variations on its input, and that can be unacceptable to the application requirements.

There are several options for amortizing the cost of software. First, we can increase the precision of the error estimate and, therefore, have fewer corrections via the feedback loop and faster acquisition time. Alternatively, the feedback path may be completely removed by resorting to *open-loop* schemes. Open-loop phase and frequency estimation schemes are completely different from classical closed-loop schemes, which, as noted, uses error feedback control to minimize the error in the received signal. Open-loop schemes are generally *feedforward* techniques, in that the input signal parameter is actively estimated and then treated as a statistic for a detection algorithm [14]. As demonstrated in Chapter 4 and 5, both of these methods are employed in this thesis.

Chapter 4

Carrier Recovery

The ATSC carrier recovery method achieves phase synchronization by locking the incoming pilot signal using a phase-locked loop. In this chapter, we derive mathematically how a phase error impacts the pilot, revealing opportunities for exploiting workstation memory or computational capability. Based on the application requirements of carrier recovery, we then formulate a feasible software-oriented approach for carrier synchronization, intended to amortize the cost of software. To enable this approach, signal processing algorithms must be devised to fulfill the particular requirements of this approach. This chapter concludes with descriptions of digital signal processing methods for software-based carrier recovery.

4.1 Pilot Phase

Section 3.4 presented a qualitative relationship between the demodulating phase error and the received pilot signal. This relation can be expressed mathematically, by recognizing that an *in-phase* component is simply the real part of a complex signal, while a *quadrature* component corresponds to the imaginary part.

Section 3.1 showed that a real-valued message baseband signal $a(t)$, is converted by the modulation format to a complex-valued signal $B(t)$ for transmission. For VSB modulation, $B(t) = a(t) + j\tilde{a}(t)$, where $\tilde{a}(t)$ is the Hilbert transform of $a(t)$ and $a(t) = real[B(t)]$. Applying these expressions to the pilot signal gives $a(t) = \mathrm{C}$, a constant that represents the specified positive pilot level. For a DC constant, $\tilde{a}(t) = 0$, and therefore, $B(t) = C$. The

VSB modulation has no effect on the pilot.

Section 3.1 also showed that performing frequency translation with a phase error will recover a distorted baseband signal, $B(t)\exp(j\theta - j\hat{\theta})$. For the pilot signal, this demodulator output is $C\exp(j\theta - j\hat{\theta})$. The relationship between the phase error and the received pilot signal, P_r, is expressed below:

$$P_r = C\exp(j\theta - j\hat{\theta}) \tag{4.1}$$

The in-phase part or the real part of (4.1) is

$$P_{rI} = C cos(\theta - \hat{\theta}) \tag{4.2}$$

and the quadrature (imaginary) part is

$$P_{rQ} = C sin(\theta - \hat{\theta}) \tag{4.3}$$

Clearly, the phase error $(\theta - \hat{\theta})$ can be directly computed by:

$$\theta - \hat{\theta} = arctan(\frac{P_{rQ}(t)}{P_{rI}(t)}) \tag{4.4}$$

When the demodulation is phase-synchronized i.e. when $\theta = \hat{\theta}$, $P_{rI} = C$ while $P_{rQ} = 0$. Evidently, seeing that $P_{rQ} = 0$ is the condition for phase-locking, the ATSC recommended carrier recovery technique achieves phase-locking by isolating the pilot with a low-pass filter, then using a PLL to drive the quadrature pilot to zero.

In the presence of a frequency offset (but no constant phase error), the demodulator produces a time-varying pilot

$$P_r(t) = C\exp(j2\pi(F_c - \hat{F}_c)t) \tag{4.5}$$

where F_c is the modulating carrier frequency and $\hat{F}_c$ is the demodulating frequency. $P_r(t)$

is thus a sinusoid with the frequency of $(F_c - \hat{F}_c)$. The ATSC specification on frequency recovery employs a frequency-locked loop to reduce the pilot frequency to zero, at which point $P_r(t)$ would be a DC constant. This discussion of phase error lends itself to a feasible phase recovery approach in the following section.

4.2 Phase Recovery

The purpose of carrier recovery is to minimize distortions due to frequency and phase errors during coherent demodulation. Unlike mobile channels, television channels do not introduce significant carrier frequency shifts. The distortion mainly come from fluctuations in the carrier phase. Therefore, a carrier synchronization structure must replicate any phase fluctuations on the reference carrier used for coherent demodulation [4]. It must have the ability to *closely* track the phase but to *loosely* track the frequency. This section addresses a software-suitable technique for phase-locking, and a software frequency-locking technique in the next section.

Assuming frequency synchronization has been accomplished, the design of this phase-locking technique is dominated by two factors. First, this application requires close tracking of transient phase variations, and therefore, fast acquisition time. However, a closed-loop solution, operating by differentially reducing a detected error, often requires more computations than its open-loop counterpart. Thus, in keeping with the goal of minimizing computation for fast acquisition time, we focus on an open-loop design.

Second, the preferred way of phase-tracking is to isolate the pilot with a low-pass filter, and then estimate the phase error from the pilot itself. This filtering operation is very expensive in software. The current implementation of the SpectrumWare approach, based on wideband digitization, digitizes data at 33 MHz. Narrowband digital filtering of a near-DC component, from a wideband spectrum, is costly. A conservative estimate puts the number of taps needed to implement a single stage digital filter at 2,000. In other words, producing one sample of the pilot signal requires 2,000 multiplications and 2,000 additions, obviously not affordable in software to support close-tracking of the carrier phase.

Therefore, it is desirable to isolate the phase without the unaffordable filtering operation. This is possible, in fact, due to the nature of the baseband DTV signal. A baseband DTV signal has 8-level symbols, randomized so that symbol value occurrence at each level is equiprobable. The signal, therefore, is zero-mean. Once the pilot is inserted by adding a DC constant to the DTV signal, the mean value of the signal is displaced from zero to the pilot DC level.

A phase-synchronization structure, based on averaging the signal to resolve the pilot level, is shown in Figure 4-1.

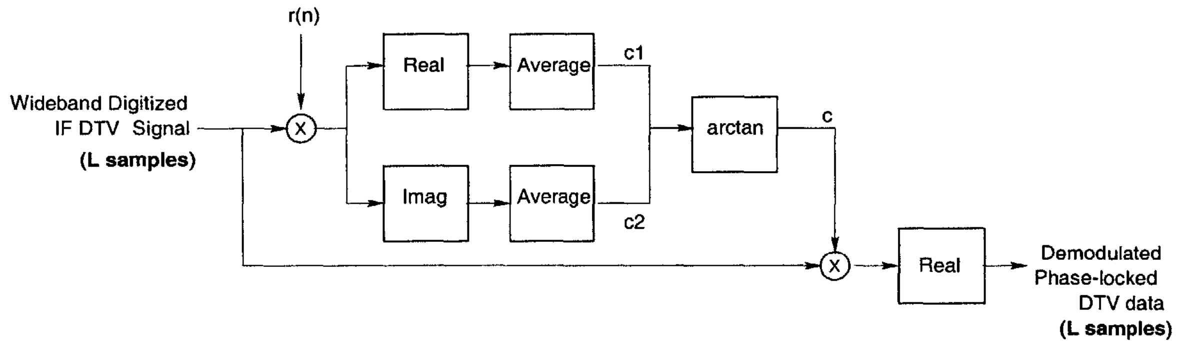
r(n)
Wideband Digitized IF DTV Signal (L samples)
Real
Average
c1
Imag
Average
c2
arctan
c
Real
Demodulated Phase-locked DTV data (L samples)

Figure 4-1: Steps for SpectrumWare phase recovery.

The incoming IF DTV (and pilot) signal is first divided into blocks of L samples. Assuming that each block of samples is then demodulated (or frequency-translated) by a fixed reference sinusoidal carrier, $r(n)$, that has the proper carrier frequency and an arbitrary phase, $r(n)$ can be expressed as $\exp(j\hat{\theta})\exp(j2\pi F_c n)$. When the real part of the demodulated signal, or its in-phase component, is averaged, the result indicates the in-phase pilot level at $c1 = Ccos(\theta - \hat{\theta})$. Taking the imaginary part of the signal, in parallel with the real part, produces its quadrature component, whose mean value represents the quadrature pilot level at $c2 = Csin(\theta - \hat{\theta})$. The demodulating phase offset, $\theta - \hat{\theta}$, as demonstrated in Section 4.1, is calculated by $arctan(\frac{c2}{c1})$. c, generated to correct this phase error, equals $exp(-(j\theta - j\hat{\theta}))$. Applying this phase correction recovers an undistorted $B(t)$, the real part of which is the original message signal $a(t)$. The in-phase, baseband DTV signal, retrieved by taking the real part of the phase-corrected $B(t)$, is delivered to other modules for subsequent processing.

This phase-recovery structure is an open-loop solution with a feedforward path. The latency associated with phase synchronization, unlike a PLL, does not reside in a feedback path. Subsequent samples can be processed in a 'pipelined' fashion before the final phase-correction is complete. A close-loop PLL, in contrast, must stabilize before subsequent samples can be processed. In addition, the computations involved in this technique is a dramatic reduction over filtering. The amount of work required to calculate L samples of phase-corrected DTV signal includes the following steps:

1. L 'real-part-of' operations
2. L adds to generate c1
3. L 'imaginary-part-of' operations
4. L adds to generate c2
5. 1 divide and 1 arctan
6. L complex multiplies

On an average, the work mandated to produce 1 sample of the phase-correct data includes 2 adds, 1 'real-part-of', 1 'imaginary-part-of', $\frac{1}{L}$ division, $\frac{1}{L}$ arctan, and one complex multiplication. This is a significant improvement over the standard approach of low-pass filtering, which needs approximately 2,000 multiplications and 2,000 additions alone to produce just 1 sample of the pilot, without any phase-correction. Using relatively inexpensive computations can support continuous processing, without having to drop data along the way. This feedforward algorithm performs processing on blocks of data, rather than on a sample-by-sample basis. The large memory afforded by software running on a general-purpose processor can easily facilitate such block-based data processing.

The relevant parameter L must be determined with a delicate balance. To accurately compute the pilot, L must be sufficiently long in order for the DTV signal samples to converge, or average out, to the actual pilot level. However, it must also be adequately short, in order to provide the temporal resolution for tracking a rapidly-varying phase parameter.

This compromise can also be characterized in the frequency domain. The operation of averaging L samples is essentially a low-pass filtering operation, where the filter has L taps and all the taps are 1. Since the averaging is performed on non-overlapping blocks of size L, the averaging result can be considered as the output of an equivalent low-pass filtering but is further decimated by L. Hence, if L is the number of taps of a filter, it must be high enough to provide sharp cutoff characteristics to pass low-frequency information while filtering out high frequency noise. However, it must be sufficiently short so as to not over-decimate the output and lose temporal resolution.

This phase-synchronization architecture, due to its ability of close-tracking, can tolerate a limited frequency offset by correcting for phase errors caused by the offset. This is only possible provided that the frequency difference is sufficiently small, such that the time-varying phase remains relatively constant within the averaging window of L samples. In other words, a window of L samples should be equivalent to a very small percentage of the period of this frequency difference.

Assume that within a window of L samples, the phase variations due to a frequency offset is limited to at most 0.01 radians between the first sample and the last sample. This constraint says the window of averaging, L, must correspond to $\frac{0.01}{2\pi} = 0.16$ *percent* of a period of the frequency offset. This requirement is quite strict even for practical applications. Under this condition, the SpectrumWare phase-recovery technique can tolerate a certain frequency offset, whose relationship to L is determined by Equation 4.6.

$$F_{offset} = \frac{0.01\ radians}{2\pi\ radians} \frac{33,000,000}{L}\ Hz \tag{4.6}$$

The following table presents the frequency offsets for some values of L, based on a 0.01 radian constraint.

L	F_{offset}
1,500	35.0141
1,000	52.5211
750	70.0282
500	105.0423
400	131.3028
samples	*Hz*

Table 4.1: F_{offset} tolerance vs. L

Evidently, using a small L provides not only finer temporal resolution for phase-tracking, but it tolerates a wider range of frequency offsets. Exploiting the frequency stability of television transmissions, we conclude that as long as an efficient, and high-precision frequency estimation technique is used for an initial estimate, the slow and limited frequency fluctuations can be well-compensated by this mechanism for phase-locking. The next section presents a design for an effective frequency estimation method, using the pilot signal.

4.3 Frequency Recovery

Without frequency acquisition, a reference demodulation is performed using a nominal frequency. The demodulated signal will have a pilot signal whose frequency is the offset between the reference carrier frequency and the carrier frequency. This difference is typically within 100 kHz [1]. The phase-locking technique of Section 4.2 cannot sufficiently compensate for such a large frequency difference. Thus, an independent frequency estimation stage is required to resolve the pilot frequency. The demodulating frequency is subsequently adjusted in order to reduce the pilot frequency to zero. When a zero-frequency pilot (a DC value) is achieved, the condition for frequency synchronization is satisfied.

For frequency synchronization, the pilot signal needs to be separated from the DTV data using a 100 kHz low-pass digital filter. While it is a significant improvement over isolating a DC value with low-pass filtering, 100-kHz low pass filtering from a 33-MHz band is far too computationally intensive for continuous tracking of frequency. Thus, we only rely on this processing to obtain an *initial* estimate of the frequency offset. The low-computation phase-tracking structure of Section 4.2 is responsible for compensating any transient frequency drifts. However, since its ability is limited to tracking small frequency errors, the initial estimate of frequency must be a *high precision* estimate. This minimizes the demodulating frequency error to within the acceptable range for the phase recovery technique.

Figure 4-2 formulates a frequency recovery approach, using a closed-loop solution to achieve high precision. A feedback loop allows multiple passes on the data; thus, the frequency error can be incrementally reduced until a pre-defined level of precision is reached. Once a satisfactory frequency estimate is resolved, the feedback path deactivates. The computations associated with the feedback are therefore amortized over time. Any small transient frequency variations are compensated for by the phase-synchronization loop.

Before proceeding to estimating the pilot frequency, the pilot samples are first *decimated.* Decimation, by decreasing the sampling rate of a signal, offers several advantages. The pilot is a low-frequency signal (< 100 kHz) over-sampled at 33 MHz. These samples not only capture the periodicity of the incoming pilot, but they capture any undesirable high-frequency

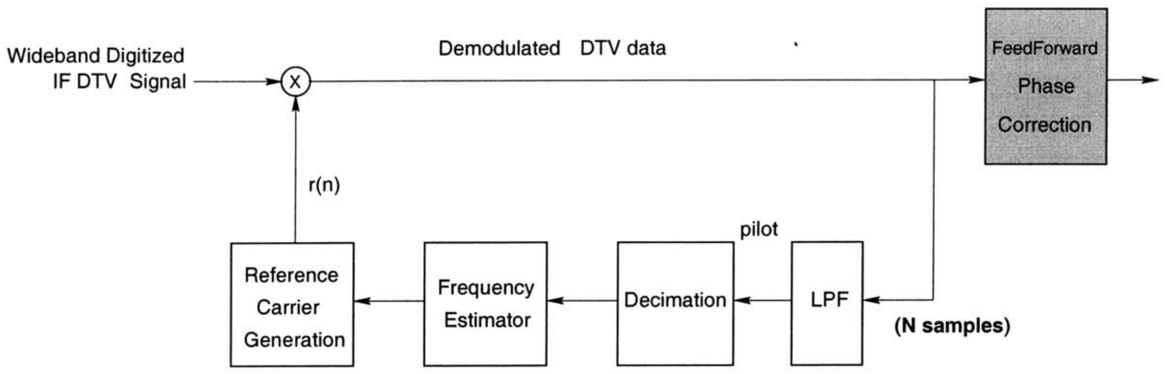
Wideband Digitized IF DTV Signal
X
Demodulated DTV data
FeedForward Phase Correction
r(n)
pilot
Reference Carrier Generation
Frequency Estimator
Decimation
LPF
(N samples)

Figure 4-2: Steps for SpectrumWare frequency recovery.

noise or interferences as well. Decimation improves the noise immunity of a *frequency estimator*, whose performance can degrade due to a low signal-to-noise ratio. It can also improve the accuracy of the estimator result, since for a fixed number of samples, samples with a lower sampling rate span a longer window in time. This provides the estimator with multiple sets of information regarding the periodicity, allowing a more accurate result.

It is possible to design an open-loop solution, where the input is always demodulated with a reference frequency, and the error is compensated downstream via a feedforward path. However, the precision of such feedforward approaches is limited, since it cannot accommodate algorithms that make multiple passes at the data.

The following section discusses the signal processing involved in the frequency estimator in Figure 4-2. The input to the estimator is a decimated sinusoid, embedded in strong presence of noise. In order to extract the frequency of this sinusoid, we resort to *Prony's method* for the implementation of a frequency estimator.

4.4 Frequency Estimatior

4.4.1 Prony's Method

Estimating the frequencies of a set of complex sinusoids in the presence of additive noise is a classical signal processing problem. The *discrete Fourier transform* (DFT), the classical method of estimating the frequency content of signals, is limited in resolution. Thus, when a high-precision estimate is essential, high-resolution techniques based on *linear prediction* are preferred.

Linear prediction is a prediction of the signal by a linear combination of past samples of that signal, plus a noise contribution. *Prony's method*, dating back to 1795, is a linear prediction technique specifically designed for modeling sampled data as a linear combination of exponentials. The original Prony's method seeks to fit a *deterministic* exponential model to the data,

$$x(n) = \sum_{k=1}^{p} \exp(j2\pi f_k n) \tag{4.7}$$

where $n = 0, 1, ...N$. $x(n)$ is the signal, p is the number of sinusoids, and f_k is the desired frequency for each sinusoid. Estimating the pilot involves the estimation of only one sinusoid, thus $p = 1$, and (4.7) becomes:

$$x(n) = \exp(j2\pi f_1 n) \tag{4.8}$$

If a method can be found to determine f_k from the data, all samples of the signal $x(n)$ can be predicted and are thus known. Prony's contribution is the discovery of such a method, which constitutes forming the following polynomial:

$$\phi = \sum_{m=0}^{m=p} a(m)(\exp(j2\pi f_1))^{p-m} \tag{4.9}$$

Simplifying (4.9)with $p = 1$, $\phi = a(0)[\exp(j2\pi f_1) + a(1)]$. By definition, $a(0)$ is normalized to 1. Multiplying $a(m)$ and (4.7) together, and performing more polynomial forming and simplifying, the following matrix equation is formed:

$$\begin{bmatrix} x(p+1) \\ x(p+2) \\ \cdots \\ x(2p) \end{bmatrix} = - \begin{bmatrix} x(p) & x(p-1) & \dots & x(1) \\ x(p+1) & x(p) & \dots & x(2) \\ \cdots & \cdots & \cdots & \cdots \\ x(2p-1) & x(2p-2) & \dots & x(p) \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \\ \cdots \\ a(p) \end{bmatrix}$$

This demonstrates the 'linear prediction' aspect of Prony's method, where, based on linear combinations of past samples, future signal values can be predicted. $a(m)$ is referred to as the *linear prediction coefficients* of the signal $x(n)$, while p is called the *prediction order* of the linear prediction. Simplifying the matrix equation with $p = 1$,

$$x(2) = -x(1)a(1) \tag{4.10}$$

or, more generally, $x(n) = -x(n-1)a(1)$. Once a(1) is determined from samples of $x(n)$, the estimated frequency f_1 can be solved from the polynomial ϕ of (4.9). Note that Prony's method requires *a priori* knowledge of p. In addition to knowing the number of complex sinusoids in the signal, Prony's method uses exactly $2p$ data samples to fit p exponentials.

The original Prony's method does not perform well in the presence of significant additive noise. In fact, the frequency estimation error has been shown to be linearly proportional to the noise in the data [12]. A great deal of research effort has been dedicated to improving the noise performance of Prony's method. The *least squares* extension of Prony's method retains the use of the assumed number of exponentials but provides a way of using more than the $2p$ number of data samples, in order to reduce the noise sensitivity [12]. The *modified least squares* Prony's method uses a larger prediction order p that helps to model and account for the noise [11]. The downside of this approach is that there are more candidate frequency values than the hypothesized number of sinusoids. Thus, methods are required to eliminate these ambiguities.

The latest innovation in Prony's method, by Tufts and Fiore, is demonstrated in detail in [12]. Based on modified least squares Prony's method, this extension involves a computationally simple and effective algorithm for going from a single linear prediction coefficient to an unambiguous frequency estimate. For ambiguity resolution, two estimations are performed with two relatively prime orders of linear prediction. Since a high prediction order (larger than the actual number of sinusoids) is intended to model the noise as exponentials, using two relatively prime orders means the noise will be modeled by different numbers of sinusoids, therefore producing different estimation results. The frequency corresponding to a true sinusoid, by contrast, remains consistent across estimates of varying orders. This particular method was developed to deliver high computational efficiency and superior noise performance, making it suitable for our application.

4.4.2 Enhanced Frequency Estimator

Clearly, the performance of a frequency estimator is affected by its ability to operate with changing signal-to-noise conditions. Thus, it is also possible to enhance estimator results by eliminating sources of input noise. This section proposes a simple iterative procedure, where a preliminary estimate is performed on noisy data, and the estimation result is used to 'clean up' the input data for successive estimations. By compromising acquisition time, this procedure further enables a high-precision, high resolution estimate.

Recall that the pilot is isolated via a wideband low pass filter to allow wideband frequency captures. However, since the pilot occupies only a single frequency, the result of a 100-kHz LPF passes other high-frequency data as additional noise on the pilot. This noise can be reduced by narrowband filtering near the actual frequency of the pilot.

A frequency estimator, enhanced using a feedback loop, is shown in Figure 4-3.

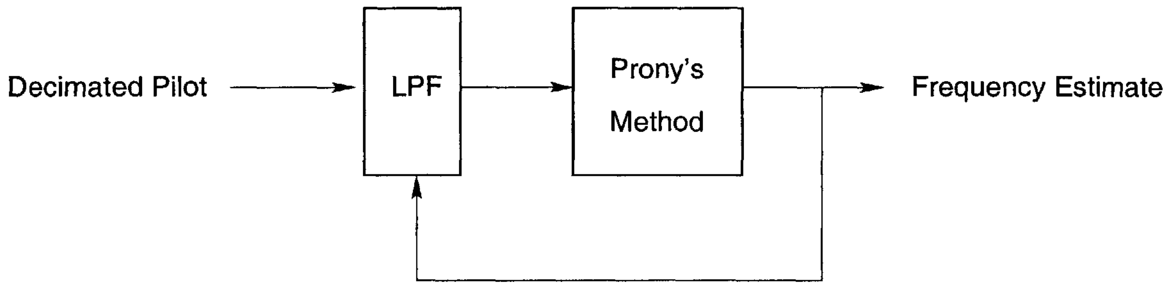

Figure 4-3: Enhanced frequency estimator.

It performs the following tasks:

1. perform preliminary estimate, using Prony's method
2. using estimate result to adapt LPF characteristics, and filter pilot data again
3. return to 1 to refine estimate with 'cleaned' data

This technique makes multiple passes through the data, whose signal-to-noise ratio is increased for a high-precision estimation during each pass. The feedback loop is terminated upon reaching limits defined by system requirements. This can be a ceiling on acquisition time or a desired precision on estimation results.

The overall frequency recovery structure actually consists of two feedback loops. The inner loop of the frequency estimator refines noise conditions for optimal estimation, while the outer loop incrementally reduces any residual estimation error. Even though the high acquisition time associated with software feedback loops makes them impractical for continuous tracking, they are preferred in frequency recovery for obtaining a high-precision initial result.

Chapter 5

Timing Recovery

The closed-loop ATSC recommendation for symbol timing recovery is feasible for a software implementation. Unlike carrier recovery, its feedback path is only active during data segment syncs, constituting less than 0.5 percent of the time. Therefore, this approach is mostly an open-loop one, without frequent delays incurred by a feedback path. A direct software implementation of this solution can support continuous operation. Since SpectrumWare samples are arbitrarily digitized without synchronization, this chapter is dedicated to a discretized-version of the ATSC timing recovery solution.

The process of acquiring synchronization, maintaining synchronization, as well as the final symbol *slicing* are shown in Figure 5-1.

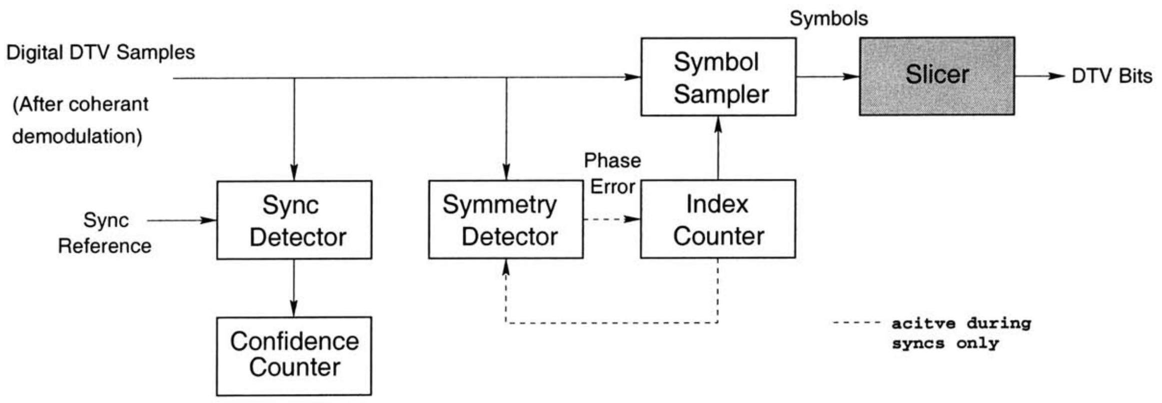

Figure 5-1: SpectrumWare timing recovery.

The signal processing modules in Figure 5-1 are presented in the following sections, while dedicating special attention to any discretizing of traditionally analog techniques.

5.1 Sync Acquisition

The objective of sync acquisition is to locate the data segment syncs from the randomized data samples. Once a sampled baseband DTV signal is obtained from coherent demodulation, the receiver can begin looking for the data segment syncs. Initially, the sync detector periodically correlates the incoming data with a reference sync pattern at the sync repetition rate. The 4-symbol data segment sync occurs once every 832 symbols, at a symbol rate of 10.76 MHz. After sampling at a fixed frequency F_s, each symbol corresponds to $\frac{F_s}{10.76\ MHz} samples$. Each sync spans $4 \times \frac{F_s}{10.76\ MHz} samples$, and it occurs once every $832 \times \frac{F_s}{10.76\ MHz} samples$. Using these figures, the sync detector regularly correlates the data with a reference sync. If a particular 4-symbol data sequence is indeed similar to a sync, the correlation result will appear as shown in Figure 5-2.

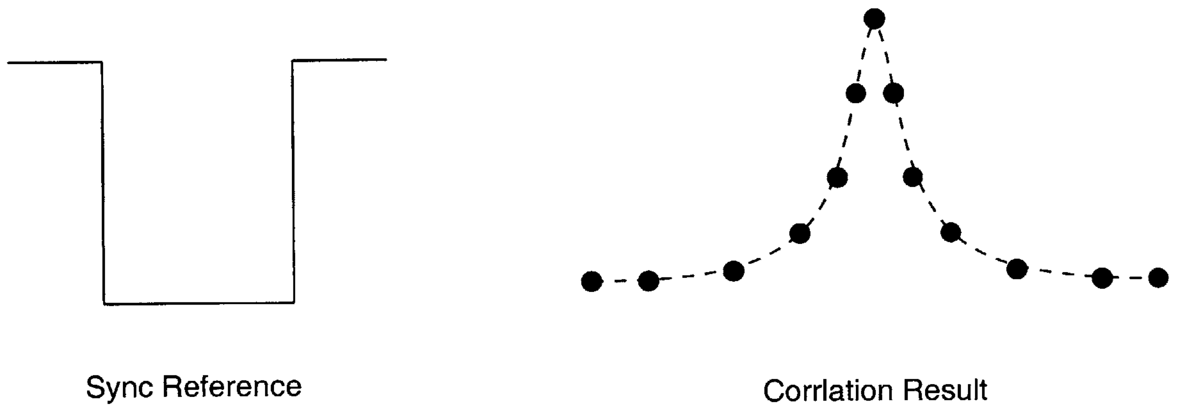

Figure 5-2: Sync correlation.

A confidence counter is incremented by one for each high correlation outcome. Upon reaching a pre-defined level of confidence, as a result of consecutive high correlations, the segment sync is formally detected from the data.

Since the segment syncs happen at a known periodic rate, once a segment sync is detected without error, the receiver can count the repetition period to the next sync, and use it for re-synchronization. Using the segment syncs, the following section describes a discretized procedure for the continuous tracking of symbol timing synchronization.

5.2 Sync Tracking

In the continuous-time domain, the goal of timing recovery is the construction of a symbol-rate clock, synchronized to sample the signal at optimal ISI-free instants. In SpectrumWare, however, the sampling frequency F_s will have an arbitrary relationship to the symbol rate. In addition, the samples available for processing are extracted without synchronization. In this discrete-time sampled system, the modified objective becomes the extraction of samples that are *nearest* to the optimal sampling times, since optimal symbol amplitudes might not have been captured by the sampling. A symbol rate clock is implemented using an *index counter*, while synchronization is performed on the segment syncs by a *symmetry detector.*

5.2.1 Index Counter

A symbol rate clock is constructed by means of a *high-precision* counter, whose value is an index into the incoming sample stream. In a general-purpose system, the sampling frequency F_s is related arbitrarily to the symbol rate; therefore, a symbol interval can be a fractional number of samples. The high-precision counter, intended to track at the symbol rate, is advanced by $\frac{F_s}{10.76\ MHz}$ samples per symbol. *Rounding* the high precision index obtains the integral value nearest to the high-precision index. Assume that a method is available to isolate the optimal sampling time (center of a symbol) from the incoming samples. Once the high-precision counter initializes to a symbol center location, it tracks subsequent symbol centers by incrementing with a fractional, high precision symbol interval. Therefore, the counter maintains the 'true' locations of symbol centers over time. Rounding the high-precision index returns the integral index of the *closest sample* to the symbol center. This sample is thus extracted and quantized to a finite level by the slicer.

Obviously, rounding introduces timing errors. By rounding, the integral index is at most 0.5 of a sample period away from the actual symbol center. As the sampling frequency F_s increases, this error is reduced by a decreasing sample interval. Table 5.1 demonstrates this timing error in response to a varying sampling frequency, both in seconds and in a percentage of a symbol interval.

The sampling frequency (33 MHz) of the current SpectrumWare implementation is limited by current A/D technology and available processor speeds. Thus, in order to reduce the

F_s	**Timing Error**	**Symbol Interval**
33	$1.51e^{-8}$	16.3
66	$7.58e^{-9}$	8.15
99	$5.05e^{-9}$	5.43
165	$3.03e^{-9}$	3.26
231	$2.16e^{-9}$	2.32
330	$1.51e^{-9}$	1.63
MHz	seconds	percent

Table 5.1: Decreasing error with increasing sampling rate

timing error to an acceptable margin, bandlimited interpolation is employed to increase the sampling rate. As long as the timing error is sufficiently small, such that the extracted symbol amplitude is still quantized to the same bit level as the ideal symbol amplitude, this error does not affect receiver performance. As A/D technology and processor speeds improve, such interpolation will not be needed.

5.2.2 Symmetry Detector

The index counter is capable of extracting approximate symbol amplitudes, as long as the high-precision counter contains the location of a 'true' symbol center in the sample stream. The objective of a symmetry detector is to periodically synchronize the high-precision counter value with a known symbol center.

The symmetry detector, otherwise suspended, is activated only during sync times by the index counter. The data segment syncs have a symmetric pattern, with the point of symmetry corresponding to a boundary between symbols. A symbol center is found by first identifying the symbol boundary, and then by shifting the boundary by half of a symbol interval.

The symmetry detector is simply a digital filter with its response and output shown in Figure 5-3.

Discrete points make up the output of the symmetry filter, while the zero-crossing of the filter output identifies the ideal symmetry point. By locating the sample before the zero-crossing (A) and the sample after the zero-crossing (B), the zero-crossing can be found by assuming that A, B, and the zero-crossing lie on a straight line [10]. Using linear

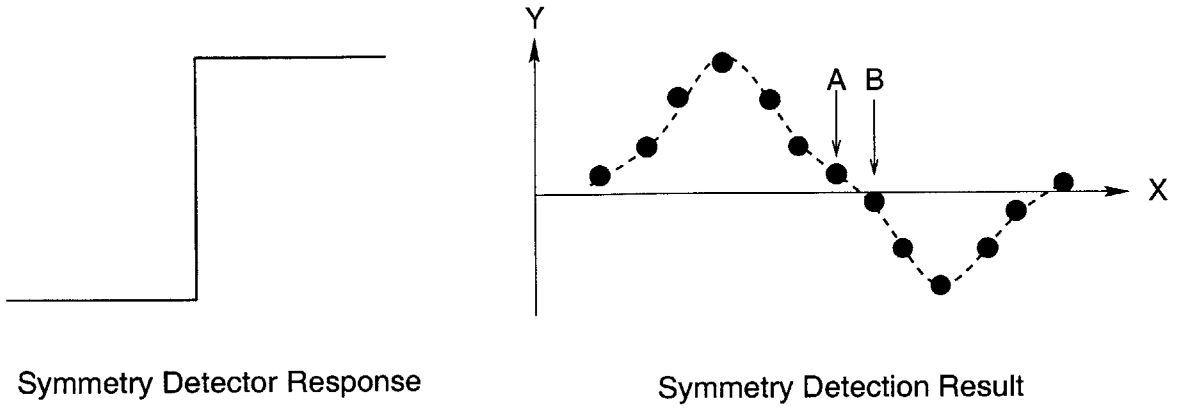

Figure 5-3: Symmetry detector.

interpolation, the zero-crossing location X_{zero} is computed via the following expression:

$$\frac{Y_B}{X_B - X_{zero}} = \frac{Y_B - Y_A}{X_B - X_A} \tag{5.1}$$

From X_{zero}, the precise time index at which symmetry occurs can be easily calculated. The optimal symbol center is half of a symbol interval away from this index, and the high-precision counter is updated with the symbol center location found via symmetry detection.

5.3 Slicer

A slicer is a nonlinear system whose purpose is to transform varying symbol amplitudes into one of a finite set of prescribed values. Each value is then mapped to a binary code, and the data is thus transformed into bits.

The slicer accepts symbols produced by the index counters, and quantize them to one of 7, 5, 3, 1, -1, -3, -5, and -7. Therefore, it must determine 9 uniformly spaced decision boundaries, shown in Figure 5-4 and labeled as -8, -6, -4, -2, 0, 2, 4, 6, and 8.

The nine decision boundaries are determined by the amplitude of the segment syncs, which are transmitted at the known levels of 5 and -5. Once +5 and -5 are known, the decision boundaries can be linearly extrapolated. Finally, each quantized symbol is then mapped to three bits, according to Table 5.2, and digital DTV data has been received from a television channel.

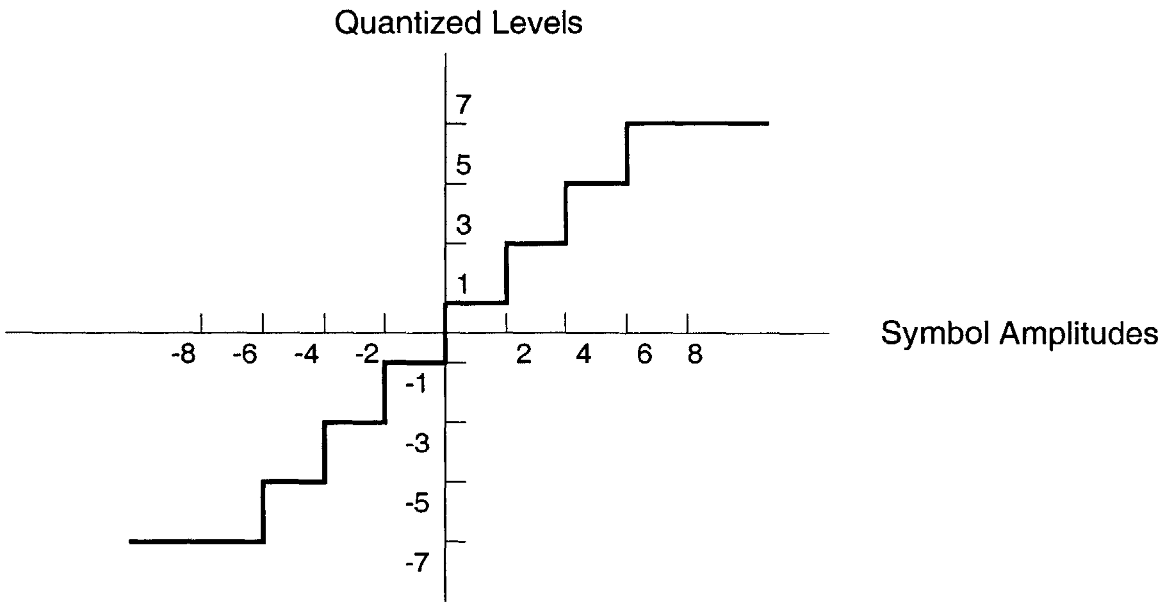

Figure 5-4: Slicer decision region.

Level	**Bits**
+7	111
+5	110
+3	101
+1	100
-1	011
-3	010
-5	001
-7	000

Table 5.2: Symbol-to-bits mapper

Chapter 6

Performance

This chapter presents the performance of the software-based algorithms considered in this thesis and the performance of a digital receiver, implemented in MatLab, using those algorithms. Exploiting the ability of SpectrumWare to digitize *real* DTV signals, the results in this chapter, unless otherwise noted, are obtained with actual DTV broadcasts. This allows us to assess the performance of the algorithms and that of the receiver in an operating environment with realistic noise conditions. A brief analysis of receiver performance leads us to conclude that, though the algorithms function accurately, a more robust receiver design is needed.

6.1 Phase Recovery Performance

The accuracy of the phase recovery algorithm is assessed by its ability to correct for a time-varying phase error. A complex sinusoid with a frequency of F_c produces a phase error that rotates from 0 to 2π radians, at the rate F_c. The phase recovery is thus verified, via simulation, in the following manner:

1. generate a discrete complex sinusoid with a known frequency F_c and a sampling rate of 33 MHz,
2. supply the sinusoid as an input to the phase recovery algorithm with L = 400, and
3. compute from the output any residual phase error

The real part of the complex sinusoidal input is a cosine function, while the imaginary part is a sine. When the phase is properly recovered, the real part of the complex output is a DC value while its imaginary part is simply zero. The residual phase error can be directly computed from the output samples $y(n)$, using $\theta_{error}(n) = arctan(\frac{imag[y(n)]}{real[y(n)]})$. Figure 6-1 demonstrates the output residual phase error, in response to a 40-Hz complex sinusoid.

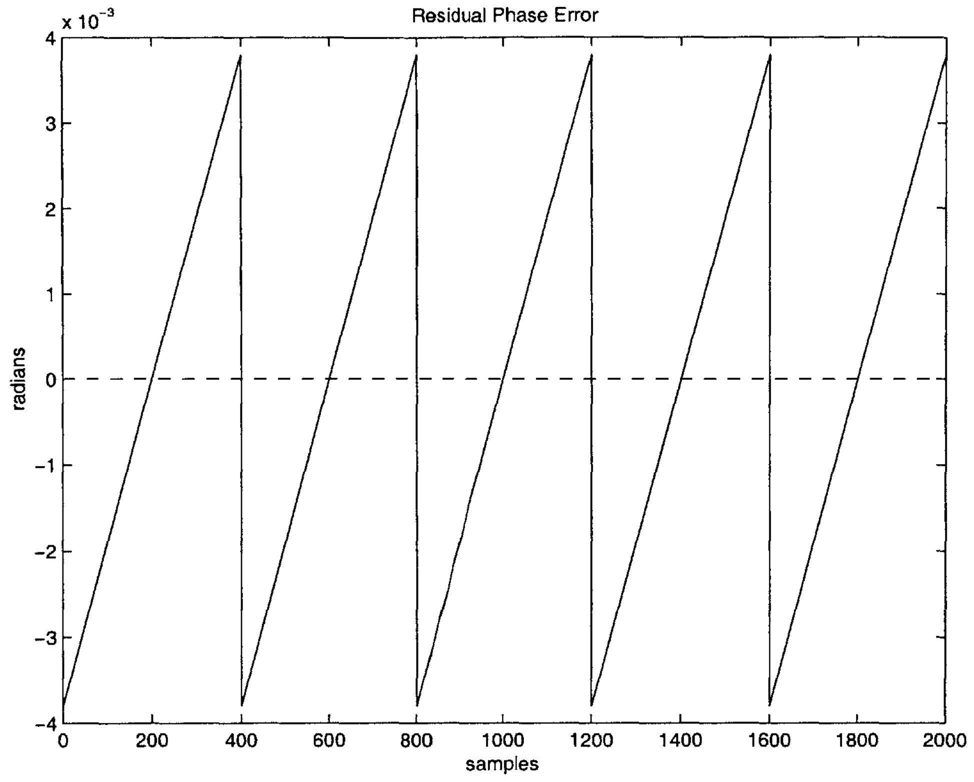

Figure 6-1: Residual phase error.

This figure illustrates that, for a block of L samples, the residual error is smallest near the center of the block, and linearly diverges towards the edges of the block. This result is not unexpected. The phase of a complex sinusoid increases linearly, while the slope of the increase is determined by the its frequency. The phase-correction structure averages the phase over the entire block, which returns the phase at $\frac{L}{2}$. This average phase, however, is used to correct the entire block, therefore the result of this correction is more accurate near $\frac{L}{2}$, and less accurate near the edges.

This leads us to conclude that smaller phase errors can be obtained by averaging over N samples, where N is larger than L. The average phase error (computed over N samples) is then used to correct L samples, centered around $\frac{N}{2}$. This is equivalent to low-pass filtering with a lower decimation rate, producing superior phase results but requiring more computations. For each phase-corrected sample, we need approximately $\frac{N}{L}$ times more computations. This technique is effective for compensating linearly varying phase errors due to frequency changes. For random or irregular phase error, however, it actually compromises temporal resolution relative to using the larger block N. Thus, we remain with the original method for the implementation of a digital receiver.

As long as the phase error at the edges remains within a narrow range, this residual linear error is tolerable. In fact, to minimize the quadrature distortion from VSB demodulation (described in Chapter 3), the phase error must not exceed the range of ± 0.032 radians [9]. Table 6.1 displays the residual phase error range in response to different values of F_c.

F_c	**Phase Error Range**
40	± 0.0015
50	± 0.0057
100	± 0.0037
150	± 0.0057
200	± 0.0075
300	± 0.0113
400	± 0.0152
500	± 0.0189
800	± 0.0304
1000	± 0.0379
Hz	radians

Table 6.1: Phase recovery performance.

Clearly, the phase synchronization structure can properly recover the phase in the presence of a frequency offset, up to 800 Hz.

6.2 Frequency Recovery Performance

The frequency estimator is verified on *real* DTV signals, since it is crucial to assess its performance under real-life interferences. The frequency recovery technique, consisting of 2 feedback loops, operates on a set of 8 DTV signals. The outer loop, responsible for reducing any residual frequency error, terminates after 4 iterations. The inner loop, intended to refine noise conditions for optimal frequency estimations, terminates after 2 iterations.

Table 6.2 shows the result of estimating the pilot frequency during each iteration of each loop. Clearly, with each iteration, the error is incrementally reduced. When the demodulation is frequency-locked, the pilot frequency should be zero. The *last* iteration result, showing a close-to-zero frequency estimate, is used for coherent demodulation.

	Outer1		**Outer2**		**Outer3**		**Outer4**	
	Inner1	**Inner2**	**Inner1**	**Inner2**	**Inner1**	**Inner2**	**Inner1**	**Inner2**
1	-903.73	-905.80	2.0588	0.0514	2.0074	$-4.11e^{-6}$	2.0074	$-2.68e^{-6}$
2	-883.89	-883.76	-0.1412	0.1128	-0.2540	$-9.842e^{-3}$	-0.2540	$-2.12e^{-7}$
3	-890.39	-890.08	0.3256	-0.0569	-0.2687	$-3.02e^{-6}$	-0.2687	$-1.73e^{-7}$
4	-886.21	-885.50	-0.7312	0.0255	-0.7657	$-1.55e^{-6}$	-0.7567	$1.25e^{-7}$
5	-901.47	-902.96	1.4849	-0.1617	1.6464	$-4.31e^{-5}$	1.6466	$-5.06e^{-6}$
6	-885.99	-885.58	-0.4325	-0.0230	-0.4095	$-2.57e^{-6}$	-0.4095	$-7.84e^{-7}$
7	-640.74	-888.31	247.58	0.0430	247.54	$-3.78e^{-4}$	247.54	$-3.76e^{-4}$
8	-898.81	-898.31	-0.5076	-0.0817	-0.5844	$-6.42e^{-6}$	-0.5893	$-8.78e^{-8}$
	Hz	Hz	Hz	Hz	Hz	Hz	Hz	Hz

Table 6.2: Frequency recovery performance.

Since the timing recovery algorithm is strictly a discretized-version of the ATSC recommendation, and the ATSC DTV signal was designed to ensure extremely reliable timing recovery, we choose to omit a direct evaluation of the algorithm. The following section directly presents the performance of a digital receiver as well as its analysis.

6.3 Digital Receiver Performance

The software-oriented carrier recovery and timing recovery techniques, described in this work, are integrated to form a basic digital receiver. The receiver is evaluated by its ability to receive symbols, quantified by the *probability of error*. The probability of error is defined as the probability that an incorrect data symbol is substituted for the correct one.

We resort to the data field sync of the DTV signal to compute the probability of error. The field sync, transmitted once every 313 segments, contains a 4-symbol segment sync followed by a *known* sequence of 511 symbols. For every demodulated and quantized data segment, the receiver compares the 515 known symbols of the field sync to symbols of the received segment.

The probability of error is calculated by how many symbols of the field syncs are received properly. Since a symbol can be one of 8 levels with equal likelihood, a typical random data segment will have $\frac{1}{8}$ of its symbols ($\simeq$ 65 symbols) that match the field sync. If a particular segment has, for instance, at least 100 symbols matching the field sync, and it also occurs once every 313 data segments, it is identified as the field sync with high likelihood. Once a field sync is identified, the number of correctly received symbols as well as the probability of error are estimated. Furthermore, it is reasonable to assume that the probability of error while receiving the known field sync also applies to unknown data. Therefore, this probability of error represents a reasonable assessment of overall receiver performance.

The following table reports the number of properly received symbols out of the known 515, as well as the probability of error it constitutes.

Signal	**Symbols**	P_{error}
1	338	0.34
2	189	0.63
3	234	0.54
4	345	0.33
5	238	0.64
6	245	0.52
7	221	0.57
8	238	0.53
mean	256	0.50

Table 6.3: Digital receiver performance.

Taking a close look at this table leads to a number of observations. The probability of error, ranging from 30 to 60 percent, is too high for acceptable receiver performance. In addition, it also exhibits large variations from signal to signal. Earlier sections of this chapter verified that the frequency recovery technique is able to reduce any frequency error to virtually zero, while the efficient phase recovery technique compensates for phase errors as well transient frequency fluctuations. However, despite the accuracy of these techniques, the probability of error remains high, along with a substantial variance.

To investigate why the overall receiver is unable to deliver satisfactory performance, we must re-examine the decision to exclude an adaptive equalization stage. In Chapter 1, we stated that such an equalization stage is necessary to compensate for distortions on the received signal, often as a result of channel effects or hardware imperfections. Without equalizing for such distortions, the received pulses will generally not meet the Nyquist criterion, and ISI will be present even at proper sampling times. The receiver design in this thesis assumes that any signal distortion is primarily a consequence of transmission channel effects. Thus, by verifying the receiver on data collected near the transmitter location, channel distortion is minimized and reasonable receiver performance is expected, even without an equalizer.

Judging from the results presented in this section, this assumption is clearly insufficient. The poor receiver performance confirms that the transmitter and receiver hardware have an equally strong impact on the signal, but the exact nature of this impact is difficult to quantify. The transmitter might be specifically built for receivers equipped with equalizers. In addition, the multi-band frontend of the SpectrumWare hardware, designed to extract a narrow channel, might not be adequate for wideband DTV reception. Since it is difficult to localize the source of the interference, we can attempt to enhance receiver performance with the adaptive capability of an equalization stage. This can lead to a more definitive assessment of how much distortion is actually present. If an equalizer fails to improve the error rate, a close examination of SpectrumWare hardware might be in order.

In this chapter, our results suggest that good carrier and timing recovery techniques alone cannot make an adequate receiver for a software DTV. Under the current receiver design, the probability of error is high due to interferences and ISI, while its variance remains large since, without an adaptive equalizer, the receiver is susceptible to random fluctuations of channel characteristics or hardware operations. A more robust and adaptive receiver is needed to combat such fluctuations.

Chapter 7

Conclusions

The transition to digital television is limited by the static functionality of hardware communication devices. Once a digital television is manufactured and deployed, existing functions cannot be modified and new ones cannot be added. This lack of flexibility results in DTV devices that cannot interoperate between standards and can become obsolete early. The primary objective of this work is to enable a flexible software DTV implementation that overcomes many limitations of current DTVs. To accomplish this goal, stages of DTV processing that rely heavily on analog techniques, namely, carrier and timing synchronization, must make the dual transition from analog to digital and from hardware to software.

In this thesis, we examined conventional analog solutions, stated their inefficiencies for software implementations, and proposed alternative approaches. For close phase-tracking, an open-loop, computationally-efficient algorithm was designed. A closed-loop, iterative technique delivers the high precision required for frequency synchronization. Finally, a discrete version of the ATSC method was proposed for symbol timing. In doing so, this thesis achieved the following objectives:

- devise algorithm structures with high efficiency,
- meet the particular requirements of carrier recovery and timing recovery, and
- provide more flexibility, less hardware dependency, and more universal alternatives to using customized hardware solutions

7.1 Future Work

The high failure rates observed in Chapter 6 motivates further investigation into more robust and adaptive receiver designs. To improve the probability of error, a receiver can be equipped with the capability to adaptively suppress interference. In most wireless systems, this involves equalization and forward error correction. Due to the additional complexity required to incorporate these features, we were not able to investigate these topics during the course of this thesis.

7.1.1 Adaptive Equalization

An adaptive equalizer compensates for linear distortions and intersymbol interference, caused by either channel effects or hardware imperfections. It is often realized in the form of a FIR (Finite Impulse Response) filter, whose filter coefficients are *adapted* to reflect changing channel characteristics [4]. The objective is to adjust the filter coefficients such that the noise and intersymbol interference at the equalizer output are eliminated. The adaptation of an equalizer is driven by an *error signal.*

In digital television, the generation of this error signal is facilitated by the known symbols of the data field sync. The equalizer can compute an error function from the known field sync and the received symbols, with the goal of minimizing this error function [2]. Alternatively, an error signal can also be generated independently of the field sync. An estimate of the transmitted signal is created by quantizing the data to the nearest level. The error signal is produced by subtracting the quantized symbols from the received symbols, with the intent of minimizing this quantization error. In the absence of interference or distortion, the received symbols would equal the quantized output, and the error signal would be zero.

7.1.2 Forward Error Correction

According to the ATSC standard, the video and audio packets must undergo several stages of coding prior to transmission. For each 188-byte packet, 20 Reed-Solomon parity bytes are added for error correction [1]. A convolutional interleaver then packages the data, such that a burst error occurring on the interleaved data will be dispersed among packets once de-interleaved. Finally, a Trellis coder adds redundancy by mapping each 2-bit symbol to 3-bits per symbol [1].

An ATSC digital receiver can be enhanced by incorporating a channel decoder to separate the redundancy from the information. Additional decoding, by inverting the mapping performed by the encoder, provides better receiver performance at the expense of implementation complexity. Since not all bit patterns are permitted by the code, the decoder can detect, and even correct, bit errors. Time did not permit the inclusion of this topic in this research.

7.2 Technology Trend

Historically, the development of signal processing algorithms has always been driven by the technology used to implement them. Digital communication techniques offer a clear technical advantage over analog formats, since any effects due to noise and distortion from the transmission medium can be completely removed. Despite the technical advantage of digital communications, the additional cost of analog-to-digital conversion and the corresponding digital-to-analog conversion was, not too long ago, an impediment to the widespread use of discrete signal processing algorithms. The study and development of digital algorithms were primarily a subject of research interest rather than of practical applications.

The advent of digital computers and digital ICs, however, not only rendered the expense associated with A/D and D/A insignificant, it also greatly reduced the cost of realizing complicated digital signal processing. Today, virtually all communications are either already digital, or under consideration for conversion. The availability of economical VLSI technology encourages development of discrete techniques, since they can be feasibly implemented.

The SpectrumWare approach, by performing all signal processing functions in application software, has a similar dependence on the advances in high-performance A/D technology and workstation processor speed. While the transition from analog to digital techniques is motivated by improved noise performance, SpectrumWare technology is motivated by the desire for increased flexibility. As demands for new communication services proliferate, this flexibility will become crucial.

SpectrumWare signal processing algorithms, especially those presented in this thesis, face radically different design constraints. As discussed throughout this thesis, they should exhibit high computational efficiency as well as exploit computational resources and development environments on conventional workstations. In addition, software-based algorithms should also be hardware-independent and, therefore, universal and portable. The techniques developed in this thesis were intentionally generalized to avoid any dependencies on specific hardware characteristics. For example, the timing recovery method was generalized for any sampling frequency, while the block size parameter in the phase-recovery structure can be readily modified as well.

As long as current A/D technology and processor speeds can only support a narrow scope of applications, the impetus for developing such software-friendly techniques will be limited. However, this trend is likely to change with the rapid evolution of computer technology. Innovations in processor architecture, such as extensions to instruction sets or higher clock speeds, are being developed every day. These advances allow software implementations of increasingly complex signal processing applications to meet the performance requirements of specific applications.

Bibliography

[1] Advanced Television Systems Committee, James C. McKinney, Chairman and Dr. Robert Hopkins, Executive Director. *ATSC Digital Television Standard.* 1995.

[2] Advanced Television Systems Committee, James C. McKinney, Chairman and Dr. Robert Hopkins, Executive Director. *Guide to the Use of the ATSC Digital Television Standard.* 1995.

[3] V. Bose, M. Ismert, M. Welborn and J. Guttag. *Virtual Radios.* 1998.

[4] A. Oppenheim and A. Willsky. *Signals & Systems.* 2nd ed. New Jersey: Prentice Hall, 1997. pp. 607-608.

[5] E. A. Lee and D. G. Messerschmitt. *Digital Communication.* 2nd ed. Massachusetts: Kluwer Academic Publishers, 1997.

[6] A. D. Houghton. *The Engineer's Error Coding Handbook.* United Kingdom: Chapman & Hall, 1997.

[7] M. Schwartz, B. R. Bennett, and S. Stein. *Communication Systems and Techniques.* New York: McGraw-Hill Book Company, 1966, pp. 192-193. Vol. 4 of Inter-University Electronics Series. 4 vols.

[8] Gary Sgrignoli. *ATSC Transmission System: VSB Tutorial.* http://www.zenith.com/main/cool/tech/vsbtutor.html

[9] L. E. Franks. *Carrier and Bit Synchronization in Data Communication - A Tutorial Review.* IEEE Transactions on Communications, 1980, pp. 1107-1121.

[10] Soheil I. Sayegh. *DSP MCD for Future IBS/IDR Services.*

[11] S. Lawrence Marple, Jr. *Digital Spectral Analysis with Applications.* New Jersey: Prentice-Hall, Inc., 1987.

[12] D. W. Tufts and P. D. Fiore. *Simple, Effective Estimation of Frequency Based on Prony's Method.* ICASSP-96 vol. 5, pages 2801-2804. IEEE 1996.

[13] Vanu Bose. *Design and Implementation of Software Radios Using a General Purpose Processor.* MIT PhD thesis, 1999.

[14] Steven P. Nicoloso. *An Investigation of Carrier Recovery Techniques for PSK Modulated Signals in CDMA and Multipath Mobile Environments.* Virgina Polytechnic Institute. Master of Science Thesis. 1997.

[15] Behzad Razavi. *Monolithic Phase-Locked Loops and Clock Recovery Circuits: Theory and Design.* IEEE Press. 1996.